

GenESyS - GENeric and Expansible SYstem Simulator
v200413 (Maibirth)

Generated by Doxygen 1.8.17

1 ReGenESys	1
2 Todo List	2
3 Namespace Index	2
3.1 Namespace List	2
4 Hierarchical Index	2
4.1 Class Hierarchy	2
5 Class Index	9
5.1 Class List	9
6 File Index	15
6.1 File List	15
7 Namespace Documentation	23
7.1 GenesysKernel Namespace Reference	23
8 Class Documentation	24
8.1 Access Class Reference	24
8.1.1 Detailed Description	27
8.1.2 Constructor & Destructor Documentation	27
8.1.3 Member Function Documentation	28
8.2 AnalysisOfVariance_if Class Reference	31
8.2.1 Detailed Description	31
8.2.2 Member Function Documentation	32
8.3 Assign Class Reference	32
8.3.1 Detailed Description	35
8.3.2 Constructor & Destructor Documentation	35
8.3.3 Member Function Documentation	36
8.4 Assign::Assignment Class Reference	41
8.4.1 Detailed Description	42
8.4.2 Constructor & Destructor Documentation	42
8.4.3 Member Function Documentation	42
8.5 Attribute Class Reference	44
8.5.1 Detailed Description	46
8.5.2 Constructor & Destructor Documentation	46
8.5.3 Member Function Documentation	47
8.6 BaseConsoleGenesysApplication Class Reference	50
8.6.1 Detailed Description	51
8.6.2 Constructor & Destructor Documentation	51
8.6.3 Member Function Documentation	51
8.7 Batch Class Reference	63
8.7.1 Detailed Description	66

8.7.2 Constructor & Destructor Documentation	66
8.7.3 Member Function Documentation	66
8.8 SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference	70
8.8.1 Detailed Description	71
8.8.2 Member Data Documentation	71
8.9 CelularAutomata Class Reference	72
8.9.1 Detailed Description	72
8.9.2 Constructor & Destructor Documentation	72
8.10 Collector_if Class Reference	73
8.10.1 Detailed Description	74
8.10.2 Member Function Documentation	74
8.11 CollectorDatafile_if Class Reference	77
8.11.1 Detailed Description	79
8.11.2 Member Function Documentation	79
8.12 CollectorDatafileDefaultImpl1 Class Reference	81
8.12.1 Detailed Description	83
8.12.2 Constructor & Destructor Documentation	83
8.12.3 Member Function Documentation	83
8.13 CollectorDefaultImpl1 Class Reference	86
8.13.1 Detailed Description	87
8.13.2 Constructor & Destructor Documentation	87
8.13.3 Member Function Documentation	88
8.14 ComponentManager Class Reference	89
8.14.1 Detailed Description	90
8.14.2 Constructor & Destructor Documentation	90
8.14.3 Member Function Documentation	91
8.15 ConnectionManager Class Reference	97
8.15.1 Detailed Description	98
8.15.2 Constructor & Destructor Documentation	98
8.15.3 Member Function Documentation	98
8.16 Counter Class Reference	109
8.16.1 Detailed Description	111
8.16.2 Constructor & Destructor Documentation	111
8.16.3 Member Function Documentation	112
8.17 Create Class Reference	117
8.17.1 Detailed Description	120
8.17.2 Constructor & Destructor Documentation	120
8.17.3 Member Function Documentation	122
8.18 Decide Class Reference	128
8.18.1 Detailed Description	131
8.18.2 Constructor & Destructor Documentation	131
8.18.3 Member Function Documentation	132

8.19 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference	137
8.19.1 Detailed Description	138
8.19.2 Constructor & Destructor Documentation	138
8.19.3 Member Data Documentation	138
8.20 Delay Class Reference	139
8.20.1 Detailed Description	142
8.20.2 Constructor & Destructor Documentation	142
8.20.3 Member Function Documentation	143
8.21 Dispose Class Reference	153
8.21.1 Detailed Description	155
8.21.2 Constructor & Destructor Documentation	155
8.21.3 Member Function Documentation	156
8.22 DropOff Class Reference	161
8.22.1 Detailed Description	163
8.22.2 Constructor & Destructor Documentation	163
8.22.3 Member Function Documentation	163
8.23 Dummy Class Reference	167
8.23.1 Detailed Description	170
8.23.2 Constructor & Destructor Documentation	170
8.23.3 Member Function Documentation	170
8.24 ElementManager Class Reference	174
8.24.1 Detailed Description	176
8.24.2 Constructor & Destructor Documentation	176
8.24.3 Member Function Documentation	176
8.25 ElementManager_if Class Reference	188
8.25.1 Detailed Description	189
8.25.2 Constructor & Destructor Documentation	189
8.26 Enter Class Reference	189
8.26.1 Detailed Description	192
8.26.2 Constructor & Destructor Documentation	193
8.26.3 Member Function Documentation	193
8.27 Entity Class Reference	200
8.27.1 Detailed Description	202
8.27.2 Constructor & Destructor Documentation	202
8.27.3 Member Function Documentation	203
8.28 EntityGroup Class Reference	211
8.28.1 Detailed Description	214
8.28.2 Constructor & Destructor Documentation	214
8.28.3 Member Function Documentation	215
8.29 EntityType Class Reference	222
8.29.1 Detailed Description	224
8.29.2 Constructor & Destructor Documentation	224

8.29.3 Member Function Documentation	225
8.30 Event Class Reference	233
8.30.1 Detailed Description	233
8.30.2 Constructor & Destructor Documentation	233
8.30.3 Member Function Documentation	234
8.31 Exit Class Reference	236
8.31.1 Detailed Description	239
8.31.2 Constructor & Destructor Documentation	239
8.31.3 Member Function Documentation	239
8.32 ExperimentDesign_if Class Reference	243
8.32.1 Detailed Description	244
8.32.2 Member Function Documentation	244
8.33 ExperimentDesignDefaultImpl1 Class Reference	245
8.33.1 Detailed Description	246
8.33.2 Constructor & Destructor Documentation	246
8.33.3 Member Function Documentation	247
8.34 ExperimentDesignDummyImpl Class Reference	248
8.34.1 Detailed Description	249
8.34.2 Constructor & Destructor Documentation	249
8.34.3 Member Function Documentation	250
8.35 ExperimentManager_if Class Reference	251
8.35.1 Detailed Description	252
8.35.2 Member Function Documentation	252
8.36 ExperimentManagerDefaultImpl1 Class Reference	254
8.36.1 Detailed Description	256
8.36.2 Constructor & Destructor Documentation	256
8.36.3 Member Function Documentation	256
8.37 FactorOrInteractionContribution Class Reference	258
8.37.1 Detailed Description	259
8.37.2 Constructor & Destructor Documentation	259
8.37.3 Member Function Documentation	259
8.38 Failure Class Reference	260
8.38.1 Detailed Description	263
8.38.2 Constructor & Destructor Documentation	263
8.38.3 Member Function Documentation	264
8.39 File Class Reference	267
8.39.1 Detailed Description	270
8.39.2 Constructor & Destructor Documentation	270
8.39.3 Member Function Documentation	271
8.40 Fitter_if Class Reference	274
8.40.1 Detailed Description	276
8.40.2 Member Function Documentation	276

8.41 FitterDefaultImpl1 Class Reference	278
8.41.1 Detailed Description	281
8.41.2 Constructor & Destructor Documentation	281
8.41.3 Member Function Documentation	281
8.42 Formula Class Reference	284
8.42.1 Detailed Description	286
8.42.2 Constructor & Destructor Documentation	286
8.42.3 Member Function Documentation	286
8.43 FullSimulationOfComplexModel Class Reference	293
8.43.1 Detailed Description	295
8.43.2 Constructor & Destructor Documentation	296
8.43.3 Member Function Documentation	296
8.44 ParserManager::GenerateNewParserResult Struct Reference	299
8.44.1 Detailed Description	299
8.44.2 Member Data Documentation	299
8.45 GenesysApplication_if Class Reference	300
8.45.1 Detailed Description	301
8.45.2 Member Function Documentation	301
8.46 GenesysConsole Class Reference	302
8.46.1 Detailed Description	303
8.46.2 Constructor & Destructor Documentation	304
8.46.3 Member Function Documentation	305
8.47 GenesysGUI Class Reference	315
8.47.1 Detailed Description	317
8.47.2 Constructor & Destructor Documentation	317
8.47.3 Member Function Documentation	317
8.48 GenesysShell_if Class Reference	318
8.48.1 Detailed Description	320
8.48.2 Member Function Documentation	320
8.49 Hold Class Reference	323
8.49.1 Detailed Description	326
8.49.2 Constructor & Destructor Documentation	326
8.49.3 Member Function Documentation	327
8.50 HypothesisTester_if Class Reference	331
8.50.1 Detailed Description	332
8.50.2 Member Enumeration Documentation	332
8.50.3 Member Function Documentation	332
8.51 HypothesisTesterBoostImpl Class Reference	333
8.51.1 Detailed Description	335
8.51.2 Constructor & Destructor Documentation	335
8.51.3 Member Function Documentation	336
8.52 HypothesisTesterDefaultImpl1 Class Reference	337

8.52.1 Detailed Description	339
8.52.2 Constructor & Destructor Documentation	340
8.52.3 Member Function Documentation	340
8.53 Integrator_if Class Reference	341
8.53.1 Detailed Description	343
8.53.2 Member Function Documentation	343
8.54 IntegratorDefaultImpl1 Class Reference	344
8.54.1 Detailed Description	346
8.54.2 Constructor & Destructor Documentation	346
8.54.3 Member Function Documentation	347
8.55 Leave Class Reference	349
8.55.1 Detailed Description	352
8.55.2 Constructor & Destructor Documentation	353
8.55.3 Member Function Documentation	353
8.56 LicenceManager Class Reference	359
8.56.1 Detailed Description	360
8.56.2 Constructor & Destructor Documentation	360
8.56.3 Member Function Documentation	360
8.57 LinkedBy Class Reference	363
8.57.1 Detailed Description	363
8.57.2 Constructor & Destructor Documentation	364
8.57.3 Member Function Documentation	364
8.58 List< T > Class Template Reference	365
8.58.1 Detailed Description	366
8.58.2 Member Typedef Documentation	366
8.58.3 Constructor & Destructor Documentation	366
8.58.4 Member Function Documentation	367
8.59 LSODE Class Reference	381
8.59.1 Detailed Description	383
8.59.2 Constructor & Destructor Documentation	383
8.59.3 Member Function Documentation	384
8.60 MarkovChain Class Reference	389
8.60.1 Detailed Description	391
8.60.2 Constructor & Destructor Documentation	391
8.60.3 Member Function Documentation	392
8.61 Match Class Reference	397
8.61.1 Detailed Description	400
8.61.2 Constructor & Destructor Documentation	400
8.61.3 Member Function Documentation	400
8.62 MathMeth Class Reference	404
8.62.1 Detailed Description	404
8.62.2 Constructor & Destructor Documentation	405

8.63 Model Class Reference	405
8.63.1 Detailed Description	406
8.63.2 Constructor & Destructor Documentation	407
8.63.3 Member Function Documentation	409
8.64 Model_AssignWrite3Seizes Class Reference	432
8.64.1 Detailed Description	433
8.64.2 Constructor & Destructor Documentation	434
8.64.3 Member Function Documentation	434
8.65 Model_CreateDelayDispose Class Reference	438
8.65.1 Detailed Description	439
8.65.2 Constructor & Destructor Documentation	440
8.65.3 Member Function Documentation	440
8.66 Model_CreateDelayDispose2 Class Reference	441
8.66.1 Detailed Description	443
8.66.2 Constructor & Destructor Documentation	444
8.66.3 Member Function Documentation	444
8.67 Model_SeizeDelayRelease1 Class Reference	446
8.67.1 Detailed Description	447
8.67.2 Constructor & Destructor Documentation	448
8.67.3 Member Function Documentation	448
8.68 Model_SeizeDelayReleaseMany Class Reference	451
8.68.1 Detailed Description	452
8.68.2 Constructor & Destructor Documentation	453
8.68.3 Member Function Documentation	453
8.69 Model_StationRouteSequence Class Reference	455
8.69.1 Detailed Description	456
8.69.2 Constructor & Destructor Documentation	457
8.69.3 Member Function Documentation	457
8.70 ModelChecker_if Class Reference	460
8.70.1 Detailed Description	461
8.70.2 Member Function Documentation	461
8.71 ModelCheckerDefaultImpl1 Class Reference	462
8.71.1 Detailed Description	463
8.71.2 Constructor & Destructor Documentation	463
8.71.3 Member Function Documentation	464
8.72 ModelComponent Class Reference	472
8.72.1 Detailed Description	473
8.72.2 Constructor & Destructor Documentation	473
8.72.3 Member Function Documentation	474
8.72.4 Member Data Documentation	486
8.73 ModelElement Class Reference	487
8.73.1 Detailed Description	489

8.73.2 Constructor & Destructor Documentation	489
8.73.3 Member Function Documentation	490
8.73.4 Member Data Documentation	506
8.74 ModelInfo Class Reference	508
8.74.1 Detailed Description	509
8.74.2 Constructor & Destructor Documentation	509
8.74.3 Member Function Documentation	509
8.75 ModelManager Class Reference	515
8.75.1 Detailed Description	515
8.75.2 Constructor & Destructor Documentation	515
8.75.3 Member Function Documentation	516
8.76 Modelo_SistemaOperacional02 Class Reference	522
8.76.1 Detailed Description	524
8.76.2 Constructor & Destructor Documentation	525
8.76.3 Member Function Documentation	525
8.77 ModelPersistence_if Class Reference	528
8.77.1 Detailed Description	529
8.77.2 Member Function Documentation	529
8.78 ModelPersistenceDefaultImpl1 Class Reference	530
8.78.1 Detailed Description	532
8.78.2 Constructor & Destructor Documentation	532
8.78.3 Member Function Documentation	532
8.79 ModelSimulation Class Reference	536
8.79.1 Detailed Description	538
8.79.2 Constructor & Destructor Documentation	538
8.79.3 Member Function Documentation	539
8.80 ParserManager::NewParser Struct Reference	558
8.80.1 Detailed Description	558
8.80.2 Member Data Documentation	558
8.81 ODEfunction Class Reference	559
8.81.1 Detailed Description	560
8.81.2 Constructor & Destructor Documentation	560
8.81.3 Member Data Documentation	560
8.82 OLD_ODElement Class Reference	561
8.82.1 Detailed Description	563
8.82.2 Constructor & Destructor Documentation	563
8.82.3 Member Function Documentation	563
8.83 OnEventManager Class Reference	568
8.83.1 Detailed Description	569
8.83.2 Constructor & Destructor Documentation	569
8.83.3 Member Function Documentation	569
8.84 Parser_if Class Reference	576

8.84.1 Detailed Description	578
8.84.2 Member Function Documentation	578
8.85 ParserChangesInformation Class Reference	579
8.85.1 Detailed Description	580
8.85.2 Member Typedef Documentation	580
8.85.3 Constructor & Destructor Documentation	581
8.85.4 Member Function Documentation	581
8.86 ParserDefaultImpl1 Class Reference	583
8.86.1 Detailed Description	584
8.86.2 Constructor & Destructor Documentation	584
8.86.3 Member Function Documentation	585
8.87 ParserManager Class Reference	586
8.87.1 Detailed Description	587
8.87.2 Constructor & Destructor Documentation	587
8.87.3 Member Function Documentation	587
8.88 PersistentObject_base Class Reference	588
8.88.1 Detailed Description	588
8.88.2 Constructor & Destructor Documentation	588
8.88.3 Member Function Documentation	589
8.89 PickStation Class Reference	590
8.89.1 Detailed Description	592
8.89.2 Constructor & Destructor Documentation	592
8.89.3 Member Function Documentation	593
8.90 PickUp Class Reference	596
8.90.1 Detailed Description	599
8.90.2 Constructor & Destructor Documentation	599
8.90.3 Member Function Documentation	599
8.91 Plugin Class Reference	603
8.91.1 Detailed Description	604
8.91.2 Constructor & Destructor Documentation	604
8.91.3 Member Function Documentation	604
8.92 PluginConnector_if Class Reference	606
8.92.1 Detailed Description	608
8.92.2 Member Function Documentation	608
8.93 PluginConnectorDummyImpl1 Class Reference	610
8.93.1 Detailed Description	611
8.93.2 Constructor & Destructor Documentation	611
8.93.3 Member Function Documentation	612
8.94 PluginInformation Class Reference	615
8.94.1 Detailed Description	616
8.94.2 Constructor & Destructor Documentation	616
8.94.3 Member Function Documentation	617

8.95 PluginManager Class Reference	625
8.95.1 Detailed Description	626
8.95.2 Constructor & Destructor Documentation	626
8.95.3 Member Function Documentation	626
8.96 ProbDistrib_if Class Reference	631
8.96.1 Detailed Description	633
8.96.2 Member Function Documentation	633
8.97 ProbDistribBoostImpl Class Reference	637
8.97.1 Detailed Description	639
8.97.2 Member Function Documentation	639
8.98 ProbDistribDefaultImpl1 Class Reference	644
8.98.1 Detailed Description	647
8.98.2 Member Function Documentation	647
8.99 Queue Class Reference	652
8.99.1 Detailed Description	655
8.99.2 Member Enumeration Documentation	655
8.99.3 Constructor & Destructor Documentation	656
8.99.4 Member Function Documentation	656
8.100 Record Class Reference	668
8.100.1 Detailed Description	670
8.100.2 Constructor & Destructor Documentation	670
8.100.3 Member Function Documentation	672
8.101 Release Class Reference	678
8.101.1 Detailed Description	681
8.101.2 Constructor & Destructor Documentation	681
8.101.3 Member Function Documentation	682
8.102 Remove Class Reference	688
8.102.1 Detailed Description	691
8.102.2 Constructor & Destructor Documentation	691
8.102.3 Member Function Documentation	691
8.103 RequirementTester Class Reference	695
8.103.1 Detailed Description	696
8.103.2 Constructor & Destructor Documentation	696
8.104 Resource Class Reference	696
8.104.1 Detailed Description	699
8.104.2 Member Typedef Documentation	700
8.104.3 Member Enumeration Documentation	700
8.104.4 Constructor & Destructor Documentation	700
8.104.5 Member Function Documentation	702
8.105 RNG_Parameters Class Reference	712
8.106 Sampler_if::RNG_Parameters Struct Reference	713
8.106.1 Detailed Description	714

8.106.2 Constructor & Destructor Documentation	714
8.107 Route Class Reference	714
8.107.1 Detailed Description	717
8.107.2 Member Enumeration Documentation	717
8.107.3 Constructor & Destructor Documentation	718
8.107.4 Member Function Documentation	718
8.108 Sampler_if Class Reference	728
8.108.1 Detailed Description	730
8.108.2 Member Function Documentation	731
8.109 SamplerBoostImpl Class Reference	733
8.109.1 Detailed Description	736
8.109.2 Constructor & Destructor Documentation	736
8.109.3 Member Function Documentation	736
8.110 SamplerDefaultImpl1 Class Reference	739
8.110.1 Detailed Description	742
8.110.2 Constructor & Destructor Documentation	742
8.110.3 Member Function Documentation	742
8.111 ScenarioExperiment_if Class Reference	750
8.111.1 Detailed Description	750
8.112 Schedule Class Reference	751
8.112.1 Detailed Description	753
8.112.2 Constructor & Destructor Documentation	753
8.112.3 Member Function Documentation	753
8.113 Search Class Reference	755
8.113.1 Detailed Description	758
8.113.2 Constructor & Destructor Documentation	758
8.113.3 Member Function Documentation	759
8.114 SeizableItemRequest Class Reference	762
8.114.1 Detailed Description	765
8.114.2 Member Enumeration Documentation	765
8.114.3 Constructor & Destructor Documentation	766
8.114.4 Member Function Documentation	766
8.115 Seize Class Reference	771
8.115.1 Detailed Description	773
8.115.2 Constructor & Destructor Documentation	774
8.115.3 Member Function Documentation	774
8.116 Sequence Class Reference	784
8.116.1 Detailed Description	787
8.116.2 Constructor & Destructor Documentation	787
8.116.3 Member Function Documentation	787
8.117 SequenceStep Class Reference	791
8.117.1 Detailed Description	792

8.117.2 Constructor & Destructor Documentation	792
8.117.3 Member Function Documentation	792
8.118 Set Class Reference	794
8.118.1 Detailed Description	796
8.118.2 Constructor & Destructor Documentation	796
8.118.3 Member Function Documentation	796
8.119 Signal Class Reference	801
8.119.1 Detailed Description	804
8.119.2 Constructor & Destructor Documentation	804
8.119.3 Member Function Documentation	804
8.120 SimulationControl Class Reference	808
8.120.1 Detailed Description	809
8.120.2 Constructor & Destructor Documentation	810
8.120.3 Member Function Documentation	810
8.121 SimulationEvent Class Reference	811
8.121.1 Detailed Description	811
8.121.2 Constructor & Destructor Documentation	811
8.121.3 Member Function Documentation	811
8.122 SimulationReporter_if Class Reference	813
8.122.1 Detailed Description	814
8.122.2 Member Function Documentation	814
8.123 SimulationReporterDefaultImpl1 Class Reference	815
8.123.1 Detailed Description	817
8.123.2 Constructor & Destructor Documentation	817
8.123.3 Member Function Documentation	818
8.124 SimulationResponse Class Reference	827
8.124.1 Detailed Description	828
8.124.2 Constructor & Destructor Documentation	829
8.124.3 Member Function Documentation	829
8.124.4 Member Data Documentation	830
8.125 SimulationScenario Class Reference	831
8.125.1 Detailed Description	832
8.125.2 Constructor & Destructor Documentation	832
8.125.3 Member Function Documentation	832
8.126 Simulator Class Reference	835
8.126.1 Detailed Description	835
8.126.2 Constructor & Destructor Documentation	835
8.126.3 Member Function Documentation	836
8.127 SinkModelComponent Class Reference	841
8.127.1 Detailed Description	843
8.127.2 Constructor & Destructor Documentation	844
8.127.3 Member Function Documentation	844

8.128 SourceModelComponent Class Reference	846
8.128.1 Detailed Description	848
8.128.2 Constructor & Destructor Documentation	848
8.128.3 Member Function Documentation	848
8.128.4 Member Data Documentation	859
8.129 Start Class Reference	860
8.129.1 Detailed Description	863
8.129.2 Constructor & Destructor Documentation	863
8.129.3 Member Function Documentation	863
8.130 Station Class Reference	867
8.130.1 Detailed Description	870
8.130.2 Constructor & Destructor Documentation	870
8.130.3 Member Function Documentation	871
8.131 Statistics_if Class Reference	879
8.131.1 Detailed Description	881
8.131.2 Member Function Documentation	881
8.132 StatisticsCollector Class Reference	887
8.132.1 Detailed Description	890
8.132.2 Constructor & Destructor Documentation	890
8.132.3 Member Function Documentation	891
8.133 StatisticsDatafile_if Class Reference	897
8.133.1 Detailed Description	898
8.133.2 Member Function Documentation	899
8.134 StatisticsDataFileDummyImpl Class Reference	900
8.134.1 Detailed Description	903
8.134.2 Constructor & Destructor Documentation	903
8.134.3 Member Function Documentation	903
8.135 StatisticsDefaultImpl1 Class Reference	908
8.135.1 Detailed Description	910
8.135.2 Constructor & Destructor Documentation	910
8.135.3 Member Function Documentation	911
8.136 Stop Class Reference	915
8.136.1 Detailed Description	917
8.136.2 Constructor & Destructor Documentation	917
8.136.3 Member Function Documentation	917
8.137 Storage Class Reference	921
8.137.1 Detailed Description	923
8.137.2 Constructor & Destructor Documentation	924
8.137.3 Member Function Documentation	924
8.138 Store Class Reference	928
8.138.1 Detailed Description	930
8.138.2 Constructor & Destructor Documentation	930

8.138.3 Member Function Documentation	930
8.139 Submodel Class Reference	934
8.139.1 Detailed Description	937
8.139.2 Constructor & Destructor Documentation	937
8.139.3 Member Function Documentation	937
8.140 TestEnterLeaveRoute Class Reference	941
8.140.1 Detailed Description	943
8.140.2 Constructor & Destructor Documentation	944
8.140.3 Member Function Documentation	944
8.141 TestMatricesOfAttributesAndVariables Class Reference	947
8.141.1 Detailed Description	949
8.141.2 Constructor & Destructor Documentation	949
8.141.3 Member Function Documentation	949
8.142 TestODE Class Reference	953
8.142.1 Detailed Description	954
8.142.2 Constructor & Destructor Documentation	955
8.142.3 Member Function Documentation	955
8.143 TestParser Class Reference	956
8.143.1 Detailed Description	957
8.143.2 Constructor & Destructor Documentation	957
8.143.3 Member Function Documentation	958
8.144 TestSimulationControlAndSimulationResponse Class Reference	959
8.144.1 Detailed Description	961
8.144.2 Constructor & Destructor Documentation	961
8.144.3 Member Function Documentation	961
8.145 ToolManager Class Reference	963
8.145.1 Detailed Description	963
8.145.2 Constructor & Destructor Documentation	963
8.145.3 Member Function Documentation	964
8.146 TraceErrorEvent Class Reference	965
8.146.1 Detailed Description	966
8.146.2 Constructor & Destructor Documentation	966
8.146.3 Member Function Documentation	966
8.147 TraceEvent Class Reference	967
8.147.1 Detailed Description	967
8.147.2 Constructor & Destructor Documentation	968
8.147.3 Member Function Documentation	968
8.148 TraceManager Class Reference	969
8.148.1 Detailed Description	971
8.148.2 Constructor & Destructor Documentation	971
8.148.3 Member Function Documentation	971
8.149 TraceSimulationEvent Class Reference	984

8.149.1 Detailed Description	986
8.149.2 Constructor & Destructor Documentation	986
8.149.3 Member Function Documentation	986
8.150 TraceSimulationProcess Class Reference	987
8.150.1 Detailed Description	988
8.150.2 Constructor & Destructor Documentation	988
8.151 Traits< T > Struct Template Reference	989
8.151.1 Detailed Description	989
8.152 GenesysKernel::Traits< T > Struct Template Reference	989
8.152.1 Detailed Description	990
8.153 Traits< Collector_if > Struct Reference	990
8.153.1 Detailed Description	990
8.153.2 Member Typedef Documentation	990
8.154 Traits< ExperimentDesign_if > Struct Reference	991
8.154.1 Detailed Description	991
8.154.2 Member Typedef Documentation	991
8.155 Traits< ExperimentManager_if > Struct Reference	991
8.155.1 Detailed Description	992
8.155.2 Member Typedef Documentation	992
8.156 Traits< Fitter_if > Struct Reference	992
8.156.1 Detailed Description	992
8.156.2 Member Typedef Documentation	993
8.157 Traits< GenesysApplication_if > Struct Reference	993
8.157.1 Detailed Description	993
8.157.2 Member Typedef Documentation	993
8.158 Traits< HypothesisTester_if > Struct Reference	994
8.158.1 Detailed Description	994
8.158.2 Member Typedef Documentation	994
8.159 Traits< Integrator_if > Struct Reference	995
8.159.1 Detailed Description	995
8.159.2 Member Typedef Documentation	995
8.159.3 Member Data Documentation	996
8.160 Traits< Model > Struct Reference	996
8.160.1 Detailed Description	997
8.160.2 Member Data Documentation	997
8.161 GenesysKernel::Traits< Model > Struct Reference	997
8.161.1 Detailed Description	998
8.161.2 Member Data Documentation	998
8.162 Traits< ModelChecker_if > Struct Reference	998
8.162.1 Detailed Description	999
8.162.2 Member Typedef Documentation	999
8.163 GenesysKernel::Traits< ModelChecker_if > Struct Reference	999

8.163.1 Detailed Description	999
8.163.2 Member Typedef Documentation	1000
8.164 GenesysKernel::Traits< ModelComponent > Struct Reference	1000
8.164.1 Detailed Description	1000
8.164.2 Member Typedef Documentation	1000
8.165 Traits< ModelComponent > Struct Reference	1001
8.165.1 Detailed Description	1002
8.165.2 Member Typedef Documentation	1002
8.165.3 Member Data Documentation	1002
8.166 Traits< ModelElement > Struct Reference	1003
8.166.1 Detailed Description	1003
8.166.2 Member Data Documentation	1003
8.167 GenesysKernel::Traits< ModelPersistence_if > Struct Reference	1004
8.167.1 Detailed Description	1004
8.167.2 Member Typedef Documentation	1004
8.168 Traits< ModelPersistence_if > Struct Reference	1005
8.168.1 Detailed Description	1005
8.168.2 Member Typedef Documentation	1005
8.169 Traits< Parser_if > Struct Reference	1005
8.169.1 Detailed Description	1006
8.169.2 Member Typedef Documentation	1006
8.170 GenesysKernel::Traits< Parser_if > Struct Reference	1006
8.170.1 Detailed Description	1006
8.170.2 Member Typedef Documentation	1007
8.171 GenesysKernel::Traits< PluginConnector_if > Struct Reference	1007
8.171.1 Detailed Description	1007
8.171.2 Member Typedef Documentation	1007
8.172 Traits< PluginConnector_if > Struct Reference	1008
8.172.1 Detailed Description	1008
8.172.2 Member Typedef Documentation	1008
8.173 Traits< ProbDistrib_if > Struct Reference	1009
8.173.1 Detailed Description	1009
8.173.2 Member Typedef Documentation	1009
8.174 Traits< Sampler_if > Struct Reference	1009
8.174.1 Detailed Description	1010
8.174.2 Member Typedef Documentation	1010
8.175 GenesysKernel::Traits< Sampler_if > Struct Reference	1010
8.175.1 Detailed Description	1011
8.175.2 Member Typedef Documentation	1011
8.176 GenesysKernel::Traits< SimulationReporter_if > Struct Reference	1011
8.176.1 Detailed Description	1012
8.176.2 Member Typedef Documentation	1012

8.177 Traits< SimulationReporter_if > Struct Reference	1012
8.177.1 Detailed Description	1012
8.177.2 Member Typedef Documentation	1013
8.178 Traits< Statistics_if > Struct Reference	1013
8.178.1 Detailed Description	1014
8.178.2 Member Typedef Documentation	1014
8.178.3 Member Data Documentation	1014
8.179 Unstore Class Reference	1015
8.179.1 Detailed Description	1017
8.179.2 Constructor & Destructor Documentation	1017
8.179.3 Member Function Documentation	1017
8.180 Util Class Reference	1021
8.180.1 Detailed Description	1023
8.180.2 Member Typedef Documentation	1023
8.180.3 Member Enumeration Documentation	1023
8.180.4 Member Function Documentation	1025
8.180.5 Member Data Documentation	1034
8.181 Variable Class Reference	1034
8.181.1 Detailed Description	1037
8.181.2 Constructor & Destructor Documentation	1037
8.181.3 Member Function Documentation	1038
8.182 Waiting Class Reference	1046
8.182.1 Detailed Description	1047
8.182.2 Constructor & Destructor Documentation	1047
8.182.3 Member Function Documentation	1047
8.183 WaitingResource Class Reference	1049
8.183.1 Detailed Description	1051
8.183.2 Constructor & Destructor Documentation	1051
8.183.3 Member Function Documentation	1052
8.184 Write Class Reference	1053
8.184.1 Detailed Description	1055
8.184.2 Member Enumeration Documentation	1055
8.184.3 Constructor & Destructor Documentation	1055
8.184.4 Member Function Documentation	1056
8.185 WriteElement Class Reference	1063
8.185.1 Detailed Description	1064
8.185.2 Constructor & Destructor Documentation	1064
8.185.3 Member Data Documentation	1065
9 File Documentation	1065
9.1 .dep.inc File Reference	1065
9.2 .dep.inc	1065

9.3 Access.cpp File Reference	1066
9.4 Access.cpp	1066
9.5 Access.h File Reference	1067
9.6 Access.h	1068
9.7 AnalysisOfVariance_if.h File Reference	1068
9.8 AnalysisOfVariance_if.h	1068
9.9 Assign.cpp File Reference	1069
9.10 Assign.cpp	1069
9.11 Assign.h File Reference	1070
9.12 Assign.h	1071
9.13 Attribute.cpp File Reference	1072
9.14 Attribute.cpp	1072
9.15 Attribute.h File Reference	1073
9.16 Attribute.h	1074
9.17 BaseConsoleGenesysApplication.cpp File Reference	1074
9.18 BaseConsoleGenesysApplication.cpp	1075
9.19 BaseConsoleGenesysApplication.h File Reference	1077
9.20 BaseConsoleGenesysApplication.h	1077
9.21 Batch.cpp File Reference	1078
9.22 Batch.cpp	1078
9.23 Batch.h File Reference	1079
9.24 Batch.h	1080
9.25 CellularAutomata.cpp File Reference	1081
9.26 CellularAutomata.cpp	1081
9.27 CellularAutomata.h File Reference	1081
9.28 CellularAutomata.h	1082
9.29 Collector_if.h File Reference	1082
9.29.1 Typedef Documentation	1083
9.29.2 Function Documentation	1083
9.30 Collector_if.h	1084
9.31 CollectorDatafile_if.h File Reference	1085
9.32 CollectorDatafile_if.h	1085
9.33 CollectorDatafileDefaultImpl1.cpp File Reference	1086
9.34 CollectorDatafileDefaultImpl1.cpp	1086
9.35 CollectorDatafileDefaultImpl1.h File Reference	1087
9.36 CollectorDatafileDefaultImpl1.h	1088
9.37 CollectorDefaultImpl1.cpp File Reference	1088
9.38 CollectorDefaultImpl1.cpp	1089
9.39 CollectorDefaultImpl1.h File Reference	1090
9.40 CollectorDefaultImpl1.h	1090
9.41 ComponentManager.cpp File Reference	1091
9.42 ComponentManager.cpp	1091

9.43 ComponentManager.h File Reference	1092
9.44 ComponentManager.h	1093
9.45 ConnectionManager.cpp File Reference	1094
9.46 ConnectionManager.cpp	1094
9.47 ConnectionManager.h File Reference	1095
9.47.1 Typedef Documentation	1096
9.48 ConnectionManager.h	1096
9.49 Counter.cpp File Reference	1097
9.50 Counter.cpp	1097
9.51 Counter.h File Reference	1099
9.52 Counter.h	1099
9.53 Create.cpp File Reference	1100
9.54 Create.cpp	1100
9.55 Create.h File Reference	1102
9.56 Create.h	1102
9.57 Decide.cpp File Reference	1103
9.58 Decide.cpp	1103
9.59 Decide.h File Reference	1105
9.60 Decide.h	1105
9.61 DefaultTraceAndEventHandlers.h File Reference	1106
9.62 DefaultTraceAndEventHandlers.h	1106
9.63 DefineGetterSetter.h File Reference	1106
9.63.1 Typedef Documentation	1108
9.63.2 Function Documentation	1108
9.64 DefineGetterSetter.h	1111
9.65 Delay.cpp File Reference	1113
9.66 Delay.cpp	1113
9.67 Delay.h File Reference	1115
9.68 Delay.h	1115
9.69 Dispose.cpp File Reference	1116
9.70 Dispose.cpp	1116
9.71 Dispose.h File Reference	1118
9.72 Dispose.h	1118
9.73 DropOff.cpp File Reference	1119
9.74 DropOff.cpp	1119
9.75 DropOff.h File Reference	1120
9.76 DropOff.h	1121
9.77 Dummy.cpp File Reference	1121
9.78 Dummy.cpp	1122
9.79 Dummy.h File Reference	1123
9.80 Dummy.h	1123
9.81 ElementManager.cpp File Reference	1124

9.82 ElementManager.cpp	1124
9.83 ElementManager.h File Reference	1127
9.84 ElementManager.h	1127
9.85 ElementManager_if.h File Reference	1128
9.86 ElementManager_if.h	1128
9.87 Enter.cpp File Reference	1129
9.88 Enter.cpp	1129
9.89 Enter.h File Reference	1130
9.90 Enter.h	1131
9.91 Entity.cpp File Reference	1132
9.92 Entity.cpp	1132
9.93 Entity.h File Reference	1134
9.94 Entity.h	1135
9.95 EntityGroup.cpp File Reference	1136
9.96 EntityGroup.cpp	1136
9.97 EntityGroup.h File Reference	1137
9.98 EntityGroup.h	1138
9.99 EntityType.cpp File Reference	1139
9.100 EntityType.cpp	1139
9.101 EntityType.h File Reference	1141
9.102 EntityType.h	1142
9.103 Event.cpp File Reference	1143
9.104 Event.cpp	1143
9.105 Event.h File Reference	1144
9.106 Event.h	1144
9.107 Exit.cpp File Reference	1145
9.108 Exit.cpp	1145
9.109 Exit.h File Reference	1146
9.110 Exit.h	1147
9.111 ExperimentDesign_if.h File Reference	1147
9.112 ExperimentDesign_if.h	1148
9.113 ExperimentDesignDefaultImpl1.cpp File Reference	1149
9.114 ExperimentDesignDefaultImpl1.cpp	1149
9.115 ExperimentDesignDefaultImpl1.h File Reference	1149
9.116 ExperimentDesignDefaultImpl1.h	1150
9.117 ExperimentDesignDummyImpl.cpp File Reference	1151
9.118 ExperimentDesignDummyImpl.cpp	1151
9.119 ExperimentDesignDummyImpl.h File Reference	1151
9.120 ExperimentDesignDummyImpl.h	1152
9.121 ExperimentManagerDefaultImpl1.cpp File Reference	1153
9.122 ExperimentManagerDefaultImpl1.cpp	1153
9.123 ExperimentManagerDefaultImpl1.h File Reference	1154

9.124 ExperimentManagerDefaultImpl1.h	1155
9.125 ExperimentManager_if.h File Reference	1155
9.126 ExperimentManager_if.h	1156
9.127 FactorOrInteractionContribution.cpp File Reference	1156
9.128 FactorOrInteractionContribution.cpp	1157
9.129 FactorOrInteractionContribution.h File Reference	1157
9.130 FactorOrInteractionContribution.h	1158
9.131 Failure.cpp File Reference	1159
9.132 Failure.cpp	1159
9.133 Failure.h File Reference	1160
9.134 Failure.h	1161
9.135 File.cpp File Reference	1162
9.136 File.cpp	1162
9.137 File.h File Reference	1163
9.138 File.h	1164
9.139 Fitter_if.h File Reference	1164
9.140 Fitter_if.h	1165
9.141 FitterDefaultImpl1.cpp File Reference	1165
9.142 FitterDefaultImpl1.cpp	1166
9.143 FitterDefaultImpl1.h File Reference	1167
9.144 FitterDefaultImpl1.h	1168
9.145 Formula.cpp File Reference	1168
9.146 Formula.cpp	1168
9.147 Formula.h File Reference	1170
9.148 Formula.h	1171
9.149 FullSimulationOfComplexModel.cpp File Reference	1171
9.150 FullSimulationOfComplexModel.cpp	1172
9.151 FullSimulationOfComplexModel.h File Reference	1173
9.152 FullSimulationOfComplexModel.h	1174
9.153 Functor.h File Reference	1174
9.154 Functor.h	1174
9.155 GenesysApplication_if.h File Reference	1175
9.156 GenesysApplication_if.h	1175
9.157 GenesysConsole.cpp File Reference	1175
9.158 GenesysConsole.cpp	1176
9.159 GenesysConsole.h File Reference	1179
9.160 GenesysConsole.h	1179
9.161 GenesysGUI.cpp File Reference	1180
9.162 GenesysGUI.cpp	1181
9.163 GenesysGUI.h File Reference	1181
9.164 GenesysGUI.h	1182
9.165 GenesysShell_if.h File Reference	1183

9.166 GenesysShell_if.h	1183
9.167 Hold.cpp File Reference	1184
9.168 Hold.cpp	1184
9.169 Hold.h File Reference	1185
9.170 Hold.h	1186
9.171 HypothesisTester_if.h File Reference	1187
9.172 HypothesisTester_if.h	1187
9.173 HypothesisTesterBoostImpl.cpp File Reference	1188
9.174 HypothesisTesterBoostImpl.cpp	1188
9.175 HypothesisTesterBoostImpl.h File Reference	1189
9.176 HypothesisTesterBoostImpl.h	1190
9.177 HypothesisTesterDefaultImpl1.cpp File Reference	1190
9.178 HypothesisTesterDefaultImpl1.cpp	1190
9.179 HypothesisTesterDefaultImpl1.h File Reference	1191
9.180 HypothesisTesterDefaultImpl1.h	1192
9.181 Integrator_if.h File Reference	1192
9.182 Integrator_if.h	1192
9.183 IntegratorDefaultImpl1.cpp File Reference	1193
9.184 IntegratorDefaultImpl1.cpp	1193
9.185 IntegratorDefaultImpl1.h File Reference	1194
9.186 IntegratorDefaultImpl1.h	1195
9.187 Leave.cpp File Reference	1196
9.188 Leave.cpp	1196
9.189 Leave.h File Reference	1197
9.190 Leave.h	1198
9.191 LicenceManager.cpp File Reference	1198
9.192 LicenceManager.cpp	1199
9.193 LicenceManager.h File Reference	1200
9.194 LicenceManager.h	1201
9.195 LinkedBy.cpp File Reference	1201
9.196 LinkedBy.cpp	1201
9.197 LinkedBy.h File Reference	1202
9.198 LinkedBy.h	1202
9.199 List.h File Reference	1203
9.200 List.h	1204
9.201 LSODE.cpp File Reference	1207
9.202 LSODE.cpp	1207
9.203 LSODE.h File Reference	1209
9.204 LSODE.h	1210
9.205 main.cpp File Reference	1210
9.205.1 Function Documentation	1211
9.206 main.cpp	1212

9.207 MarkovChain.cpp File Reference	1212
9.208 MarkovChain.cpp	1212
9.209 MarkovChain.h File Reference	1214
9.210 MarkovChain.h	1215
9.211 Match.cpp File Reference	1216
9.212 Match.cpp	1216
9.213 Match.h File Reference	1217
9.214 Match.h	1218
9.215 MathMeth.cpp File Reference	1218
9.216 MathMeth.cpp	1219
9.217 MathMeth.h File Reference	1219
9.218 MathMeth.h	1220
9.219 Model.cpp File Reference	1220
9.219.1 Function Documentation	1221
9.220 Model.cpp	1221
9.221 Model.h File Reference	1226
9.222 Model.h	1227
9.223 Model_AssignWrite3Seizes.cpp File Reference	1228
9.224 Model_AssignWrite3Seizes.cpp	1229
9.225 Model_AssignWrite3Seizes.h File Reference	1231
9.226 Model_AssignWrite3Seizes.h	1231
9.227 Model_CreateDelayDispose.cpp File Reference	1232
9.228 Model_CreateDelayDispose.cpp	1232
9.229 Model_CreateDelayDispose.h File Reference	1233
9.230 Model_CreateDelayDispose.h	1234
9.231 Model_CreateDelayDispose2.cpp File Reference	1234
9.232 Model_CreateDelayDispose2.cpp	1234
9.233 Model_CreteDelayDispose2.h File Reference	1235
9.234 Model_CreteDelayDispose2.h	1236
9.235 Model_SeizeDelayRelease1.cpp File Reference	1237
9.236 Model_SeizeDelayRelease1.cpp	1237
9.237 Model_SeizeDelayRelease1.h File Reference	1238
9.238 Model_SeizeDelayRelease1.h	1239
9.239 Model_SeizeDelayReleaseMany.cpp File Reference	1239
9.240 Model_SeizeDelayReleaseMany.cpp	1240
9.241 Model_SeizeDelayReleaseMany.h File Reference	1240
9.242 Model_SeizeDelayReleaseMany.h	1241
9.243 Model_StatationRouteSequence.cpp File Reference	1242
9.244 Model_StatationRouteSequence.cpp	1242
9.245 Model_StatationRouteSequence.h File Reference	1243
9.246 Model_StatationRouteSequence.h	1244
9.247 ModelChecker_if.h File Reference	1245

9.248 ModelChecker_if.h	1245
9.249 ModelCheckerDefaultImpl1.cpp File Reference	1246
9.250 ModelCheckerDefaultImpl1.cpp	1246
9.251 ModelCheckerDefaultImpl1.h File Reference	1249
9.252 ModelCheckerDefaultImpl1.h	1249
9.253 ModelComponent.cpp File Reference	1250
9.254 ModelComponent.cpp	1250
9.255 ModelComponent.h File Reference	1252
9.256 ModelComponent.h	1253
9.257 ModelElement.cpp File Reference	1253
9.258 ModelElement.cpp	1254
9.259 ModelElement.h File Reference	1256
9.260 ModelElement.h	1257
9.261 ModellInfo.cpp File Reference	1258
9.262 ModellInfo.cpp	1258
9.263 ModellInfo.h File Reference	1260
9.264 ModellInfo.h	1260
9.265 ModelManager.cpp File Reference	1261
9.266 ModelManager.cpp	1261
9.267 ModelManager.h File Reference	1262
9.268 ModelManager.h	1263
9.269 Modelo_SistemaOperacional02.cpp File Reference	1264
9.270 Modelo_SistemaOperacional02.cpp	1264
9.271 Modelo_SistemaOperacional02.h File Reference	1266
9.272 Modelo_SistemaOperacional02.h	1266
9.273 ModelPersistence_if.h File Reference	1267
9.274 ModelPersistence_if.h	1267
9.275 ModelPersistenceDefaultImpl1.cpp File Reference	1268
9.276 ModelPersistenceDefaultImpl1.cpp	1268
9.277 ModelPersistenceDefaultImpl1.h File Reference	1273
9.278 ModelPersistenceDefaultImpl1.h	1273
9.279 ModelSimulation.cpp File Reference	1274
9.280 ModelSimulation.cpp	1274
9.281 ModelSimulation.h File Reference	1282
9.282 ModelSimulation.h	1283
9.283 OLD_ODEelement.cpp File Reference	1285
9.284 OLD_ODEelement.cpp	1285
9.285 OLD_ODEelement.h File Reference	1286
9.286 OLD_ODEelement.h	1287
9.287 OnEventManager.cpp File Reference	1288
9.288 OnEventManager.cpp	1288
9.289 OnEventManager.h File Reference	1290

9.289.1 Typedef Documentation	1290
9.290 OnEventManager.h	1291
9.291 Parser_if.h File Reference	1292
9.292 Parser_if.h	1293
9.293 ParserChangesInformation.cpp File Reference	1294
9.294 ParserChangesInformation.cpp	1294
9.295 ParserChangesInformation.h File Reference	1295
9.296 ParserChangesInformation.h	1296
9.297 ParserDefaultImpl1.cpp File Reference	1296
9.298 ParserDefaultImpl1.cpp	1297
9.299 ParserDefaultImpl1.h File Reference	1298
9.300 ParserDefaultImpl1.h	1299
9.301 ParserManager.cpp File Reference	1299
9.302 ParserManager.cpp	1300
9.303 ParserManager.h File Reference	1300
9.304 ParserManager.h	1301
9.305 PersistentObject_base.h File Reference	1301
9.306 PersistentObject_base.h	1302
9.307 PickStation.cpp File Reference	1302
9.308 PickStation.cpp	1303
9.309 PickStation.h File Reference	1303
9.310 PickStation.h	1304
9.311 PickUp.cpp File Reference	1305
9.312 PickUp.cpp	1305
9.313 PickUp.h File Reference	1306
9.314 PickUp.h	1307
9.315 Plugin.cpp File Reference	1307
9.316 Plugin.cpp	1308
9.317 Plugin.h File Reference	1309
9.318 Plugin.h	1310
9.319 PluginConnector_if.h File Reference	1310
9.320 PluginConnector_if.h	1311
9.321 PluginConnectorDummyImpl1.cpp File Reference	1311
9.322 PluginConnectorDummyImpl1.cpp	1312
9.323 PluginConnectorDummyImpl1.h File Reference	1314
9.324 PluginConnectorDummyImpl1.h	1315
9.325 PluginInformation.cpp File Reference	1316
9.326 PluginInformation.cpp	1316
9.327 PluginInformation.h File Reference	1318
9.327.1 Typedef Documentation	1319
9.328 PluginInformation.h	1319
9.329 PluginManager.cpp File Reference	1320

9.330 PluginManager.cpp	1321
9.331 PluginManager.h File Reference	1323
9.332 PluginManager.h	1323
9.333 Prob_Distrib_if.h File Reference	1324
9.334 Prob_Distrib_if.h	1324
9.335 ProbDistribBoostImpl.cpp File Reference	1325
9.336 ProbDistribBoostImpl.cpp	1325
9.337 ProbDistribBoostImpl.h File Reference	1326
9.338 ProbDistribBoostImpl.h	1327
9.339 ProbDistribDefaultImpl1.cpp File Reference	1328
9.340 ProbDistribDefaultImpl1.cpp	1328
9.341 ProbDistribDefaultImpl1.h File Reference	1329
9.342 ProbDistribDefaultImpl1.h	1330
9.343 Queue.cpp File Reference	1331
9.344 Queue.cpp	1331
9.345 Queue.h File Reference	1333
9.346 Queue.h	1334
9.347 README.md File Reference	1335
9.348 Record.cpp File Reference	1335
9.349 Record.cpp	1336
9.350 Record.h File Reference	1337
9.351 Record.h	1338
9.352 Release.cpp File Reference	1339
9.353 Release.cpp	1339
9.354 Release.h File Reference	1341
9.355 Release.h	1342
9.356 Remove.cpp File Reference	1343
9.357 Remove.cpp	1343
9.358 Remove.h File Reference	1344
9.359 Remove.h	1345
9.360 RequirementTester.cpp File Reference	1346
9.361 RequirementTester.cpp	1346
9.362 RequirementTester.h File Reference	1346
9.363 RequirementTester.h	1347
9.364 Resource.cpp File Reference	1347
9.365 Resource.cpp	1347
9.366 Resource.h File Reference	1350
9.367 Resource.h	1350
9.368 Route.cpp File Reference	1352
9.369 Route.cpp	1352
9.370 Route.h File Reference	1355
9.371 Route.h	1355

9.372 Sampler_if.h File Reference	1356
9.373 Sampler_if.h	1356
9.374 SamplerBoostImpl.cpp File Reference	1357
9.375 SamplerBoostImpl.cpp	1357
9.376 SamplerBoostImpl.h File Reference	1358
9.377 SamplerBoostImpl.h	1359
9.378 SamplerDefaultImpl1.cpp File Reference	1360
9.379 SamplerDefaultImpl1.cpp	1360
9.380 SamplerDefaultImpl1.h File Reference	1362
9.381 SamplerDefaultImpl1.h	1363
9.382 ScenarioExperiment_if.h File Reference	1364
9.383 ScenarioExperiment_if.h	1364
9.384 Schedule.cpp File Reference	1364
9.385 Schedule.cpp	1365
9.386 Schedule.h File Reference	1365
9.387 Schedule.h	1366
9.388 Search.cpp File Reference	1366
9.389 Search.cpp	1367
9.390 Search.h File Reference	1368
9.391 Search.h	1369
9.392 SeizableItemRequest.cpp File Reference	1369
9.393 SeizableItemRequest.cpp	1369
9.394 SeizableItemRequest.h File Reference	1371
9.395 SeizableItemRequest.h	1372
9.396 Seize.cpp File Reference	1373
9.397 Seize.cpp	1373
9.398 Seize.h File Reference	1376
9.399 Seize.h	1377
9.400 Sequence.cpp File Reference	1378
9.401 Sequence.cpp	1378
9.402 Sequence.h File Reference	1379
9.403 Sequence.h	1380
9.404 Set.cpp File Reference	1381
9.405 Set.cpp	1381
9.406 Set.h File Reference	1383
9.407 Set.h	1383
9.408 Signal.cpp File Reference	1384
9.409 Signal.cpp	1384
9.410 Signal.h File Reference	1385
9.411 Signal.h	1386
9.412 SimulationControl.cpp File Reference	1387
9.413 SimulationControl.cpp	1387

9.414 SimulationControl.h File Reference	1387
9.415 SimulationControl.h	1388
9.416 SimulationReporter_if.h File Reference	1389
9.417 SimulationReporter_if.h	1389
9.418 SimulationReporterDefaultImpl1.cpp File Reference	1390
9.419 SimulationReporterDefaultImpl1.cpp	1390
9.420 SimulationReporterDefaultImpl1.h File Reference	1394
9.421 SimulationReporterDefaultImpl1.h	1394
9.422 SimulationResponse.cpp File Reference	1395
9.423 SimulationResponse.cpp	1395
9.424 SimulationResponse.h File Reference	1396
9.425 SimulationResponse.h	1397
9.426 SimulationScenario.cpp File Reference	1397
9.427 SimulationScenario.cpp	1398
9.428 SimulationScenario.h File Reference	1399
9.429 SimulationScenario.h	1400
9.430 Simulator.cpp File Reference	1400
9.430.1 Function Documentation	1401
9.431 Simulator.cpp	1402
9.432 Simulator.h File Reference	1403
9.432.1 Typedef Documentation	1403
9.433 Simulator.h	1404
9.434 SinkModelComponent.cpp File Reference	1404
9.435 SinkModelComponent.cpp	1405
9.436 SinkModelComponent.h File Reference	1405
9.437 SinkModelComponent.h	1406
9.438 SourceModelComponent.cpp File Reference	1407
9.439 SourceModelComponent.cpp	1407
9.440 SourceModelComponent.h File Reference	1409
9.441 SourceModelComponent.h	1409
9.442 Start.cpp File Reference	1410
9.443 Start.cpp	1410
9.444 Start.h File Reference	1411
9.445 Start.h	1412
9.446 Station.cpp File Reference	1413
9.447 Station.cpp	1413
9.448 Station.h File Reference	1415
9.449 Station.h	1416
9.450 Statistics_if.h File Reference	1417
9.451 Statistics_if.h	1418
9.452 StatisticsCollector.cpp File Reference	1418
9.452.1 Typedef Documentation	1418

9.453 StatisticsCollector.cpp	1419
9.454 StatisticsCollector.h File Reference	1420
9.455 StatisticsCollector.h	1421
9.456 StatisticsDataFile_if.h File Reference	1421
9.457 StatisticsDataFile_if.h	1423
9.458 StatisticsDataFileDefaultImpl.cpp File Reference	1423
9.459 StatisticsDataFileDefaultImpl.cpp	1423
9.460 StatisticsDataFileDefaultImpl.h File Reference	1424
9.461 StatisticsDataFileDefaultImpl.h	1426
9.462 StatisticsDefaultImpl1.cpp File Reference	1426
9.463 StatisticsDefaultImpl1.cpp	1426
9.464 StatisticsDefaultImpl1.h File Reference	1428
9.465 StatisticsDefaultImpl1.h	1429
9.466 Stop.cpp File Reference	1430
9.467 Stop.cpp	1430
9.468 Stop.h File Reference	1431
9.469 Stop.h	1432
9.470 Storage.cpp File Reference	1433
9.471 Storage.cpp	1433
9.472 Storage.h File Reference	1434
9.473 Storage.h	1435
9.474 Store.cpp File Reference	1436
9.475 Store.cpp	1436
9.476 Store.h File Reference	1437
9.477 Store.h	1438
9.478 Submodel.cpp File Reference	1438
9.479 Submodel.cpp	1438
9.480 Submodel.h File Reference	1439
9.481 Submodel.h	1440
9.482 TestEnterLeaveRoute.cpp File Reference	1441
9.483 TestEnterLeaveRoute.cpp	1441
9.484 TestEnterLeaveRoute.h File Reference	1443
9.485 TestEnterLeaveRoute.h	1444
9.486 TestFunctions.cpp File Reference	1444
9.486.1 Typedef Documentation	1445
9.486.2 Function Documentation	1445
9.486.3 Variable Documentation	1445
9.487 TestFunctions.cpp	1446
9.488 TestFunctions.h File Reference	1447
9.489 TestFunctions.h	1447
9.490 TestMatricesOfAttributesAndVariables.cpp File Reference	1448
9.491 TestMatricesOfAttributesAndVariables.cpp	1448

9.492 TestMatricesOfAttributesAndVariables.h File Reference	1450
9.493 TestMatricesOfAttributesAndVariables.h	1450
9.494 TestParser.cpp File Reference	1451
9.495 TestParser.cpp	1451
9.496 TestParser.h File Reference	1452
9.497 TestParser.h	1452
9.498 TestSimulationControlAndSimulationResponse.cpp File Reference	1453
9.499 TestSimulationControlAndSimulationResponse.cpp	1453
9.500 TestSimulationControlAndSimulationResponse.h File Reference	1454
9.501 TestSimulationControlAndSimulationResponse.h	1455
9.502 ToolManager.cpp File Reference	1455
9.503 ToolManager.cpp	1455
9.504 ToolManager.h File Reference	1456
9.505 ToolManager.h	1457
9.506 TraceManager.cpp File Reference	1457
9.507 TraceManager.cpp	1457
9.508 TraceManager.h File Reference	1459
9.508.1 Typedef Documentation	1460
9.509 TraceManager.h	1461
9.510 Traits.h File Reference	1463
9.511 Traits.h	1465
9.512 TraitsKernel.h File Reference	1467
9.513 TraitsKernel.h	1468
9.514 Unstore.cpp File Reference	1469
9.515 Unstore.cpp	1470
9.516 Unstore.h File Reference	1471
9.517 Unstore.h	1472
9.518 Util.cpp File Reference	1472
9.519 Util.cpp	1472
9.520 Util.h File Reference	1474
9.520.1 Function Documentation	1475
9.521 Util.h	1479
9.522 Variable.cpp File Reference	1481
9.523 Variable.cpp	1481
9.524 Variable.h File Reference	1483
9.525 Variable.h	1484
9.526 Write.cpp File Reference	1484
9.527 Write.cpp	1485
9.528 Write.h File Reference	1487
9.529 Write.h	1487

1 ReGenESyS

Reborn Generic and Expansible System Simulator

Genesys is a result of teaching and research activities of Professor Dr. Ing Rafael Luiz Cancian. It began in early 2002 as a way to teach students the basics and simulation techniques of systems implemented by other commercial simulation tools, such as Arena. In Genesys development he replicated all the SIMAN language, used by Arena software, and Genesys has become a clone of that tool, including its graphical interface. Genesys allowed the inclusion of new simulation components through dynamic link libraries and also the parallel execution of simulation models in a distributed environment. The development of Genesys continued until 2007, when the professor stopped teaching systems simulation classes. Ten years later the professor starts again to teach systems simulation classes and to carry out scientific research in the area. So in 2018 Genesys is reborn as ReGenesys, with new language and programming techniques, and even more ambitious goals.

Developed by [rlcancian](#)

2 Todo List

Member [PluginConnectorDummyImpl1::check](#) (const std::string dynamicLibraryFilename)

:To implement

Member [PluginInformation::PluginInformation](#) (std::string pluginTypename, [StaticLoaderComponent](#)←
Instance componentloader)

:

Class [TraceSimulationProcess](#)

: CLASS NOT FULLY IMPLEMENTED (to be implemented for process analyser)

3 Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[GenesysKernel](#)

23

4 Hierarchical Index

4.1 Class Hierarchy

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

[AnalysisOfVariance_if](#)

31

[Assign::Assignment](#)

41

[CelularAutomata](#)

72

Collector_if	73
CollectorDatafile_if	77
CollectorDatafileDefaultImpl1	81
CollectorDefaultImpl1	86
ComponentManager	89
ConnectionManager	97
ElementManager	174
ElementManager_if	188
Event	233
ExperimentDesign_if	243
ExperimentDesignDefaultImpl1	245
ExperimentDesignDummyImpl	248
ExperimentManager_if	251
ExperimentManagerDefaultImpl1	254
FactorOrInteractionContribution	258
Fitter_if	274
FitterDefaultImpl1	278
ParserManager::GenerateNewParserResult	299
GenesysApplication_if	300
BaseConsoleGenesysApplication	50
FullSimulationOfComplexModel	293
Model_AssignWrite3Seizes	432
Model_CreateDelayDispose	438
Model_CreateDelayDispose2	441
Model_SeizeDelayRelease1	446
Model_SeizeDelayReleaseMany	451
Model_StatationRouteSequence	455
Modelo_SistemaOperacional02	522
TestEnterLeaveRoute	941
TestMatricesOfAttributesAndVariables	947
TestODE	953
TestSimulationControlAndSimulationResponse	959

GenesysConsole	302
GenesysGUI	315
GenesysShell_if	318
TestParser	956
HypothesisTester_if	331
HypothesisTesterBoostImpl	333
HypothesisTesterDefaultImpl1	337
Integrator_if	341
IntegratorDefaultImpl1	344
LicenceManager	359
LinkedBy	363
List< T >	365
List< Assign::Assignment * >	365
List< Connection * >	365
List< double >	365
List< Entity * >	365
List< Event * >	365
List< Model * >	365
List< ModelComponent * >	365
List< ModelElement * >	365
List< ODEfunction * >	365
List< Plugin * >	365
List< ResourceEventHandler >	365
List< SeizableItemRequest * >	365
List< SequenceStep * >	365
List< ShellCommand * >	365
List< SimulationControl * >	365
List< simulationEventHandler >	365
List< simulationEventHandlerMethod >	365
List< SimulationResponse * >	365
List< StatisticsCollector * >	365
List< std::map< std::string, double > * >	365

List< std::string >	365
List< traceErrorListener >	365
List< traceErrorListenerMethod >	365
List< traceListener >	365
List< traceListenerMethod >	365
List< traceSimulationListener >	365
List< traceSimulationListenerMethod >	365
List< unsigned int >	365
List< Waiting * >	365
List< WriteElement * >	365
MathMeth	404
Model	405
ModelChecker_if	460
ModelCheckerDefaultImpl1	462
ModellInfo	508
ModelManager	515
ModelPersistence_if	528
ModelPersistenceDefaultImpl1	530
ModelSimulation	536
ParserManager::NewParser	558
ODEfunction	559
OnEventManager	568
Parser_if	576
ParserDefaultImpl1	583
ParserChangesInformation	579
ParserManager	586
PersistentObject_base	588
ModelElement	487
Attribute	44
Counter	109
Entity	200
EntityGroup	211

EntityType	222
Failure	260
File	267
Formula	284
ModelComponent	472
Access	24
Assign	32
Batch	63
Decide	128
Delay	139
DropOff	161
Dummy	167
Enter	189
Exit	236
Hold	323
Leave	349
LSODE	381
MarkovChain	389
Match	397
PickStation	590
PickUp	596
Record	668
Release	678
Remove	688
Route	714
Search	755
Seize	771
Signal	801
SinkModelComponent	841
Dispose	153
SourceModelComponent	846
Create	117

Start	860
Stop	915
Store	928
Submodel	934
Unstore	1015
Write	1053
OLD_ODElement	561
Queue	652
Resource	696
Schedule	751
Sequence	784
Set	794
Station	867
StatisticsCollector	887
Storage	921
Variable	1034
SeizableItemRequest	762
Plugin	603
PluginConnector_if	606
PluginConnectorDummyImpl1	610
PluginInformation	615
PluginManager	625
ProbDistrib_if	631
ProbDistribBoostImpl	637
ProbDistribDefaultImpl1	644
RequirementTester	695
RNG_Parameters	712
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters	137
Sampler_if::RNG_Parameters	713
SamplerBoostImpl::BoostImplRNG_Parameters	70
Sampler_if	728
SamplerBoostImpl	733

SamplerDefaultImpl1	739
ScenarioExperiment_if	750
SequenceStep	791
SimulationEvent	811
SimulationReporter_if	813
SimulationReporterDefaultImpl1	815
SimulationResponse	827
SimulationControl	808
SimulationScenario	831
Simulator	835
Statistics_if	879
StatisticsDatafile_if	897
StatisticsDataFileDummyImpl	900
StatisticsDefaultImpl1	908
ToolManager	963
TraceEvent	967
TraceErrorEvent	965
TraceSimulationEvent	984
TraceSimulationProcess	987
TraceManager	969
Traits< T >	989
GenesysKernel::Traits< T >	989
Traits< Collector_if >	990
Traits< ExperimentDesign_if >	991
Traits< ExperimentManager_if >	991
Traits< Fitter_if >	992
Traits< GenesysApplication_if >	993
Traits< HypothesisTester_if >	994
Traits< Integrator_if >	995
Traits< Model >	996
GenesysKernel::Traits< Model >	997
Traits< ModelChecker_if >	998

GenesysKernel::Traits< ModelChecker_if >	999
GenesysKernel::Traits< ModelComponent >	1000
Traits< ModelComponent >	1001
Traits< ModelElement >	1003
GenesysKernel::Traits< ModelPersistence_if >	1004
Traits< ModelPersistence_if >	1005
Traits< Parser_if >	1005
GenesysKernel::Traits< Parser_if >	1006
GenesysKernel::Traits< PluginConnector_if >	1007
Traits< PluginConnector_if >	1008
Traits< ProbDistrib_if >	1009
Traits< Sampler_if >	1009
GenesysKernel::Traits< Sampler_if >	1010
GenesysKernel::Traits< SimulationReporter_if >	1011
Traits< SimulationReporter_if >	1012
Traits< Statistics_if >	1013
Util	1021
Waiting	1046
WaitingResource	1049
WriteElement	1063

5 Class Index

5.1 Class List

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

Access	24
AnalysisOfVariance_if	31
Assign	32
Assign::Assignment	41
Attribute	44
BaseConsoleGenesysApplication	50
Batch	63

SamplerBoostImpl::BoostImplRNG_Parameters	70
CelularAutomata	72
Collector_if	73
CollectorDatafile_if	77
CollectorDatafileDefaultImpl1	81
CollectorDefaultImpl1	86
ComponentManager	89
ConnectionManager	97
Counter	109
Create	117
Decide	128
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters	137
Delay	139
Dispose	153
DropOff	161
Dummy	167
ElementManager	174
ElementManager_if	188
Enter	189
Entity	200
EntityGroup	211
EntityType	222
Event	233
Exit	236
ExperimentDesign_if	243
ExperimentDesignDefaultImpl1	245
ExperimentDesignDummyImpl	248
ExperimentManager_if	251
ExperimentManagerDefaultImpl1	254
FactorOrInteractionContribution	258
Failure	260
File	267

Fitter_if	274
FitterDefaultImpl1	278
Formula	284
FullSimulationOfComplexModel	293
ParserManager::GenerateNewParserResult	299
GenesysApplication_if	300
GenesysConsole	302
GenesysGUI	315
GenesysShell_if	318
Hold	323
HypothesisTester_if	331
HypothesisTesterBoostImpl	333
HypothesisTesterDefaultImpl1	337
Integrator_if	341
IntegratorDefaultImpl1	344
Leave	349
LicenceManager	359
LinkedBy	363
List< T >	365
LSODE	381
MarkovChain	389
Match	397
MathMeth	404
Model	405
Model_AssignWrite3Seizes	432
Model_CreateDelayDispose	438
Model_CreateDelayDispose2	441
Model_SeizeDelayRelease1	446
Model_SeizeDelayReleaseMany	451
Model_StationRouteSequence	455
ModelChecker_if	460
ModelCheckerDefaultImpl1	462

ModelComponent	472
ModelElement	487
ModelInfo	508
ModelManager	515
Modelo_SistemaOperacional02	522
ModelPersistence_if	528
ModelPersistenceDefaultImpl1	530
ModelSimulation	536
ParserManager::NewParser	558
ODEfunction	559
OLD_ODElement	561
OnEventManager	568
Parser_if	576
ParserChangesInformation	579
ParserDefaultImpl1	583
ParserManager	586
PersistentObject_base	588
PickStation	590
PickUp	596
Plugin	603
PluginConnector_if	606
PluginConnectorDummyImpl1	610
PluginInformation	615
PluginManager	625
ProbDistrib_if	631
ProbDistribBoostImpl	637
ProbDistribDefaultImpl1	644
Queue	652
Record	668
Release	678
Remove	688
RequirementTester	695

Resource	696
RNG_Parameters	712
Sampler_if::RNG_Parameters	713
Route	714
Sampler_if	728
SamplerBoostImpl	733
SamplerDefaultImpl1	739
ScenarioExperiment_if	750
Schedule	751
Search	755
SeizableItemRequest	762
Seize	771
Sequence	784
SequenceStep	791
Set	794
Signal	801
SimulationControl	808
SimulationEvent	811
SimulationReporter_if	813
SimulationReporterDefaultImpl1	815
SimulationResponse	827
SimulationScenario	831
Simulator	835
SinkModelComponent	841
SourceModelComponent	846
Start	860
Station	867
Statistics_if	879
StatisticsCollector	887
StatisticsDatafile_if	897
StatisticsDataFileDummyImpl	900
StatisticsDefaultImpl1	908

Stop	915
Storage	921
Store	928
Submodel	934
TestEnterLeaveRoute	941
TestMatricesOfAttributesAndVariables	947
TestODE	953
TestParser	956
TestSimulationControlAndSimulationResponse	959
ToolManager	963
TraceErrorEvent	965
TraceEvent	967
TraceManager	969
TraceSimulationEvent	984
TraceSimulationProcess	987
Traits< T >	989
GenesysKernel::Traits< T >	989
Traits< Collector_if >	990
Traits< ExperimentDesign_if >	991
Traits< ExperimentManager_if >	991
Traits< Fitter_if >	992
Traits< GenesysApplication_if >	993
Traits< HypothesisTester_if >	994
Traits< Integrator_if >	995
Traits< Model >	996
GenesysKernel::Traits< Model >	997
Traits< ModelChecker_if >	998
GenesysKernel::Traits< ModelChecker_if >	999
GenesysKernel::Traits< ModelComponent >	1000
Traits< ModelComponent >	1001
Traits< ModelElement >	1003
GenesysKernel::Traits< ModelPersistence_if >	1004

Traits< ModelPersistence_if >	1005
Traits< Parser_if >	1005
GenesysKernel::Traits< Parser_if >	1006
GenesysKernel::Traits< PluginConnector_if >	1007
Traits< PluginConnector_if >	1008
Traits< ProbDistrib_if >	1009
Traits< Sampler_if >	1009
GenesysKernel::Traits< Sampler_if >	1010
GenesysKernel::Traits< SimulationReporter_if >	1011
Traits< SimulationReporter_if >	1012
Traits< Statistics_if >	1013
Unstore	1015
Util	1021
Variable	1034
Waiting	1046
WaitingResource	1049
Write	1053
WriteElement	1063

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

.dep.inc	1065
Access.cpp	1066
Access.h	1067
AnalysisOfVariance_if.h	1068
Assign.cpp	1069
Assign.h	1070
Attribute.cpp	1072
Attribute.h	1073
BaseConsoleGenesysApplication.cpp	1074

BaseConsoleGenesysApplication.h	1077
Batch.cpp	1078
Batch.h	1079
CellularAutomata.cpp	1081
CellularAutomata.h	1081
Collector_if.h	1082
CollectorDatafile_if.h	1085
CollectorDatafileDefaultImpl1.cpp	1086
CollectorDatafileDefaultImpl1.h	1087
CollectorDefaultImpl1.cpp	1088
CollectorDefaultImpl1.h	1090
ComponentManager.cpp	1091
ComponentManager.h	1092
ConnectionManager.cpp	1094
ConnectionManager.h	1095
Counter.cpp	1097
Counter.h	1099
Create.cpp	1100
Create.h	1102
Decide.cpp	1103
Decide.h	1105
DefaultTraceAndEventHandlers.h	1106
DefineGetterSetter.h	1106
Delay.cpp	1113
Delay.h	1115
Dispose.cpp	1116
Dispose.h	1118
DropOff.cpp	1119
DropOff.h	1120
Dummy.cpp	1121
Dummy.h	1123
ElementManager.cpp	1124

ElementManager.h	1127
ElementManager_if.h	1128
Enter.cpp	1129
Enter.h	1130
Entity.cpp	1132
Entity.h	1134
EntityGroup.cpp	1136
EntityGroup.h	1137
EntityType.cpp	1139
EntityType.h	1141
Event.cpp	1143
Event.h	1144
Exit.cpp	1145
Exit.h	1146
ExperimentDesign_if.h	1147
ExperimentDesignDefaultImpl1.cpp	1149
ExperimentDesignDefaultImpl1.h	1149
ExperimentDesignDummyImpl.cpp	1151
ExperimentDesignDummyImpl.h	1151
ExperimentManagerDefaultImpl1.cpp	1153
ExperimentManagerDefaultImpl1.h	1154
ExperimentManager_if.h	1155
FactorOrInteractionContribution.cpp	1156
FactorOrInteractionContribution.h	1157
Failure.cpp	1159
Failure.h	1160
File.cpp	1162
File.h	1163
Fitter_if.h	1164
FitterDefaultImpl1.cpp	1165
FitterDefaultImpl1.h	1167
Formula.cpp	1168

Formula.h	1170
FullSimulationOfComplexModel.cpp	1171
FullSimulationOfComplexModel.h	1173
Functor.h	1174
GenesysApplication_if.h	1175
GenesysConsole.cpp	1175
GenesysConsole.h	1179
GenesysGUI.cpp	1180
GenesysGUI.h	1181
GenesysShell_if.h	1183
Hold.cpp	1184
Hold.h	1185
HypothesisTester_if.h	1187
HypothesisTesterBoostImpl.cpp	1188
HypothesisTesterBoostImpl.h	1189
HypothesisTesterDefaultImpl1.cpp	1190
HypothesisTesterDefaultImpl1.h	1191
Integrator_if.h	1192
IntegratorDefaultImpl1.cpp	1193
IntegratorDefaultImpl1.h	1194
Leave.cpp	1196
Leave.h	1197
LicenceManager.cpp	1198
LicenceManager.h	1200
LinkedBy.cpp	1201
LinkedBy.h	1202
List.h	1203
LSODE.cpp	1207
LSODE.h	1209
main.cpp	1210
MarkovChain.cpp	1212
MarkovChain.h	1214

Match.cpp	1216
Match.h	1217
MathMeth.cpp	1218
MathMeth.h	1219
Model.cpp	1220
Model.h	1226
Model_AssignWrite3Seizes.cpp	1228
Model_AssignWrite3Seizes.h	1231
Model_CreateDelayDispose.cpp	1232
Model_CreateDelayDispose.h	1233
Model_CreateDelayDispose2.cpp	1234
Model_CreteDelayDispose2.h	1235
Model_SeizeDelayRelease1.cpp	1237
Model_SeizeDelayRelease1.h	1238
Model_SeizeDelayReleaseMany.cpp	1239
Model_SeizeDelayReleaseMany.h	1240
Model_StatationRouteSequence.cpp	1242
Model_StatationRouteSequence.h	1243
ModelChecker_if.h	1245
ModelCheckerDefaultImpl1.cpp	1246
ModelCheckerDefaultImpl1.h	1249
ModelComponent.cpp	1250
ModelComponent.h	1252
ModelElement.cpp	1253
ModelElement.h	1256
ModellInfo.cpp	1258
ModellInfo.h	1260
ModelManager.cpp	1261
ModelManager.h	1262
Modelo_SistemaOperacional02.cpp	1264
Modelo_SistemaOperacional02.h	1266
ModelPersistence_if.h	1267

ModelPersistenceDefaultImpl1.cpp	1268
ModelPersistenceDefaultImpl1.h	1273
ModelSimulation.cpp	1274
ModelSimulation.h	1282
OLD_ODElement.cpp	1285
OLD_ODElement.h	1286
OnEventManager.cpp	1288
OnEventManager.h	1290
Parser_if.h	1292
ParserChangesInformation.cpp	1294
ParserChangesInformation.h	1295
ParserDefaultImpl1.cpp	1296
ParserDefaultImpl1.h	1298
ParserManager.cpp	1299
ParserManager.h	1300
PersistentObject_base.h	1301
PickStation.cpp	1302
PickStation.h	1303
PickUp.cpp	1305
PickUp.h	1306
Plugin.cpp	1307
Plugin.h	1309
PluginConnector_if.h	1310
PluginConnectorDummyImpl1.cpp	1311
PluginConnectorDummyImpl1.h	1314
PluginInformation.cpp	1316
PluginInformation.h	1318
PluginManager.cpp	1320
PluginManager.h	1323
Prob_Distrib_if.h	1324
ProbDistribBoostImpl.cpp	1325
ProbDistribBoostImpl.h	1326

ProbDistribDefaultImpl1.cpp	1328
ProbDistribDefaultImpl1.h	1329
Queue.cpp	1331
Queue.h	1333
Record.cpp	1335
Record.h	1337
Release.cpp	1339
Release.h	1341
Remove.cpp	1343
Remove.h	1344
RequirementTester.cpp	1346
RequirementTester.h	1346
Resource.cpp	1347
Resource.h	1350
Route.cpp	1352
Route.h	1355
Sampler_if.h	1356
SamplerBoostImpl.cpp	1357
SamplerBoostImpl.h	1358
SamplerDefaultImpl1.cpp	1360
SamplerDefaultImpl1.h	1362
ScenarioExperiment_if.h	1364
Schedule.cpp	1364
Schedule.h	1365
Search.cpp	1366
Search.h	1368
SeizableItemRequest.cpp	1369
SeizableItemRequest.h	1371
Seize.cpp	1373
Seize.h	1376
Sequence.cpp	1378
Sequence.h	1379

Set.cpp	1381
Set.h	1383
Signal.cpp	1384
Signal.h	1385
SimulationControl.cpp	1387
SimulationControl.h	1387
SimulationReporter_if.h	1389
SimulationReporterDefaultImpl1.cpp	1390
SimulationReporterDefaultImpl1.h	1394
SimulationResponse.cpp	1395
SimulationResponse.h	1396
SimulationScenario.cpp	1397
SimulationScenario.h	1399
Simulator.cpp	1400
Simulator.h	1403
SinkModelComponent.cpp	1404
SinkModelComponent.h	1405
SourceModelComponent.cpp	1407
SourceModelComponent.h	1409
Start.cpp	1410
Start.h	1411
Station.cpp	1413
Station.h	1415
Statistics_if.h	1417
StatisticsCollector.cpp	1418
StatisticsCollector.h	1420
StatisticsDataFile_if.h	1421
StatisticsDataFileDefaultImpl.cpp	1423
StatisticsDataFileDefaultImpl.h	1424
StatisticsDefaultImpl1.cpp	1426
StatisticsDefaultImpl1.h	1428
Stop.cpp	1430

Stop.h	1431
Storage.cpp	1433
Storage.h	1434
Store.cpp	1436
Store.h	1437
Submodel.cpp	1438
Submodel.h	1439
TestEnterLeaveRoute.cpp	1441
TestEnterLeaveRoute.h	1443
TestFunctions.cpp	1444
TestFunctions.h	1447
TestMatricesOfAttributesAndVariables.cpp	1448
TestMatricesOfAttributesAndVariables.h	1450
TestParser.cpp	1451
TestParser.h	1452
TestSimulationControlAndSimulationResponse.cpp	1453
TestSimulationControlAndSimulationResponse.h	1454
ToolManager.cpp	1455
ToolManager.h	1456
TraceManager.cpp	1457
TraceManager.h	1459
Traits.h	1463
TraitsKernel.h	1467
Unstore.cpp	1469
Unstore.h	1471
Util.cpp	1472
Util.h	1474
Variable.cpp	1481
Variable.h	1483
Write.cpp	1484
Write.h	1487

7 Namespace Documentation

7.1 GenesysKernel Namespace Reference

Classes

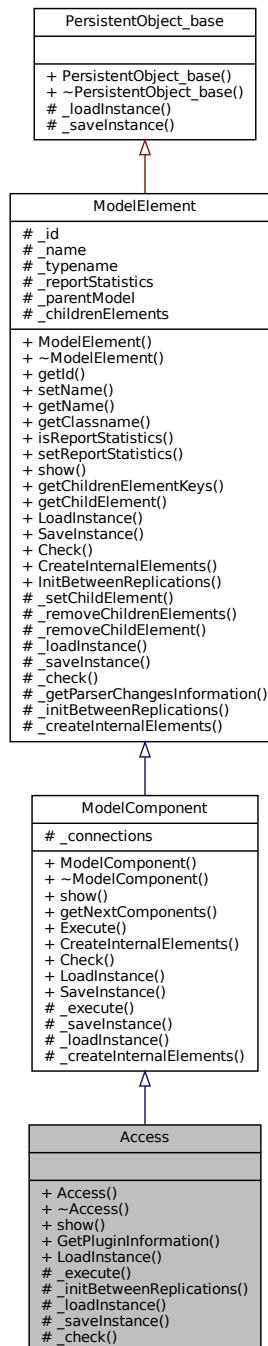
- struct [Traits](#)
- struct [Traits< Model >](#)
- struct [Traits< ModelChecker_if >](#)
- struct [Traits< ModelComponent >](#)
- struct [Traits< ModelPersistence_if >](#)
- struct [Traits< Parser_if >](#)
- struct [Traits< PluginConnector_if >](#)
- struct [Traits< Sampler_if >](#)
- struct [Traits< SimulationReporter_if >](#)

8 Class Documentation

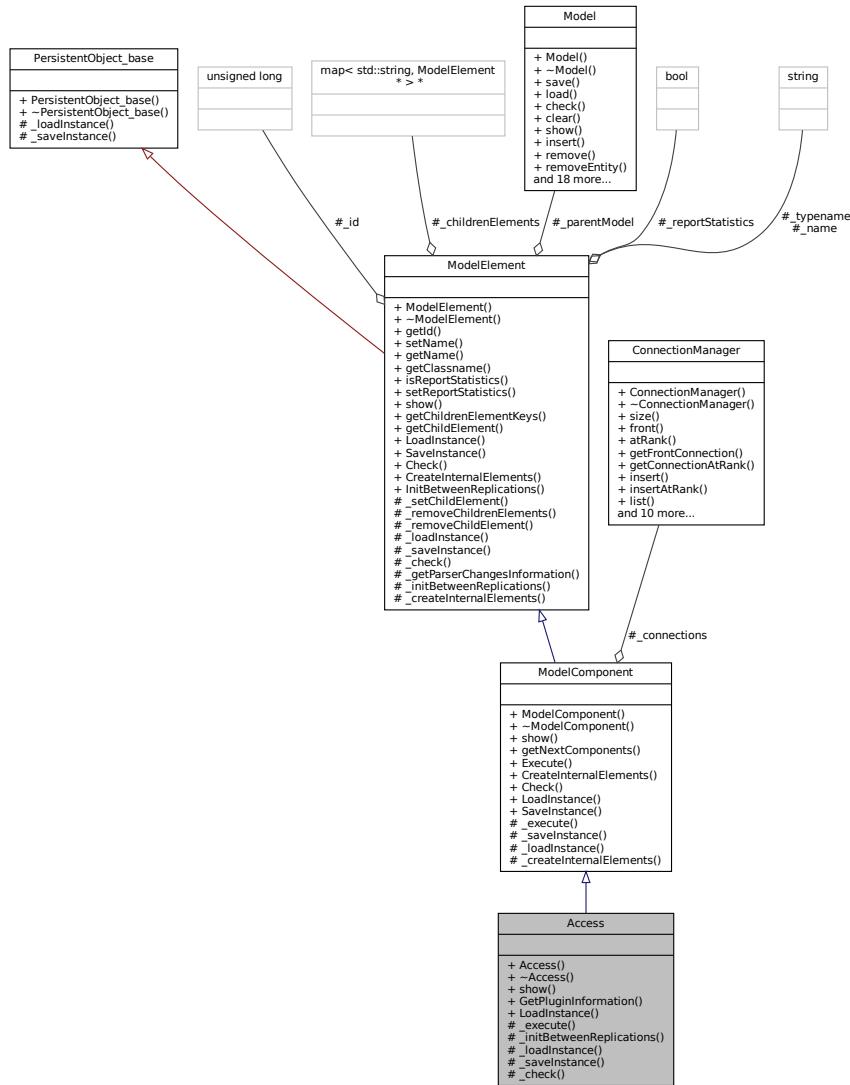
8.1 Access Class Reference

```
#include <Access.h>
```

Inheritance diagram for Access:



Collaboration diagram for Access:



Public Member Functions

- `Access (Model *model, std::string name="")`
- virtual `~Access ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.1.1 Detailed Description

Access module DESCRIPTION The [Access](#) module allocates one or more cells of a conveyor to an entity for movement from one station to another. Once the entity has control of the cells on the conveyor, it may then be conveyed to the next station. When an entity arrives at an [Access](#) module, it will wait until the appropriate number of contiguous cells on the conveyor are empty and aligned with the entity's station location. TYPICAL USES Parts accessing a conveyor to be sent to a paint booth Glass accessing a conveyor to be transferred to a cutting station PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor that the entity desires.

of Cells Number of contiguous conveyor cells the entity requires for

movement on the conveyor. **Queue** Type Determines the type of queue used to hold the entities, either an individual [Queue](#), a queue [Set](#), and Internal queue or an [Attribute](#) or Expression that evaluate to the queue name. **Queue Name** Name of the queue that will hold the entity until it accesses the conveyor. **Set Name** Name of the set of queues. **Set Index** Defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entries for a queue set containing three members is an expression that evaluates to 1, 2, or 3. **Attribute Name** Defines the name of the attribute that stores the queue name to which entities will reside. Expression Defines the name of the expression that stores the queue name to which entities will reside.

Definition at line 53 of file [Access.h](#).

8.1.2 Constructor & Destructor Documentation

```
8.1.2.1 Access() Access::Access (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Access.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Access>(), name) {
00019 }
```

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

Here is the caller graph for this function:



8.1.2.2 ~Access() virtual Access::~Access () [virtual], [default]

8.1.3 Member Function Documentation

8.1.3.1 `_check()` `bool Access::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Access.cpp](#).

```
00057                                     {  
00058     bool resultAll = true;  
00059     //...  
00060     return resultAll;  
00061 }
```

8.1.3.2 `_execute()` `void Access::_execute (Entity * entity) [protected], [virtual]`

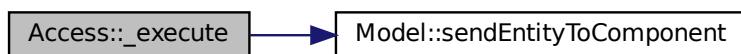
Implements [ModelComponent](#).

Definition at line 35 of file [Access.cpp](#).

```
00035                                     {  
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);  
00038 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.1.3.3 `_initBetweenReplications()` `void Access::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Access.cpp](#).

```
00048                                     {  
00049 }
```

```
8.1.3.4 __loadInstance() bool Access::__loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Access.cpp](#).

```
00040
00041     bool res = ModelComponent::__loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

References [ModelComponent::__loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.1.3.5 __saveInstance() std::map< std::string, std::string > * Access::__saveInstance ( ) [protected], [virtual]
```

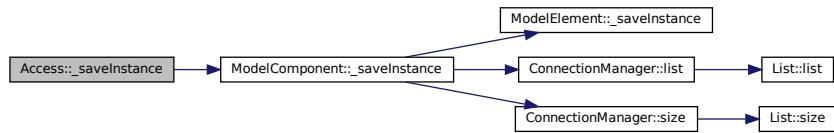
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Access.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::__saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::__saveInstance\(\)](#).

Here is the call graph for this function:



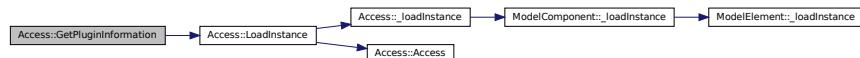
8.1.3.6 GetPluginInformation() `PluginInformation * Access::GetPluginInformation () [static]`

Definition at line 63 of file [Access.cpp](#).

```
00063 {  
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Access>(), &Access::LoadInstance);  
00065     // ...  
00066     return info;  
00067 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.1.3.7 LoadInstance() `ModelComponent * Access::LoadInstance (`

```
Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

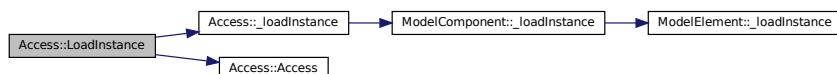
Definition at line 25 of file [Access.cpp](#).

```
00025 {  
00026     Access* newComponent = new Access(model);  
00027     try {  
00028         newComponent->_loadInstance(fields);  
00029     } catch (const std::exception& e) {  
00030     }  
00031     return newComponent;  
00032 }
```

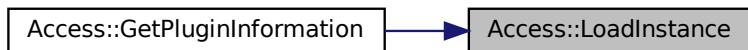
References [_loadInstance\(\)](#), and [Access\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.3.8 `show()` std::string Access::show () [virtual]

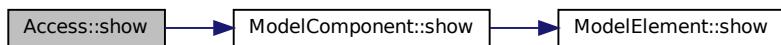
Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Access.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



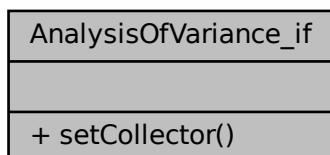
The documentation for this class was generated from the following files:

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

8.2 AnalysisOfVariance_if Class Reference

```
#include <AnalysisOfVariance_if.h>
```

Collaboration diagram for AnalysisOfVariance_if:



Public Member Functions

- virtual void [setCollector \(\)=0](#)

8.2.1 Detailed Description

Definition at line 17 of file [AnalysisOfVariance_if.h](#).

8.2.2 Member Function Documentation

8.2.2.1 setCollector() virtual void AnalysisOfVariance_if::setCollector () [pure virtual]

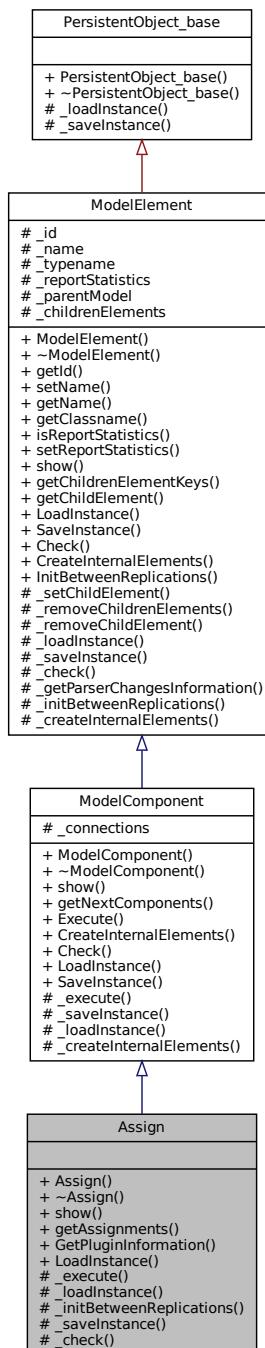
The documentation for this class was generated from the following file:

- [AnalysisOfVariance_if.h](#)

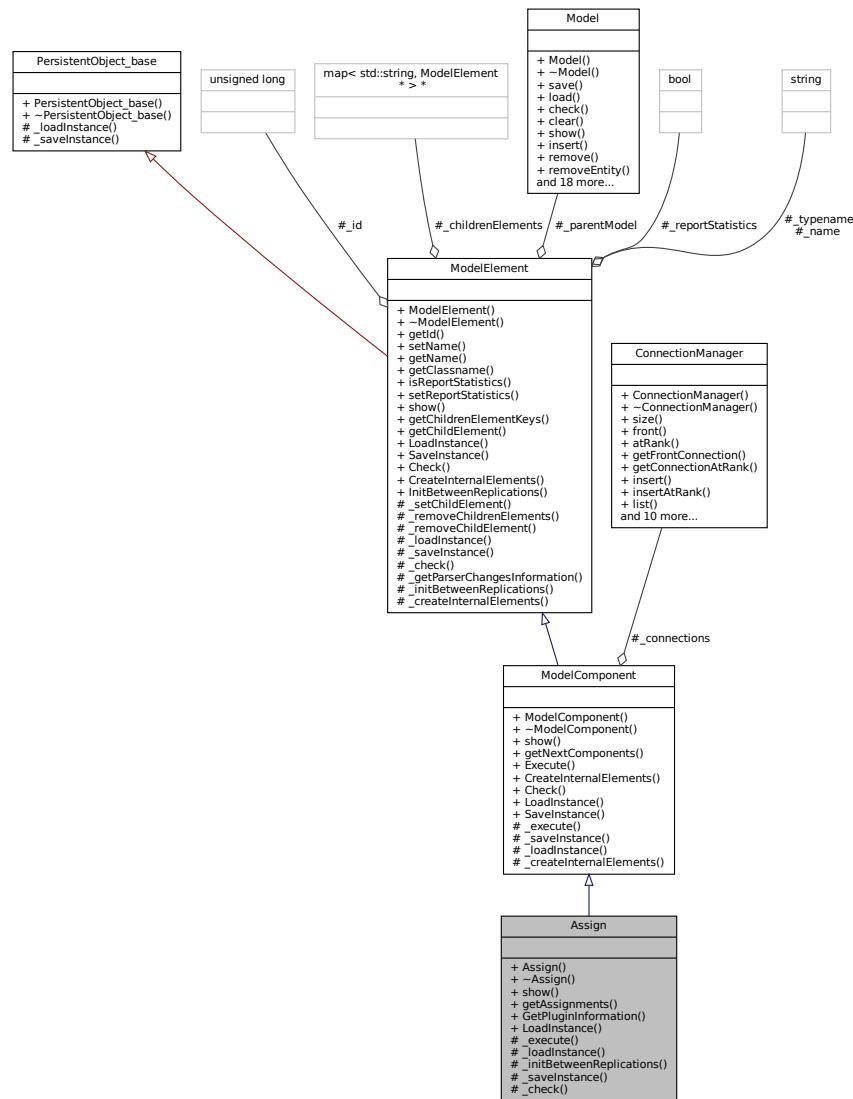
8.3 Assign Class Reference

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

Inheritance diagram for Assign:



Collaboration diagram for Assign:



Classes

- class [Assignment](#)

Public Member Functions

- **Assign (Model *model, std::string name="")**
- virtual **~Assign ()=default**
- virtual std::string **show ()**
- **List< Assignment * > * getAssignments () const**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.3.1 Detailed Description

Assign module DESCRIPTION This module is used for assigning new values to variables, entity attributes, entity types, entity pictures, or other system variables. Multiple assignments can be made with a single **Assign** module. TYPICAL USES Accumulate the number of subassemblies added to a part Change an entity's type to represent the customer copy of a multi-page form Establish a customer's priority PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Assignments Specifies the one or more assignments that will be made when an entity executes the module. Type Type of assignment to be made. Other can include system variables, such as resource capacity or simulation end time. Variable Name Name of the variable that will be assigned a new value when an entity enters the module. Applies only when Type is **Variable**, **Variable Array (1D)**, or **Variable Array (2D)**. Row Specifies the row index for a variable array. Column Specifies the column index for a variable array. Attribute Name Name of the entity attribute that will be assigned a new value when the entity enters the module. Applies only when Type is **Attribute**. Entity Type New entity type that will be assigned to the entity when the entity enters the module. Applies only when Type is **Entity Type**. Entity Picture New entity picture that will be assigned to the entity when the entity enters the module. Applies only when Type is **Entity Picture**. Other Identifies the special system variable that will be assigned a new value when an entity enters the module. Applies only when Type is **Other**. New Value **Assignment** value of the attribute, variable, or other system variable. Does not apply when Type is **Entity Type** or **Entity Picture**.

Definition at line 58 of file [Assign.h](#).

8.3.2 Constructor & Destructor Documentation

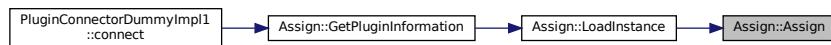
8.3.2.1 Assign() `Assign::Assign (`
 `Model * model,`
 `std::string name = "")`

Definition at line 21 of file [Assign.cpp](#).

```
00021 : ModelComponent (model, Util::TypeOf<Assign>(), name) {  
00022 }
```

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

Here is the caller graph for this function:



8.3.2.2 ~Assign() virtual Assign::~Assign () [virtual], [default]

8.3.3 Member Function Documentation

8.3.3.1 _check() bool Assign::_check (std::string * errorMessage) [protected], [virtual]

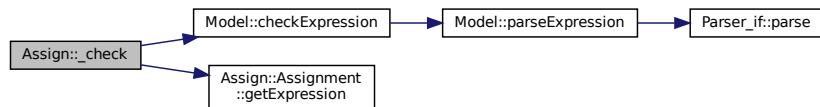
Reimplemented from [ModelElement](#).

Definition at line 96 of file [Assign.cpp](#).

```
00096     Assignment* let;
00097     bool resultAll = true;
00098     // \todo: Reimplement it. Since 201910, attributes may have index, just like "attrib1[2]" or
00099     // "att[10,1]". Because of that, the string may contain not only the name of the attribute, but also its
00100     // index and therefore, fails on the test bellow.
00100     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->end(); it++) {
00101         let = (*it);
00102         resultAll &= \_parentModel->checkExpression(let->getExpression(), "assignment", errorMessage);
00103     }
00104     return resultAll;
00105 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), and [Assign::Assignment::getExpression\(\)](#).

Here is the call graph for this function:



8.3.3.2 _execute() void Assign::_execute (Entity * entity) [protected], [virtual]

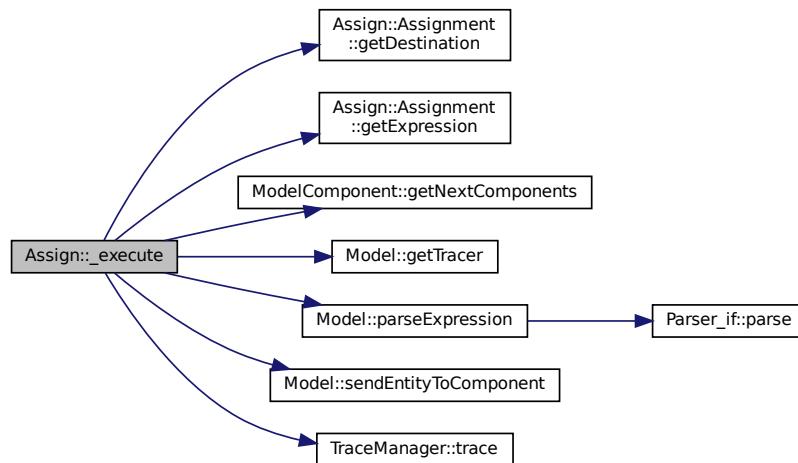
Implements [ModelComponent](#).

Definition at line 50 of file [Assign.cpp](#).

```
00050     Assignment* let;
00051     std::list<Assignment*>* lets = this->_assignments->list();
00052     for (std::list<Assignment*>::iterator it = lets->begin(); it != lets->end(); it++) {
00053         let = (*it);
00054         double value = \_parentModel->parseExpression(let->getExpression());
00055         \_parentModel->parseExpression(let->getDestination() + "=" + std::to_string(value));
00056         \_parentModel->getTracer\(\)->trace("Let \" " + let->getDestination() + " \" = " +
00057             std::to_string(value) + " // " + let->getExpression());
00058     }
00059
00060     this->\_parentModel->sendEntityToComponent(entity, this->getNextComponents\(\)->getFrontConnection(),
00061     0.0);
00061 }
```

References [ModelElement::_parentModel](#), [Assign::Assignment::getDestination\(\)](#), [Assign::Assignment::getExpression\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::parseExpression\(\)](#), [Model::sendEntityToComponent\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.3.3.3 `_initBetweenReplications()`

`void Assign::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 63 of file [Assign.cpp](#).

```
00063
00064 }
```

8.3.3.4 `_loadInstance()`

`bool Assign::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

Definition at line 66 of file [Assign.cpp](#).

```
00066
00067     bool res = ModelComponent::_loadInstance(fields);
00068     if (res) {
00069         unsigned int nv = std::stoi((*(fields->find("assignments"))).second);
00070         for (unsigned int i = 0; i < nv; i++) {
00071             //DestinationType dt = static_cast<DestinationType>
00072             (std::stoi((*(fields->find("destinationType") + std::to_string(i))).second));
00073             std::string dest = ((*(fields->find("destination" + std::to_string(i)))).second);
00074             std::string exp = ((*(fields->find("expression" + std::to_string(i)))).second);
00075             Assignment* assmt = new Assignment(dest, exp);
00076             this->assignments->insert(assmt);
00077         }
00078     }
00079 }
```

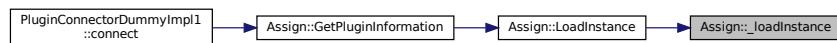
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reimplemented from [ModelComponent](#).

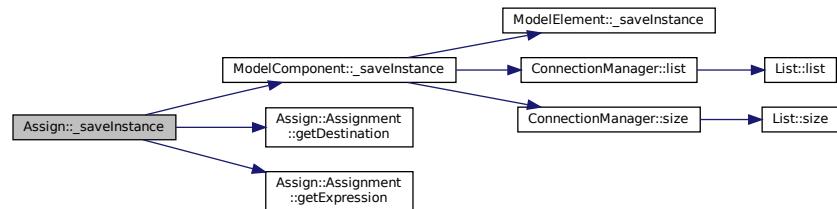
Definition at line 81 of file [Assign.cpp](#).

```

00081 {
00082     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);
00083     //Util::TypeOf<Assign>();
00084     Assignment* let;
00085     fields->emplace("assignments", std::to_string(_assignments->size()));
00086     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->end(); it++) {
00087         let = (*it);
00088         //fields->emplace("destinationType" + std::to_string(i), std::to_string(static_cast<int>(let->getDestinationType())));
00089         fields->emplace("destination" + std::to_string(i), let->getDestination());
00090         fields->emplace("expression" + std::to_string(i), "\"" + let->getExpression() + "\"");
00091     }
00092 }
00093 return fields;
00094 }
```

References [ModelComponent::_saveInstance\(\)](#), [Assign::Assignment::getDestination\(\)](#), and [Assign::Assignment::getExpression\(\)](#).

Here is the call graph for this function:



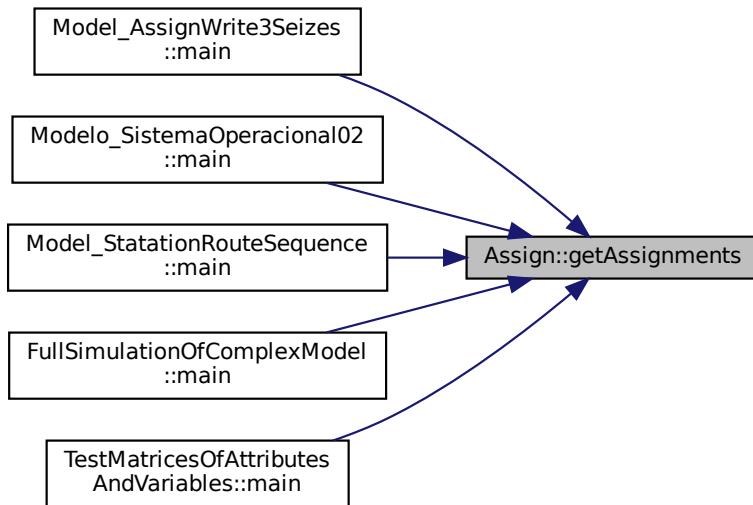
8.3.3.6 getAssignments() `List< Assign::Assignment * > * Assign::getAssignments () const`Definition at line 29 of file `Assign.cpp`.

```

00029
00030     return _assignments;
00031 }
```

Referenced by `Model_AsignWrite3Seizes::main()`, `Modelo_SistemaOperacional02::main()`, `Model_StatationRouteSequence::main()`, `FullSimulationOfComplexModel::main()`, and `TestMatricesOfAttributesAndVariables::main()`.

Here is the caller graph for this function:

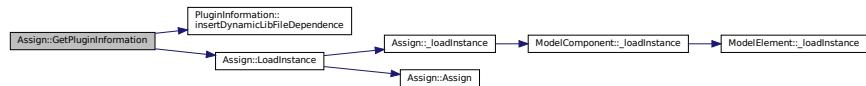
**8.3.3.7 GetPluginInformation()** `PluginInformation * Assign::GetPluginInformation () [static]`Definition at line 33 of file `Assign.cpp`.

```

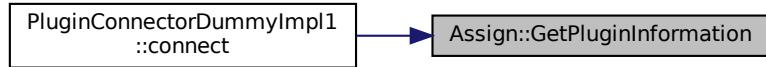
00033
00034     PluginInformation* info = new PluginInformation(Util::TypeOf<Assign>(), &Assign::LoadInstance);
00035     info->insertDynamicLibFileDependence("attribute.so");
00036     info->insertDynamicLibFileDependence("variable.so");
00037     return info;
00038 }
```

References `PluginInformation::insertDynamicLibFileDependence()`, and `LoadInstance()`.Referenced by `PluginConnectorDummyImpl1::connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.3.3.8 LoadInstance() `ModelComponent * Assign::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

Definition at line 40 of file `Assign.cpp`.

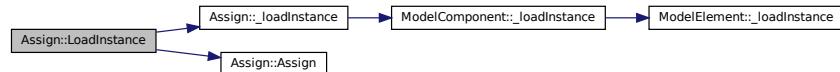
```

00040
00041     Assign* newComponent = new Assign(model);
00042     try {
00043         newComponent->_loadInstance(fields);
00044     } catch (const std::exception& e) {
00045     }
00046
00047     return newComponent;
00048 }
```

References [_loadInstance\(\)](#), and [Assign\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.3.3.9 show() std::string Assign::show () [virtual]

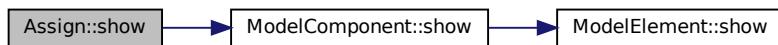
Reimplemented from [ModelComponent](#).

Definition at line 24 of file [Assign.cpp](#).

```
00024     {
00025     return ModelComponent::show() +
00026     "";
00027 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



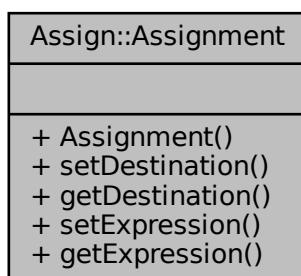
The documentation for this class was generated from the following files:

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

8.4 Assign::Assignment Class Reference

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

Collaboration diagram for Assign::Assignment:



Public Member Functions

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

8.4.1 Detailed Description

While the `Assign` class allows you to perform multiple assignments, the `Assignment` class defines an assignment itself.

Definition at line 64 of file [Assign.h](#).

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Assignment() `Assign::Assignment::Assignment (std::string destination, std::string expression) [inline]`

Definition at line 67 of file [Assign.h](#).

```
00067
00068      this->_destination = destination;
00069      this->_expression = expression;
00070      // an assignment is always in the form:
00071      // (destinationType) destination = expression
00072  };
```

8.4.3 Member Function Documentation

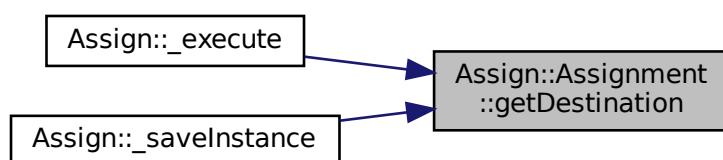
8.4.3.1 getDestination() `std::string Assign::Assignment::getDestination () const [inline]`

Definition at line 78 of file [Assign.h](#).

```
00078
00079      {
00080          return _destination;
00081      }
```

Referenced by [Assign::_execute\(\)](#), and [Assign::_saveInstance\(\)](#).

Here is the caller graph for this function:



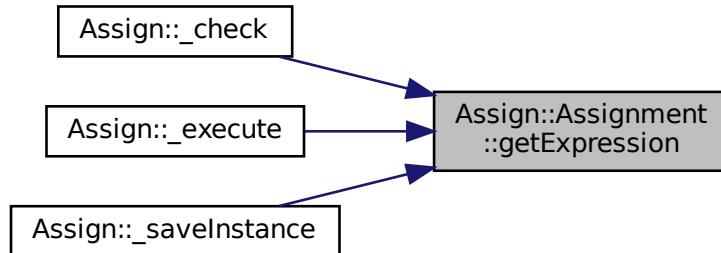
8.4.3.2 getExpression() std::string Assign::Assignment::getExpression () const [inline]

Definition at line 86 of file [Assign.h](#).

```
00086     {
00087         return _expression;
00088     }
```

Referenced by [Assign::_check\(\)](#), [Assign::_execute\(\)](#), and [Assign::_saveInstance\(\)](#).

Here is the caller graph for this function:

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

Definition at line 74 of file [Assign.h](#).

```
00074 {
00075     this->_destination = _destination;
00076 }
```

8.4.3.4 setExpression() void Assign::Assignment::setExpression (std::string _expression) [inline]

Definition at line 82 of file [Assign.h](#).

```
00082 {
00083     this->_expression = _expression;
00084 }
```

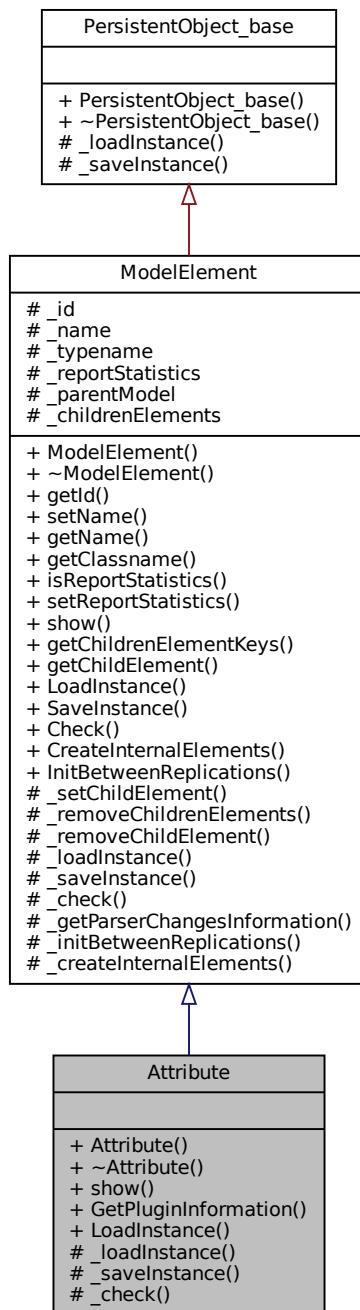
The documentation for this class was generated from the following file:

- [Assign.h](#)

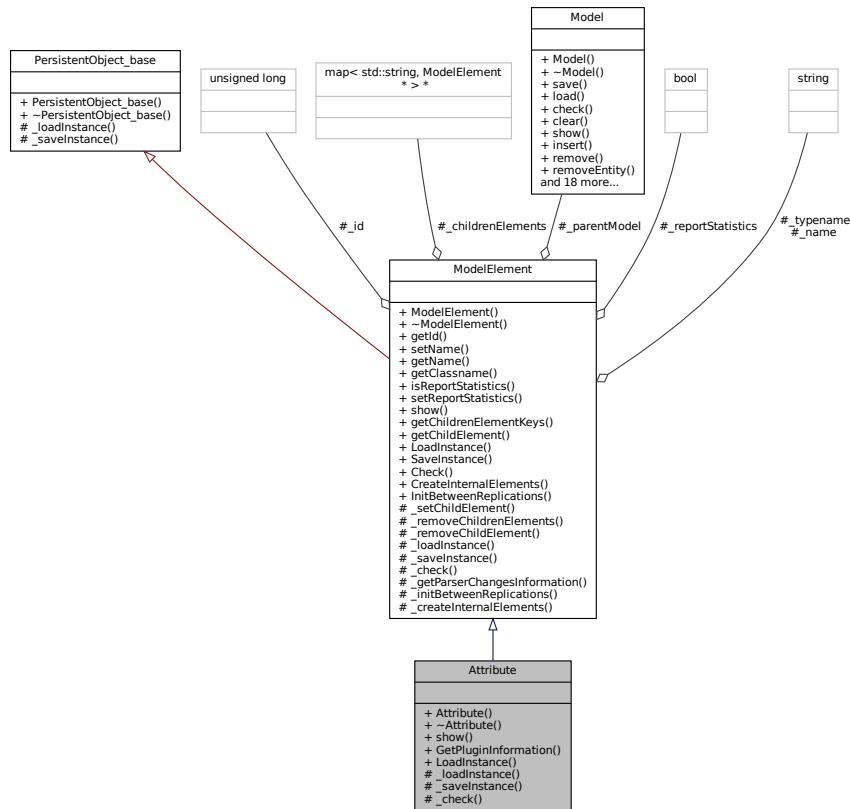
8.5 Attribute Class Reference

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

Inheritance diagram for Attribute:



Collaboration diagram for Attribute:



Public Member Functions

- **Attribute (Model *model, std::string name="")**
- virtual **~Attribute ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.5.1 Detailed Description

Attribute module DESCRIPTION This data module is used to define an attribute's dimension, data type and initial value(s). An attribute is a characteristic of all entities created, but with a specific value that can differ from one entity to another. Attributes can be referenced in other modules (for example, the **Decide** module), can be reassigned a new value with the **Assign** module, and can be used in any expression. **Attribute** values are unique for each entity, as compared to Variables which are global to the simulation module. There are three methods for manually editing the Initial Values of an **Attribute** module: Using the standard spreadsheet interface. In the module spreadsheet, rightclick on the Initial Values cell and select the Edit via spreadsheet menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the module dialog box. In the module spreadsheet, right-click on any cell and select the Edit via dialog menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the two-dimensional (2-D) spreadsheet interface. In the module spreadsheet, click on the Initial Values cell. TYPICAL USES Due date of an order (entity) Priority of an order (entity) Color of a part (entity) PROMPTS Prompt Description Name The unique name of the attribute being defined. Rows Number of rows in a one- or two-dimensional attribute. Columns Number of columns in a two-dimensional attribute. Data Type The data type of the values stored in the attribute. Valid types are Real and String. The default type is Real. Initial Values Lists the initial value or values of the attribute. You can assign new values to the attribute by using the **Assign** module. Initial Value Entity attribute value when entity is created and enters the system.

Definition at line 63 of file [Attribute.h](#).

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Attribute() `Attribute::Attribute (`
 `Model * model,`
 `std::string name = "")`

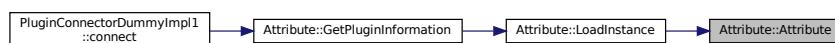
Definition at line 18 of file [Attribute.cpp](#).

```
00018     name) { : ModelElement (model, Util::TypeOf<Attribute>(),  

00019 }
```

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

Here is the caller graph for this function:



8.5.2.2 ~Attribute() `virtual Attribute::~Attribute () [virtual], [default]`

8.5.3 Member Function Documentation

8.5.3.1 `_check()` `bool Attribute::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 50 of file [Attribute.cpp](#).

```
00050
00051     return true;
00052 }
```

8.5.3.2 `_loadInstance()` `bool Attribute::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

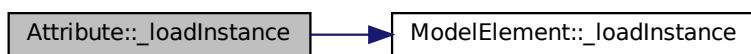
Definition at line 25 of file [Attribute.cpp](#).

```
00025
00026     return ModelElement::_loadInstance(fields);
00027 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

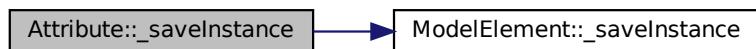
Reimplemented from [ModelElement](#).

Definition at line 45 of file [Attribute.cpp](#).

```
00045     {
00046         std::map<std::string, std::string>* fields = ModelElement::\_saveInstance \(\);
00047         //Util::TypeOf<Attribute>());
00048         return fields;
00049     }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.5.3.4 `GetPluginInformation()` `PluginInformation * Attribute::GetPluginInformation ()` [static]

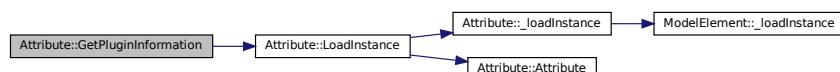
Definition at line 29 of file [Attribute.cpp](#).

```
00029     {
00030         PluginInformation* info = new PluginInformation(Util::TypeOf<Attribute>(),
00031             &Attribute::LoadInstance);
00032         return info;
00033     }
```

References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.5.3.5 LoadInstance() ModelElement * Attribute::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

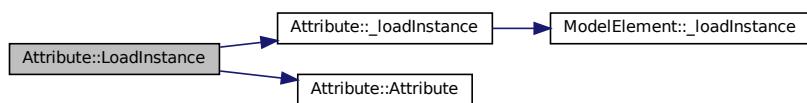
Definition at line 35 of file [Attribute.cpp](#).

```
00035
00036     Attribute* newElement = new Attribute(model);
00037     try {
00038         newElement->_loadInstance(fields);
00039     } catch (const std::exception& e) {
00040
00041     }
00042     return newElement;
00043 }
```

References [_loadInstance\(\)](#), and [Attribute\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reimplemented from [ModelElement](#).

Definition at line 21 of file [Attribute.cpp](#).

```
00021
00022     {
00023     return ModelElement::show();
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



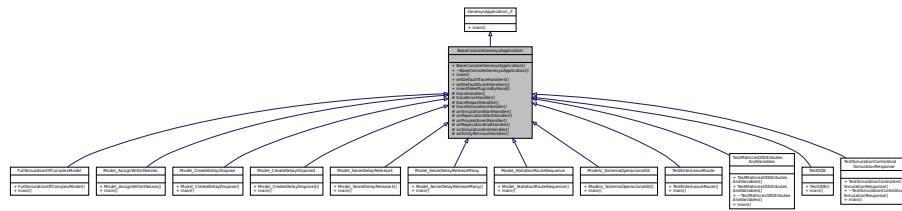
The documentation for this class was generated from the following files:

- Attribute.h
 - Attribute.cpp

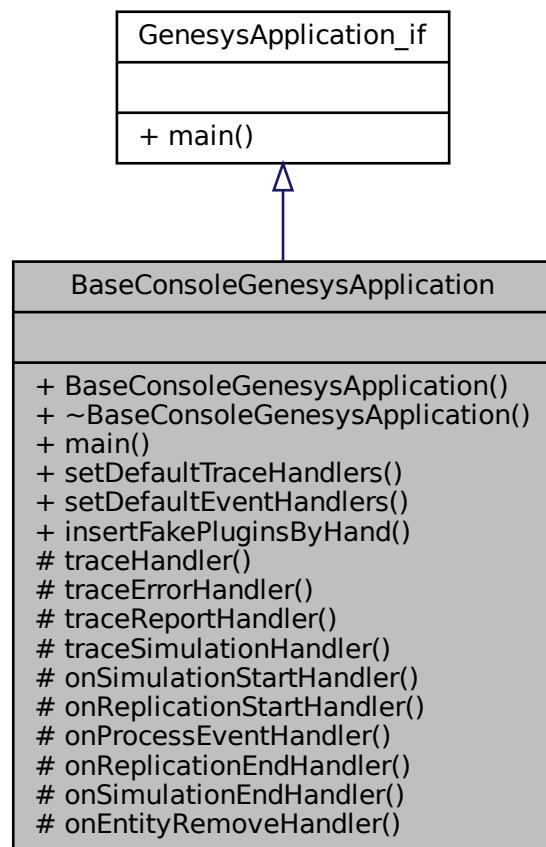
8.6 BaseConsoleGenesysApplication Class Reference

```
#include <BaseConsoleGenesysApplication.h>
```

Inheritance diagram for BaseConsoleGenesysApplication:



Collaboration diagram for BaseConsoleGenesysApplication:



Public Member Functions

- `BaseConsoleGenesysApplication ()`
- `virtual ~BaseConsoleGenesysApplication ()=default`
- `virtual int main (int argc, char **argv)=0`
- `void setDefaultTraceHandlers (TraceManager *tm)`
- `void setDefaultEventHandlers (OnEventManager *oem)`
- `void insertFakePluginsByHand (Simulator *simulator)`

Protected Member Functions

- `virtual void traceHandler (TraceEvent e)`
- `virtual void traceErrorHandler (TraceErrorEvent e)`
- `virtual void traceReportHandler (TraceEvent e)`
- `virtual void traceSimulationHandler (TraceSimulationEvent e)`
- `virtual void onSimulationStartHandler (SimulationEvent *re)`
- `virtual void onReplicationStartHandler (SimulationEvent *re)`
- `virtual void onProcessEventHandler (SimulationEvent *re)`
- `virtual void onReplicationEndHandler (SimulationEvent *re)`
- `virtual void onSimulationEndHandler (SimulationEvent *re)`
- `virtual void onEntityRemoveHandler (SimulationEvent *re)`

8.6.1 Detailed Description

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

8.6.2 Constructor & Destructor Documentation

8.6.2.1 BaseConsoleGenesysApplication() `BaseConsoleGenesysApplication::BaseConsoleGenesysApplication ()`

Definition at line 19 of file `BaseConsoleGenesysApplication.cpp`.
00019
00020 }

8.6.2.2 ~BaseConsoleGenesysApplication() `virtual BaseConsoleGenesysApplication::~BaseConsoleGenesysApplication () [virtual], [default]`

8.6.3 Member Function Documentation

8.6.3.1 insertFakePluginsByHand() void BaseConsoleGenesysApplication::insertFakePluginsByHand (Simulator * simulator)

Definition at line 84 of file [BaseConsoleGenesysApplication.cpp](#).

```

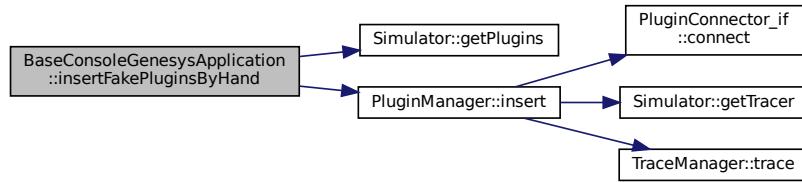
00084
00085 // model components
00086 // arena basic process
00087 simulator->getPlugins()->insert("create.so");
00088 simulator->getPlugins()->insert("dispose.so");
00089 simulator->getPlugins()->insert("decide.so");
00090 simulator->getPlugins()->insert("batch.so");
00091 simulator->getPlugins()->insert("separate.so");
00092 simulator->getPlugins()->insert("assign.so");
00093 simulator->getPlugins()->insert("record.so");
00094 simulator->getPlugins()->insert("submodel.so");
00095 simulator->getPlugins()->insert("entitytype.so");
00096 simulator->getPlugins()->insert("entitygroup.so");
00097 simulator->getPlugins()->insert("attribute.so");
00098 simulator->getPlugins()->insert("counter.so");
00099 simulator->getPlugins()->insert("queue.so");
00100 simulator->getPlugins()->insert("set.so");
00101 simulator->getPlugins()->insert("resource.so");
00102 simulator->getPlugins()->insert("variable.so");
00103 simulator->getPlugins()->insert("schedule.so");
00104 simulator->getPlugins()->insert("entitygroup.so");
00105 // arena advanced process
00106 simulator->getPlugins()->insert("delay.so");
00107 simulator->getPlugins()->insert("dropoff.so");
00108 simulator->getPlugins()->insert("hold.so");
00109 simulator->getPlugins()->insert("match.so");
00110 simulator->getPlugins()->insert("pickup.so");
00111 simulator->getPlugins()->insert("read.so");
00112 simulator->getPlugins()->insert("write.so");
00113 simulator->getPlugins()->insert("release.so");
00114 simulator->getPlugins()->insert("remove.so");
00115 simulator->getPlugins()->insert("seize.so");
00116 simulator->getPlugins()->insert("search.so");
00117 simulator->getPlugins()->insert("signal.so");
00118 simulator->getPlugins()->insert("store.so");
00119 simulator->getPlugins()->insert("unstore.so");
00120 simulator->getPlugins()->insert("expression.so");
00121 simulator->getPlugins()->insert("failure.so");
00122 simulator->getPlugins()->insert("file.so");
00123 simulator->getPlugins()->insert("statisticscollector.so");
00124 simulator->getPlugins()->insert("storage.so");
00125 // arena transfer station
00126 simulator->getPlugins()->insert("enter.so");
00127 simulator->getPlugins()->insert("leave.so");
00128 simulator->getPlugins()->insert("pickstation.so");
00129 simulator->getPlugins()->insert("route.so");
00130 simulator->getPlugins()->insert("sequence.so");
00131 simulator->getPlugins()->insert("station.so");
00132 // arena transfer conveyor
00133 simulator->getPlugins()->insert("access.so");
00134 simulator->getPlugins()->insert("exit.so");
00135 simulator->getPlugins()->insert("start.so");
00136 simulator->getPlugins()->insert("stop.so");
00137 simulator->getPlugins()->insert("conveyour.so");
00138 simulator->getPlugins()->insert("segment.so");
00139 // arena transfer transport
00140 simulator->getPlugins()->insert("alocate.so");
00141 simulator->getPlugins()->insert("free.so");
00142 simulator->getPlugins()->insert("halt.so");
00143 simulator->getPlugins()->insert("move.so");
00144 simulator->getPlugins()->insert("request.so");
00145 simulator->getPlugins()->insert("transporter.so");
00146 simulator->getPlugins()->insert("distance.so");
00147 simulator->getPlugins()->insert("network.so");
00148 simulator->getPlugins()->insert("networklink.so");
00149 // others
00150 simulator->getPlugins()->insert("dummy.so");
00151 simulator->getPlugins()->insert("lode.so");
00152 simulator->getPlugins()->insert("biochemical.so");
00153 simulator->getPlugins()->insert("markovchain.so");
00154 simulator->getPlugins()->insert("cellularautomata.so");
00155 simulator->getPlugins()->insert("cppforgenesys.so");
00156 }
```

References [Simulator::getPlugins\(\)](#), and [PluginManager::insert\(\)](#).

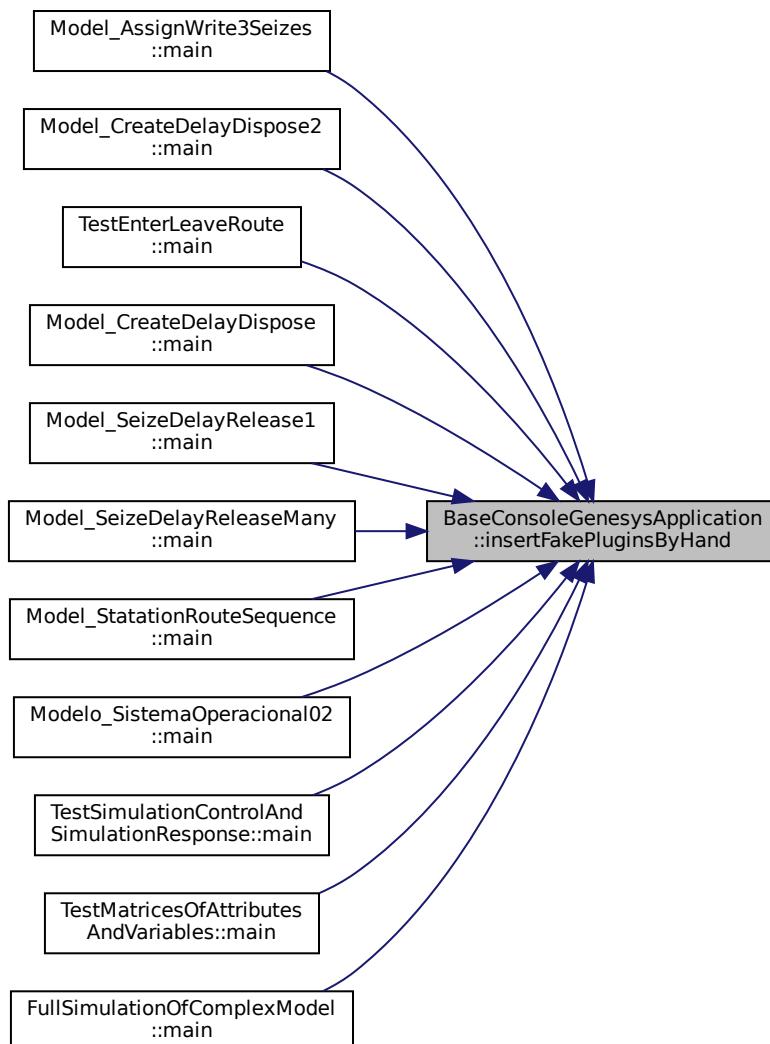
Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#),

[Model_SeizeDelayReleaseMany::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.6.3.2 main() virtual int BaseConsoleGenesysApplication::main (
    int argc,
    char ** argv ) [pure virtual]
```

Implements [GenesysApplication_if](#).

Implemented in [FullSimulationOfComplexModel](#), [TestMatricesOfAttributesAndVariables](#), [TestSimulationControlAndSimulationResponses](#), [Model_AssignWrite3Seizes](#), [Model_CreateDelayDispose](#), [Model_CreateDelayDispose2](#), [Model_SeizeDelayRelease1](#), [Model_SeizeDelayReleaseMany](#), [Model_StationRouteSequence](#), [Modelo_SistemaOperacional02](#), [TestEnterLeaveRoute](#), and [TestODE](#).

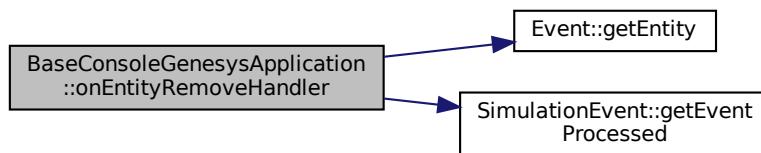
```
8.6.3.3 onEntityRemoveHandler() void BaseConsoleGenesysApplication::onEntityRemoveHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 64 of file [BaseConsoleGenesysApplication.cpp](#).

```
00064
00065     std::cout << "(Event Handler) " << "Entity " << re->getEventProcessed()->getEntity() << " was
00066     removed." << std::endl;
00066 }
```

References [Event::getEntity\(\)](#), and [SimulationEvent::getEventProcessed\(\)](#).

Here is the call graph for this function:



```
8.6.3.4 onProcessEventHandler() void BaseConsoleGenesysApplication::onProcessEventHandler (
    SimulationEvent * re ) [protected], [virtual]
```

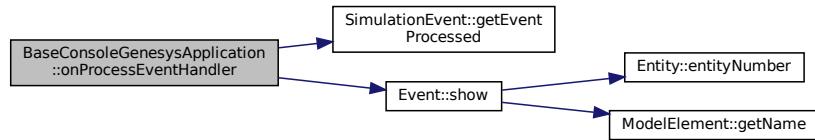
Definition at line 52 of file [BaseConsoleGenesysApplication.cpp](#).

```
00052
00053     std::cout << "(Event Handler) " << "Processing event " << re->getEventProcessed()->show() <<
00054     std::endl;
00054 }
```

References [SimulationEvent::getEventProcessed\(\)](#), and [Event::show\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.5 `onReplicationEndHandler()` `void BaseConsoleGenesysApplication::onReplicationEndHandler (SimulationEvent * re) [protected], [virtual]`

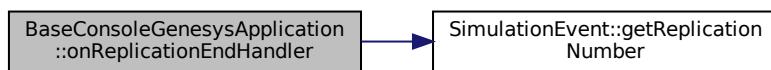
Definition at line 56 of file [BaseConsoleGenesysApplication.cpp](#).

```

00056
00057     std::cout << "(Event Handler) " << "Replication " << re->getReplicationNumber() << " ending." <
00058     std::endl;
  
```

References [SimulationEvent::getReplicationNumber\(\)](#).

Here is the call graph for this function:



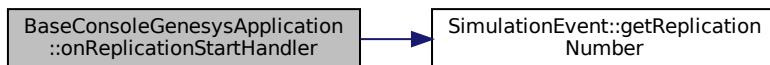
```
8.6.3.6 onReplicationStartHandler() void BaseConsoleGenesysApplication::onReplicationStart<-
Handler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 48 of file [BaseConsoleGenesysApplication.cpp](#).

```
00048
00049     std::cout << "(Event Handler) " << "Replication " << re->getReplicationNumber() << " starting." <<
00050     std::endl;
```

References [SimulationEvent::getReplicationNumber\(\)](#).

Here is the call graph for this function:



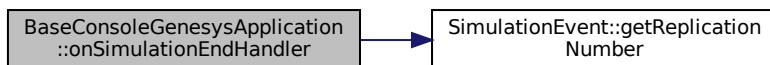
```
8.6.3.7 onSimulationEndHandler() void BaseConsoleGenesysApplication::onSimulationEndHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 60 of file [BaseConsoleGenesysApplication.cpp](#).

```
00060
00061     std::cout << "Event (Handler) " << "Replication " << re->getReplicationNumber() << " ending." <<
00062     std::endl;
```

References [SimulationEvent::getReplicationNumber\(\)](#).

Here is the call graph for this function:



```
8.6.3.8 onSimulationStartHandler() void BaseConsoleGenesysApplication::onSimulationStartHandler
(
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 44 of file [BaseConsoleGenesysApplication.cpp](#).

```
00044
00045     std::cout << "(Event Handler) " << "Simulation is starting" << std::endl;
00046 }
```

8.6.3.9 setDefaultEventHandlers() void BaseConsoleGenesysApplication::setDefaultEventHandlers (OnEventManager * oem)

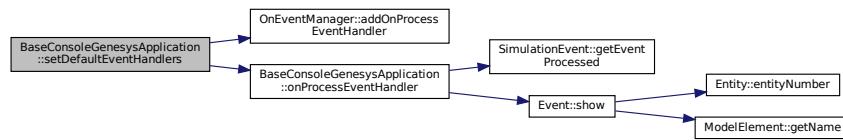
Definition at line 68 of file [BaseConsoleGenesysApplication.cpp](#).

```
00068
00069     //oem->addOnSimulationStartHandler(this,
00070     //    &BaseConsoleGenesysApplication::onSimulationStartHandler);
00071     //oem->addOnReplicationStartHandler(this,
00072     //    &BaseConsoleGenesysApplication::onReplicationStartHandler);
00073     oem->addOnProcessEventHandler(this, &BaseConsoleGenesysApplication::onProcessEventHandler);
00074     //oem->addOnReplicationEndHandler(this, &BaseConsoleGenesysApplication::onReplicationEndHandler);
00075     //oem->addOnSimulationEndHandler(this, &BaseConsoleGenesysApplication::onSimulationEndHandler);
00076     //oem->addOnEntityRemoveHandler(this, &BaseConsoleGenesysApplication::onEntityRemoveHandler);
00077 }
```

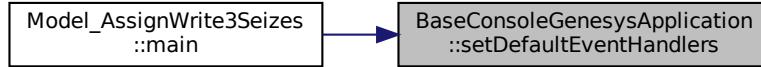
References [OnEventManager::addOnProcessEventHandler\(\)](#), and [onProcessEventHandler\(\)](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.10 setDefaultTraceHandlers() void BaseConsoleGenesysApplication::setDefaultTraceHandlers (TraceManager * tm)

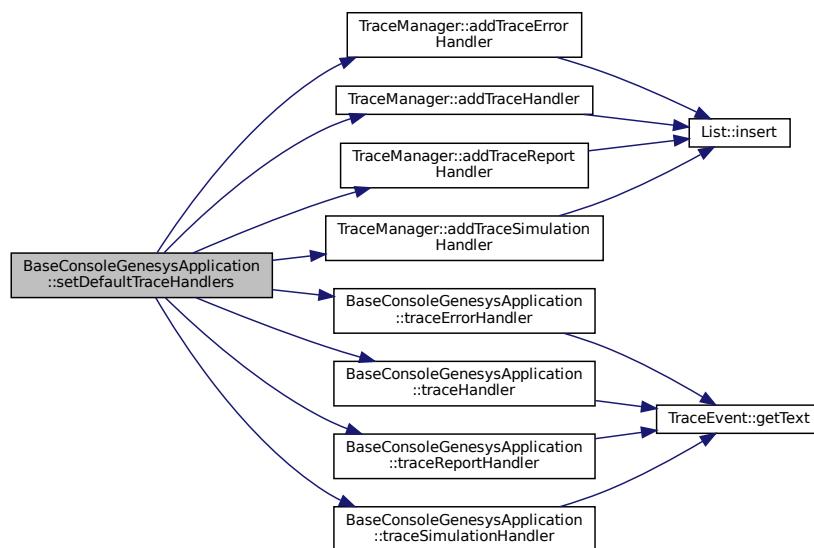
Definition at line 77 of file [BaseConsoleGenesysApplication.cpp](#).

```
00077
00078     tm->addTraceHandler<BaseConsoleGenesysApplication>(this,
00079     &BaseConsoleGenesysApplication::traceHandler);
00080     tm->addTraceErrorHandler<BaseConsoleGenesysApplication>(this,
00081     &BaseConsoleGenesysApplication::traceErrorHandler);
00082     tm->addTraceReportHandler<BaseConsoleGenesysApplication>(this,
00083     &BaseConsoleGenesysApplication::traceReportHandler);
00084     tm->addTraceSimulationHandler<BaseConsoleGenesysApplication>(this,
00085     &BaseConsoleGenesysApplication::traceSimulationHandler);
00086 }
```

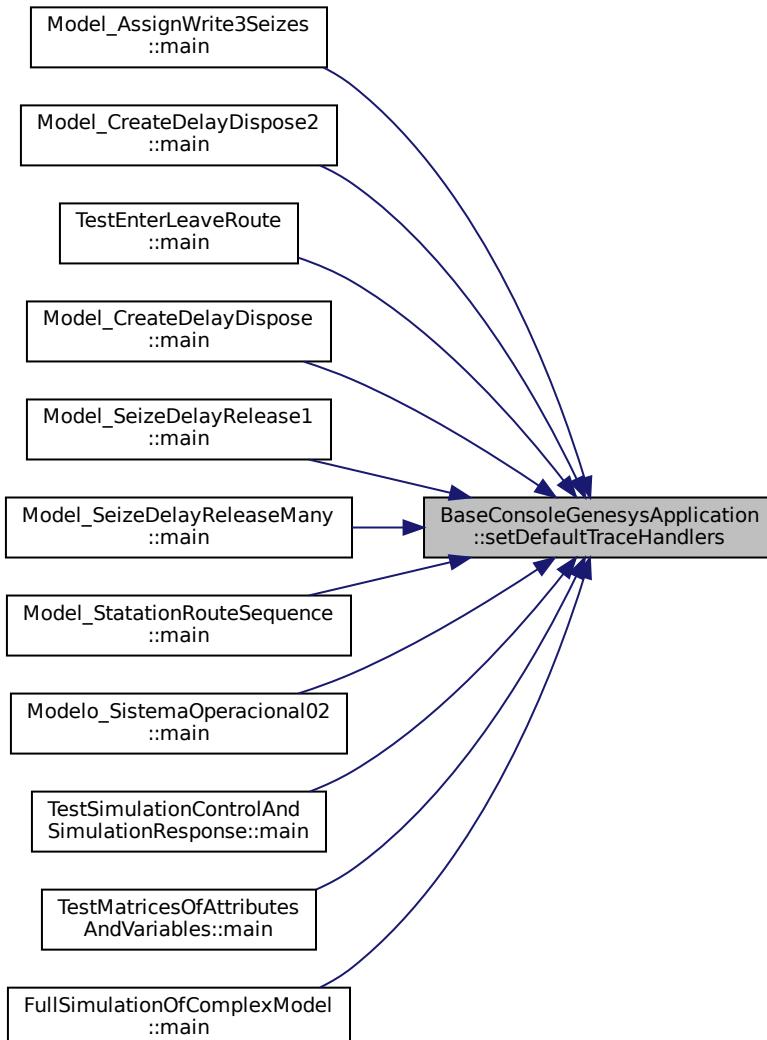
References [TraceManager::addTraceErrorHandler\(\)](#), [TraceManager::addTraceHandler\(\)](#), [TraceManager::addTraceReportHandler\(\)](#), [TraceManager::addTraceSimulationHandler\(\)](#), [traceErrorHandler\(\)](#), [traceHandler\(\)](#), [traceReportHandler\(\)](#), and [traceSimulationHandler\(\)](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.11 traceErrorHandler() void BaseConsoleGenesysApplication::traceErrorHandler (
 TraceErrorEvent e) [protected], [virtual]

Definition at line 29 of file [BaseConsoleGenesysApplication.cpp](#).

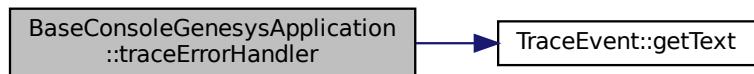
```

00029
00030     std::cout << e.getText() << std::endl;
00031 }
```

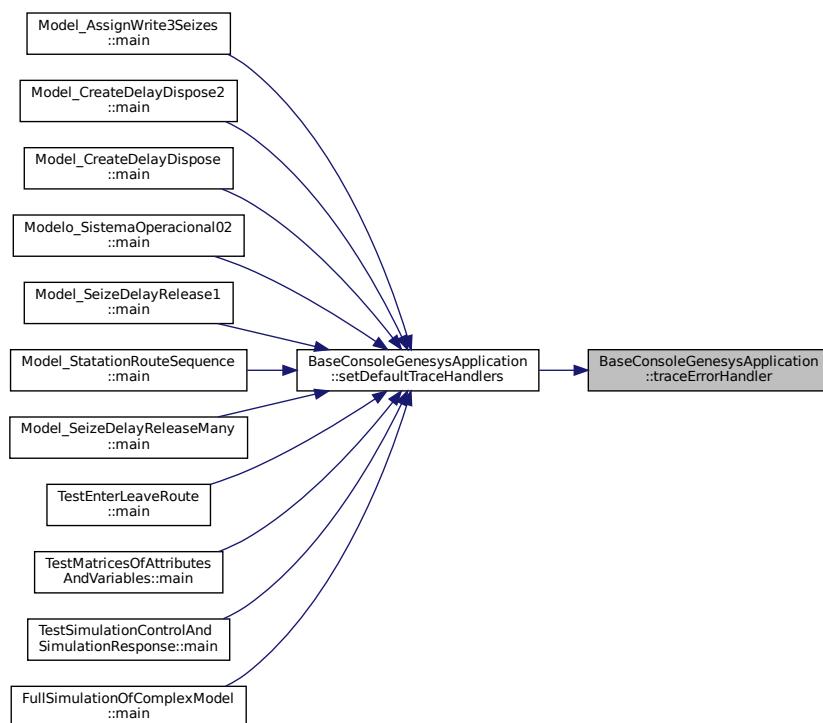
References [TraceEvent::getText\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.12 traceHandler() void BaseConsoleGenesysApplication::traceHandler ([TraceEvent e](#)) [protected], [virtual]

Definition at line 25 of file [BaseConsoleGenesysApplication.cpp](#).

```

00025     std::cout << e.getText() << std::endl;
00026 }
00027 }
```

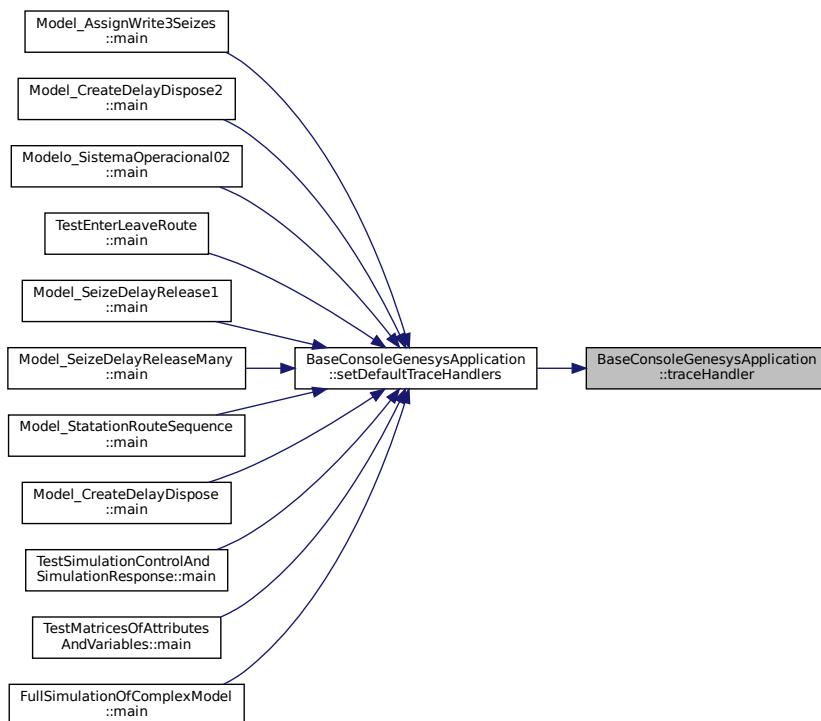
References [TraceEvent::getText\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.13 traceReportHandler() void BaseConsoleGenesysApplication::traceReportHandler ([TraceEvent e](#)) [protected], [virtual]

Definition at line 33 of file [BaseConsoleGenesysApplication.cpp](#).

```

00033
00034     std::cout << "" << e.getText() << "" << std::endl;
00035 }
```

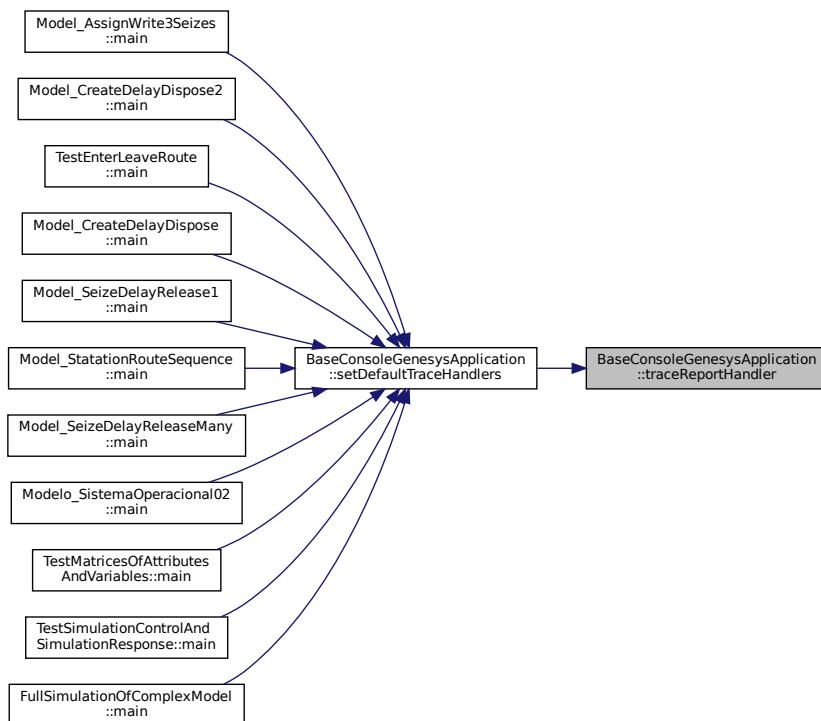
References [TraceEvent::getText\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.3.14 traceSimulationHandler() void BaseConsoleGenesysApplication::traceSimulationHandler ([TraceSimulationEvent e](#)) [protected], [virtual]

Definition at line 37 of file [BaseConsoleGenesysApplication.cpp](#).

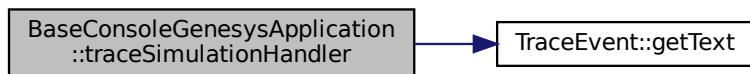
```

00037
00038     std::cout << e.getText() << std::endl;
00039 }
  
```

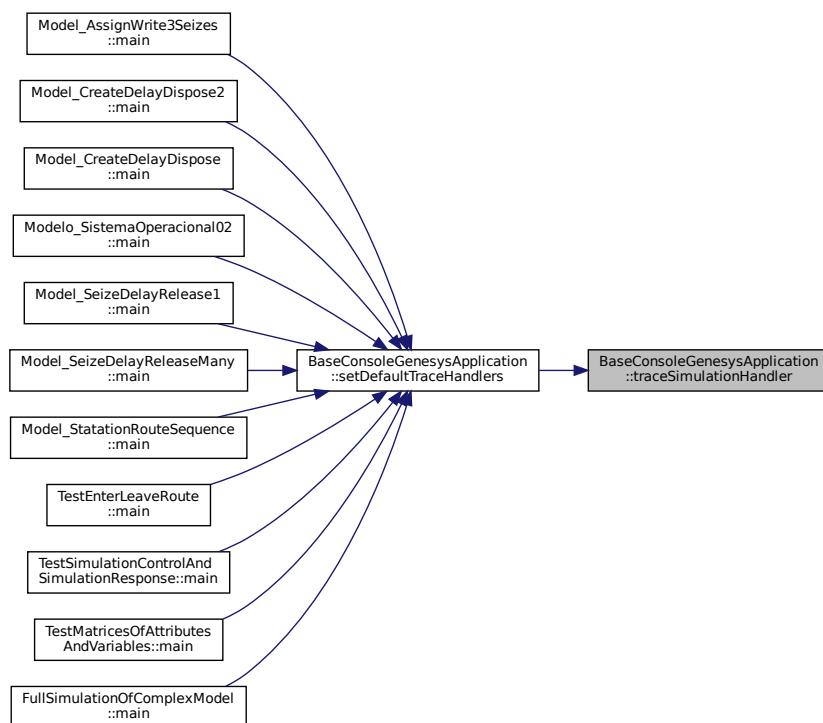
References [TraceEvent::getText\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



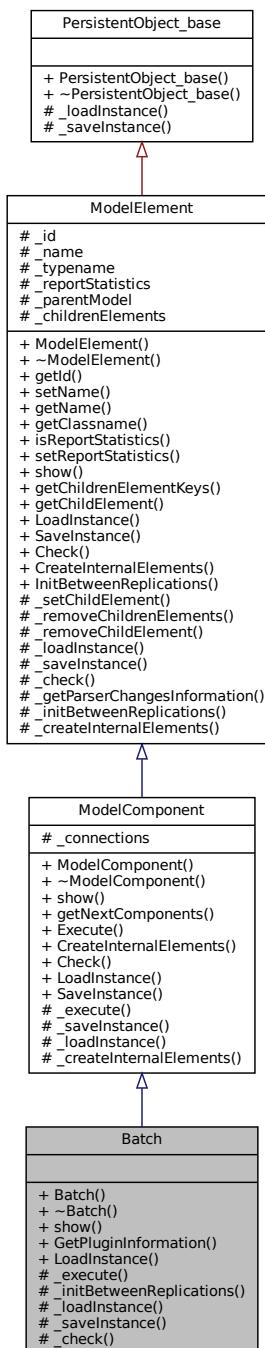
The documentation for this class was generated from the following files:

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

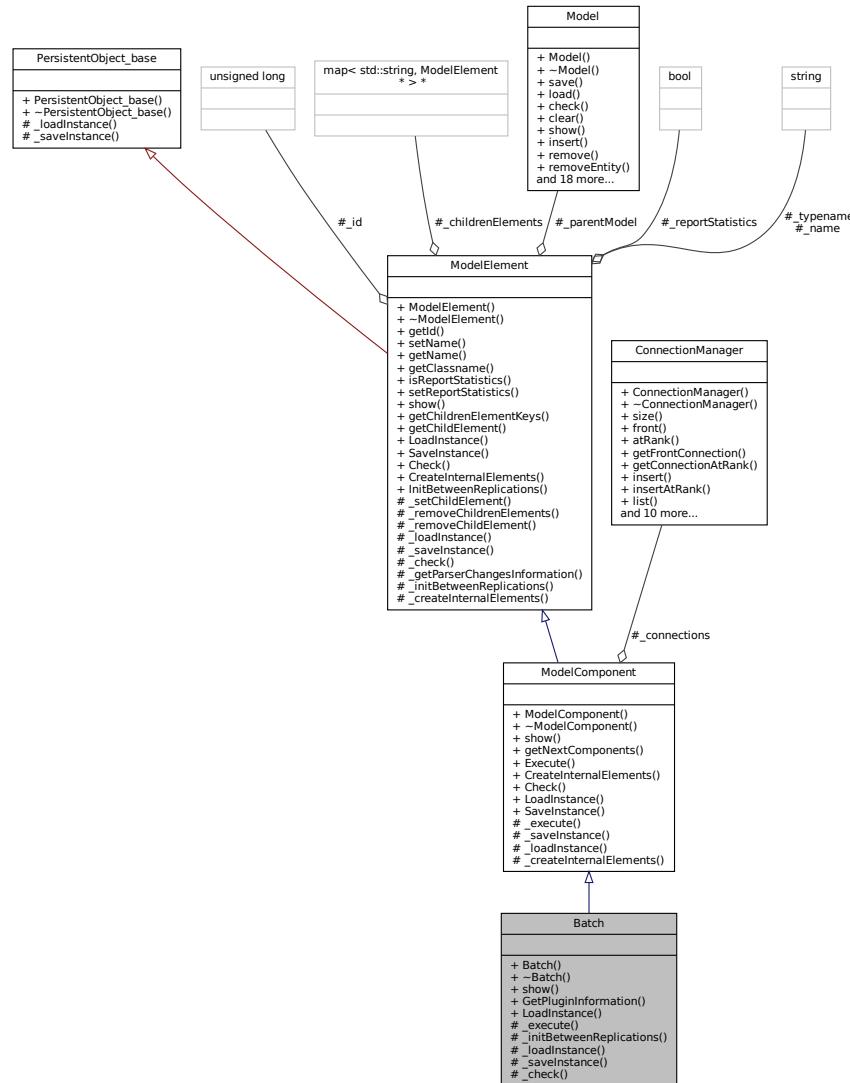
8.7 Batch Class Reference

```
#include <Batch.h>
```

Inheritance diagram for Batch:



Collaboration diagram for Batch:



Public Member Functions

- `Batch (Model *model, std::string name="")`
- virtual `~Batch ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.7.1 Detailed Description

Batch module DESCRIPTION This module is intended as the grouping mechanism within the simulation model. Batches can be permanently or temporarily grouped. Temporary batches must later be split using the Separate module. Batches may be made with any specified number of entering entities or may be matched together based on an attribute. Entities arriving at the **Batch** module are placed in a queue until the required number of entities has accumulated. Once accumulated, a new representative entity is created. TYPICAL USES Collect a number of parts before starting processing Reassemble previously separated copies of a form Bring together a patient and his record before commencing an appointment PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of batching entities together. **Batch** Size Number of entities to be batched. Save Criterion Method for assigning representative entity's user-defined attribute values. Rule Determines how incoming entities will be batched. Any **Entity** will take the first "Batch Size" number of entities and put them together. By **Attribute** signifies that the values of the specified attribute must match for entities to be grouped. For example, if **Attribute** Name is Color, all entities must have the same Color value to be grouped; otherwise, they will wait at the module for additional incoming entities. **Attribute** Name Name of the attribute whose value must match the value of the other incoming entities in order for a group to be made. Applies only when Rule is By **Attribute**. Representative **Entity** The entity type for the representative entity.

Definition at line 53 of file [Batch.h](#).

8.7.2 Constructor & Destructor Documentation

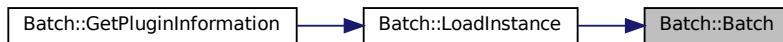
8.7.2.1 Batch() `Batch::Batch (`
 `Model * model,`
 `std::string name = "")`

Definition at line 18 of file [Batch.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Batch>(), name) {  
00019 }
```

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

Here is the caller graph for this function:



8.7.2.2 ~Batch() `virtual Batch::~Batch () [virtual], [default]`

8.7.3 Member Function Documentation

```
8.7.3.1 _check() bool Batch::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Batch.cpp](#).

```
00057
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
```

```
8.7.3.2 _execute() void Batch::_execute (
    Entity * entity ) [protected], [virtual]
```

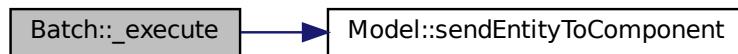
Implements [ModelComponent](#).

Definition at line 35 of file [Batch.cpp](#).

```
00035
00036     {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00038         this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
    }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



```
8.7.3.3 _initBetweenReplications() void Batch::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Batch.cpp](#).

```
00048
00049 }
```

8.7.3.4 `_loadInstance()` `bool Batch::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

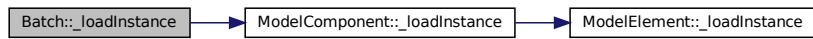
Definition at line 40 of file [Batch.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

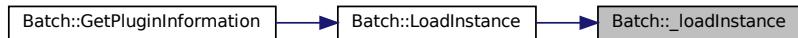
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.5 `_saveInstance()` `std::map< std::string, std::string > * Batch::_saveInstance () [protected], [virtual]`

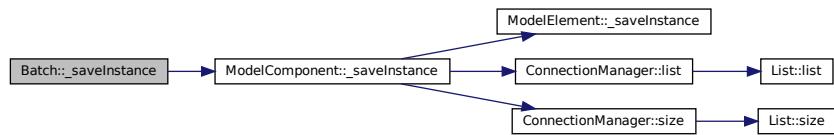
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Batch.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



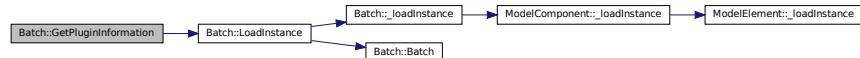
8.7.3.6 GetPluginInformation() `PluginInformation * Batch::GetPluginInformation () [static]`

Definition at line 63 of file [Batch.cpp](#).

```
00063                                         {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Batch>(), &Batch::LoadInstance);
00065     // ...
00066     return info;
00067 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.7.3.7 LoadInstance() `ModelComponent * Batch::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

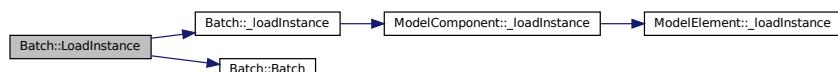
Definition at line 25 of file [Batch.cpp](#).

```
00025
00026     Batch* newComponent = new Batch(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
```

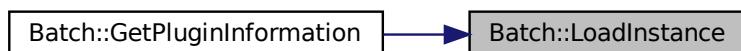
References [_loadInstance\(\)](#), and [Batch\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.8 `show()` `std::string Batch::show() [virtual]`

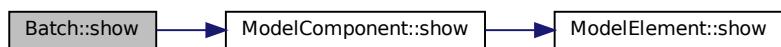
Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Batch.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



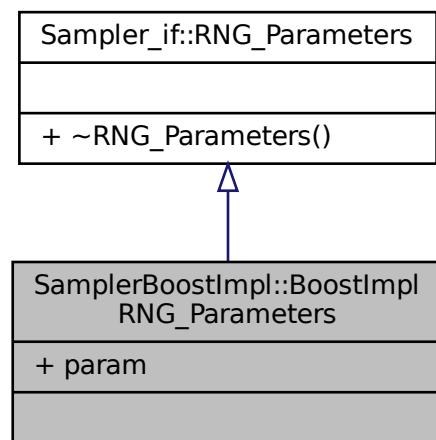
The documentation for this class was generated from the following files:

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

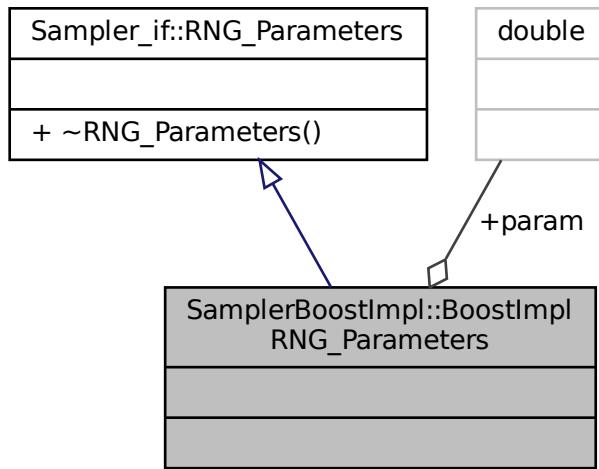
8.8 SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference

```
#include <SamplerBoostImpl.h>
```

Inheritance diagram for SamplerBoostImpl::BoostImplRNG_Parameters:



Collaboration diagram for SamplerBoostImpl::BoostImplRNG_Parameters:



Public Attributes

- `double param`

Additional Inherited Members

8.8.1 Detailed Description

Definition at line 23 of file [SamplerBoostImpl.h](#).

8.8.2 Member Data Documentation

8.8.2.1 `param` `double SamplerBoostImpl::BoostImplRNG_Parameters::param`

Definition at line 24 of file [SamplerBoostImpl.h](#).

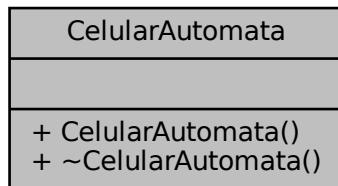
The documentation for this struct was generated from the following file:

- [SamplerBoostImpl.h](#)

8.9 CelularAutomata Class Reference

```
#include <CellularAutomata.h>
```

Collaboration diagram for CelularAutomata:



Public Member Functions

- [CelularAutomata \(\)](#)
- virtual [~CelularAutomata \(\)](#)=default

8.9.1 Detailed Description

Definition at line 17 of file [CellularAutomata.h](#).

8.9.2 Constructor & Destructor Documentation

8.9.2.1 CelularAutomata() [CellularAutomata::CelularAutomata \(\)](#)

Definition at line 16 of file [CellularAutomata.cpp](#).

```
00016 {  
00017 }
```

8.9.2.2 ~CelularAutomata() [virtual CellularAutomata::~CelularAutomata \(\) \[virtual\], \[default\]](#)

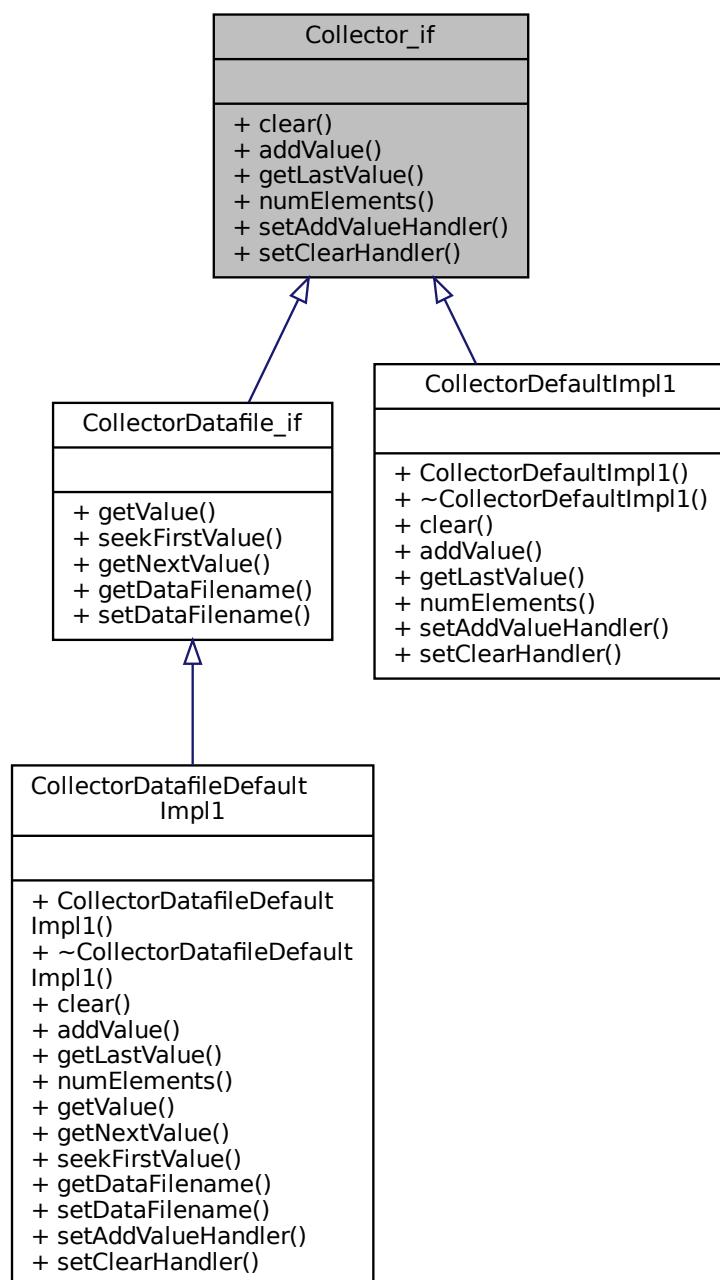
The documentation for this class was generated from the following files:

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

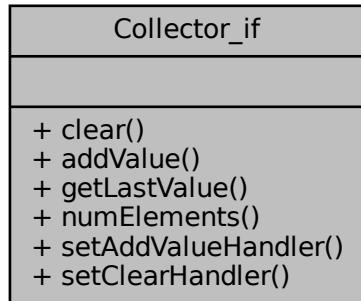
8.10 Collector_if Class Reference

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

Inheritance diagram for Collector_if:



Collaboration 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`

8.10.1 Detailed Description

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

Definition at line 39 of file [Collector_if.h](#).

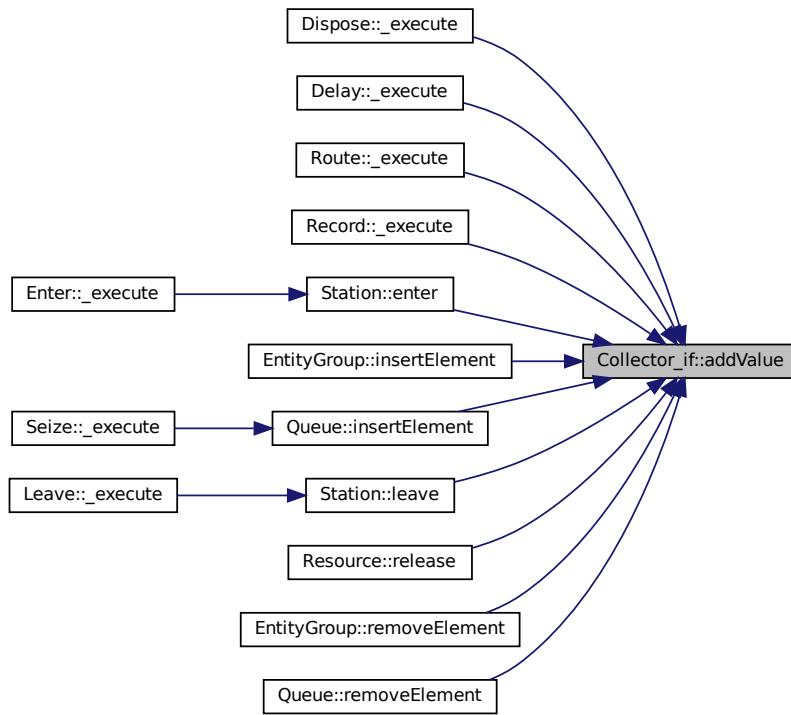
8.10.2 Member Function Documentation

8.10.2.1 `addValue()` `virtual void Collector_if::addValue (`
 `double value) [pure virtual]`

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

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Route::_execute\(\)](#), [Record::_execute\(\)](#), [Station::enter\(\)](#), [EntityGroup::insertElement\(\)](#), [Queue::insertElement\(\)](#), [Station::leave\(\)](#), [Resource::release\(\)](#), [EntityGroup::removeElement\(\)](#), and [Queue::removeElement\(\)](#).

Here is the caller graph for this function:

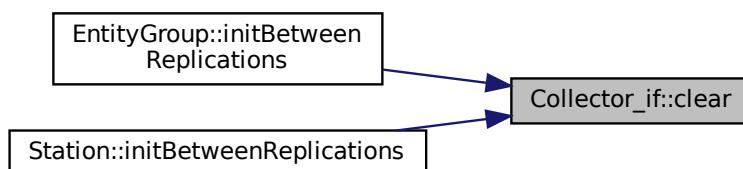


8.10.2.2 `clear()` virtual void Collector_if::clear () [pure virtual]

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

Referenced by [EntityGroup::initBetweenReplications\(\)](#), and [Station::initBetweenReplications\(\)](#).

Here is the caller graph for this function:



8.10.2.3 getLastValue() virtual double Collector_if::getLastValue () [pure virtual]

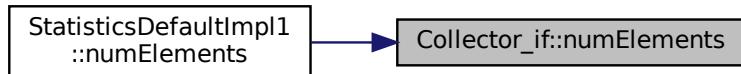
Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

8.10.2.4 numElements() virtual unsigned long Collector_if::numElements () [pure virtual]

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

Referenced by [StatisticsDefaultImpl1::numElements\(\)](#).

Here is the caller graph for this function:



8.10.2.5 setAddValueHandler() virtual void Collector_if::setAddValueHandler ([CollectorAddValueHandler](#) addValueHandler) [pure virtual]

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

Referenced by [StatisticsDefaultImpl1::StatisticsDefaultImpl1\(\)](#).

Here is the caller graph for this function:

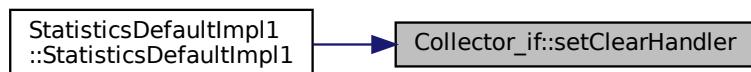


```
8.10.2.6 setClearHandler() virtual void Collector_if::setClearHandler (
    CollectorClearHandler clearHandler ) [pure virtual]
```

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

Referenced by [StatisticsDefaultImpl1::StatisticsDefaultImpl1\(\)](#).

Here is the caller graph for this function:



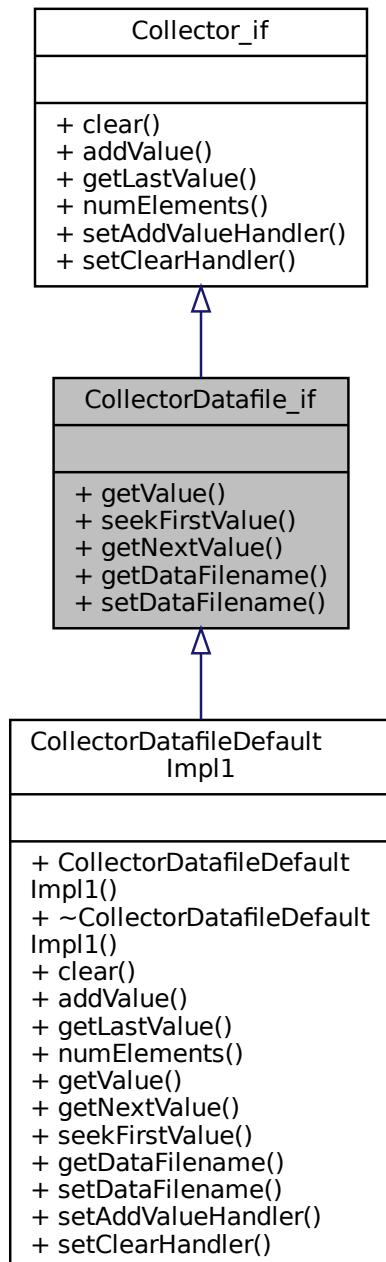
The documentation for this class was generated from the following file:

- [Collector_if.h](#)

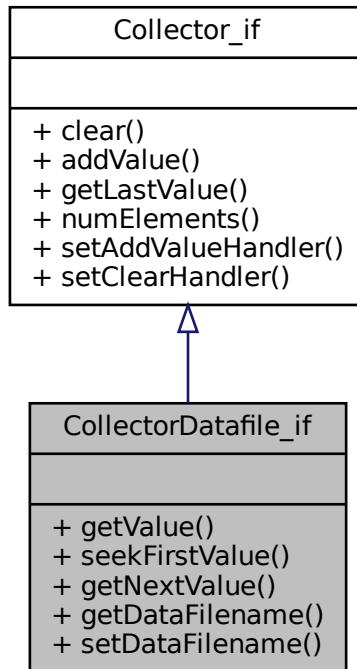
8.11 CollectorDatafile_if Class Reference

```
#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

8.11.1 Detailed Description

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

Definition at line 22 of file [CollectorDatafile_if.h](#).

8.11.2 Member Function Documentation

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

Get the next value in the file and advances the pointer

Implemented in [CollectorDatafileDefaultImpl1](#).

8.11.2.2 `getNextValue()` `virtual double CollectorDatafile_if::getNextValue () [pure virtual]`

Set the pointer to the first value in the file

Implemented in [CollectorDatafileDefaultImpl1](#).

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

Implemented in [CollectorDatafileDefaultImpl1](#).

8.11.2.4 `seekFirstValue()` `virtual void CollectorDatafile_if::seekFirstValue () [pure virtual]`

Get a value from a specific position

Implemented in [CollectorDatafileDefaultImpl1](#).

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

Implemented in [CollectorDatafileDefaultImpl1](#).

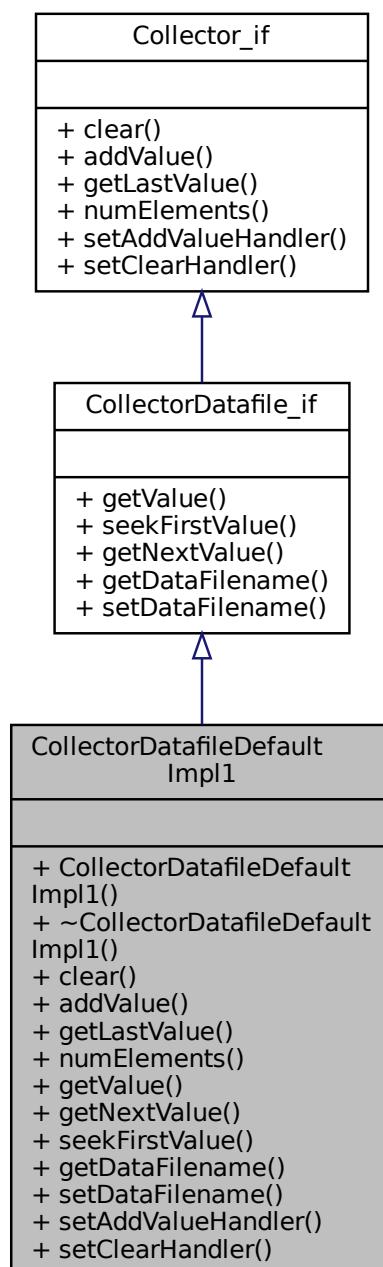
The documentation for this class was generated from the following file:

- [CollectorDatafile_if.h](#)

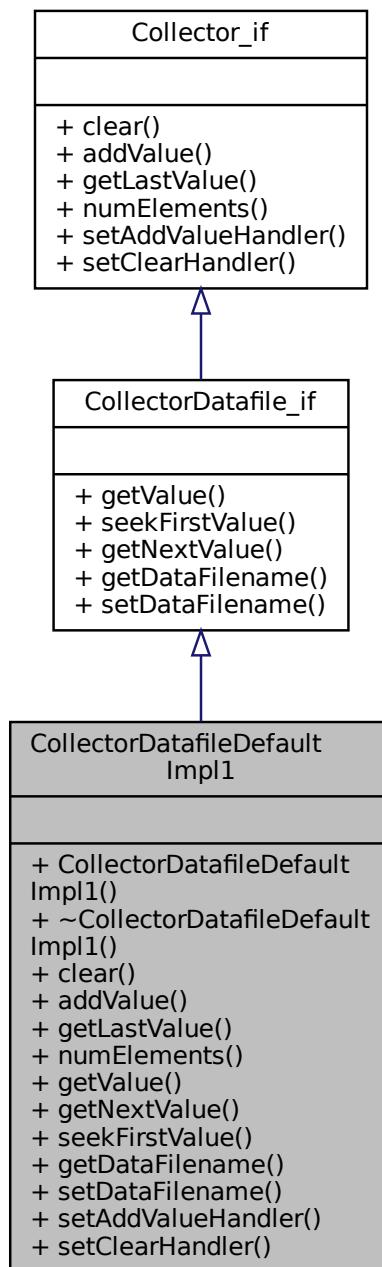
8.12 CollectorDatafileDefaultImpl1 Class Reference

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

Inheritance diagram for CollectorDatafileDefaultImpl1:



Collaboration diagram for CollectorDatafileDefaultImpl1:



Public Member Functions

- `CollectorDatafileDefaultImpl1 ()`
- virtual `~CollectorDatafileDefaultImpl1 ()=default`
- `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)`

8.12.1 Detailed Description

Definition at line 21 of file [CollectorDatafileDefaultImpl1.h](#).

8.12.2 Constructor & Destructor Documentation

8.12.2.1 CollectorDatafileDefaultImpl1() `CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ()`

Definition at line 16 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00016 {  
00017 }
```

8.12.2.2 ~CollectorDatafileDefaultImpl1() `virtual CollectorDatafileDefaultImpl1::~CollectorDatafileDefaultImpl1 () [virtual], [default]`

8.12.3 Member Function Documentation

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

Implements [Collector_if](#).

Definition at line 22 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00022 {  
00023 }
```

8.12.3.2.clear() `void CollectorDatafileDefaultImpl1::clear () [virtual]`

Implements [Collector_if](#).

Definition at line 19 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00019 {  
00020 }
```

8.12.3.3 getDataFilename() std::string CollectorDatafileDefaultImpl1::getDataFilename () [virtual]

Get the next value in the file and advances the pointer

Implements [CollectorDatafile_if](#).

Definition at line 44 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00044
00045     return _filename;
00046 }
```

8.12.3.4 getLastValue() double CollectorDatafileDefaultImpl1::getLastValue () [virtual]

Implements [Collector_if](#).

Definition at line 25 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00025
00026     return 0.0; // \todo:
00027 }
```

8.12.3.5 getNextValue() double CollectorDatafileDefaultImpl1::getNextValue () [virtual]

Set the pointer to the first value in the file

Implements [CollectorDatafile_if](#).

Definition at line 37 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00037
00038     return 0.0; // \todo:
00039 }
```

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

Implements [CollectorDatafile_if](#).

Definition at line 33 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00033
00034     return 0.0; // \todo:
00035 }
```

8.12.3.7 numElements() unsigned long CollectorDatafileDefaultImpl1::numElements () [virtual]

Implements [Collector_if](#).

Definition at line 29 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00029
00030     return 0.0; // \todo:
00031 }
```

8.12.3.8 seekFirstValue() void CollectorDatafileDefaultImpl1::seekFirstValue () [virtual]

Get a value from a specific position

Implements [CollectorDatafile_if](#).

Definition at line 41 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00041
00042 }
```

8.12.3.9 setAddValueHandler() void CollectorDatafileDefaultImpl1::setAddValueHandler ([CollectorAddValueHandler](#) addValueHandler) [virtual]

Implements [Collector_if](#).

Definition at line 52 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00052
00053
00054 }
```

8.12.3.10 setClearHandler() void CollectorDatafileDefaultImpl1::setClearHandler ([CollectorClearHandler](#) clearHandler) [virtual]

Implements [Collector_if](#).

Definition at line 56 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00056
00057
00058 }
```

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

Implements [CollectorDatafile_if](#).

Definition at line 48 of file [CollectorDatafileDefaultImpl1.cpp](#).

```
00048
00049     _filename = filename;
00050 }
```

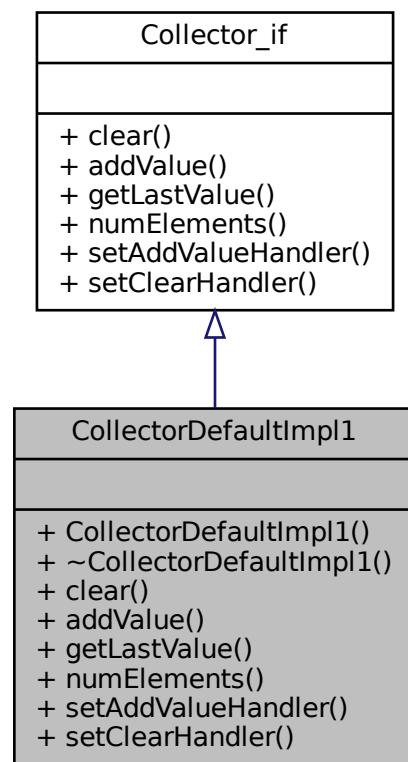
The documentation for this class was generated from the following files:

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

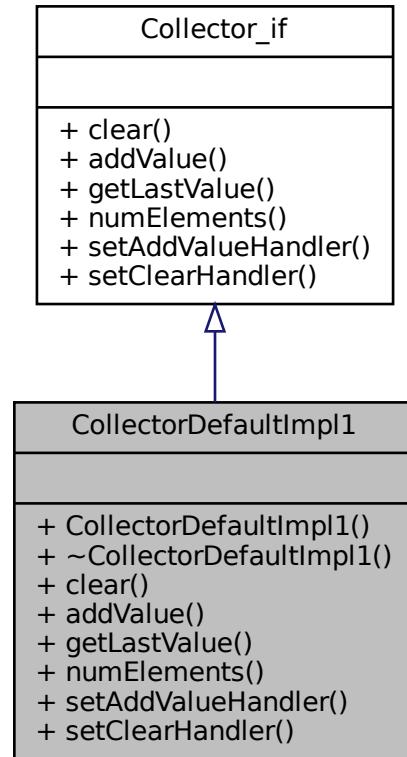
8.13 CollectorDefaultImpl1 Class Reference

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

Inheritance diagram for CollectorDefaultImpl1:



Collaboration diagram for CollectorDefaultImpl1:



Public Member Functions

- `CollectorDefaultImpl1 ()`
- virtual `~CollectorDefaultImpl1 ()=default`
- `void clear ()`
- `void addValue (double value)`
- `double getLastValue ()`
- `unsigned long numElements ()`
- `void setAddValueHandler (CollectorAddValueHandler addValueHandler)`
- `void setClearHandler (CollectorClearHandler clearHandler)`

8.13.1 Detailed Description

Definition at line 20 of file [CollectorDefaultImpl1.h](#).

8.13.2 Constructor & Destructor Documentation

8.13.2.1 CollectorDefaultImpl1() `CollectorDefaultImpl1::CollectorDefaultImpl1 ()`

Definition at line 18 of file [CollectorDefaultImpl1.cpp](#).

```
00018 {  
00019 }
```

8.13.2.2 ~CollectorDefaultImpl1() `virtual CollectorDefaultImpl1::~CollectorDefaultImpl1 () [virtual], [default]`**8.13.3 Member Function Documentation****8.13.3.1 addValue()** `void CollectorDefaultImpl1::addValue (double value) [virtual]`

Implements [Collector_if](#).

Definition at line 28 of file [CollectorDefaultImpl1.cpp](#).

```
00028 {  
00029     _lastValue = value;  
00030     _numElements++;  
00031     if (_addValueHandler != nullptr) {  
00032         _addValueHandler(value);  
00033     }  
00034 }
```

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

Implements [Collector_if](#).

Definition at line 21 of file [CollectorDefaultImpl1.cpp](#).

```
00021 {  
00022     _numElements = 0;  
00023     if (_clearHandler != nullptr) {  
00024         _clearHandler();  
00025     }  
00026 }
```

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

Implements [Collector_if](#).

Definition at line 36 of file [CollectorDefaultImpl1.cpp](#).

```
00036 {  
00037     return this->_lastValue;  
00038 }
```

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

Implements [Collector_if](#).

Definition at line 40 of file [CollectorDefaultImpl1.cpp](#).

```
00040
00041     return this->_numElements;
00042 }
```

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

Implements [Collector_if](#).

Definition at line 44 of file [CollectorDefaultImpl1.cpp](#).

```
00044
00045     _addValueHandler = addValueHandler;
00046 }
```

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

Implements [Collector_if](#).

Definition at line 48 of file [CollectorDefaultImpl1.cpp](#).

```
00048
00049     _clearHandler = clearHandler;
00050 }
```

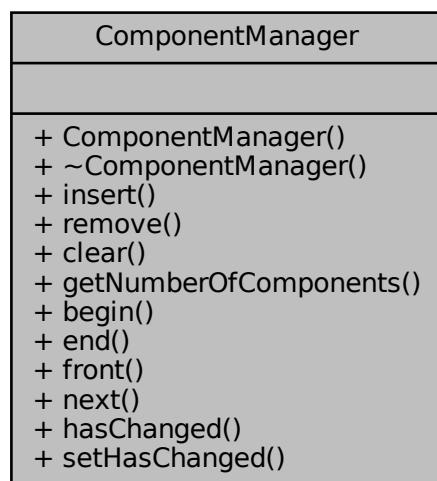
The documentation for this class was generated from the following files:

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

8.14 ComponentManager Class Reference

```
#include <ComponentManager.h>
```

Collaboration diagram for ComponentManager:



Public Member Functions

- `ComponentManager (Model *model)`
- `virtual ~ComponentManager ()=default`
- `bool insert (ModelComponent *comp)`
- `void remove (ModelComponent *comp)`
- `void clear ()`
- `unsigned int getNumberOfComponents ()`
- `std::list< ModelComponent * >::iterator begin ()`
- `std::list< ModelComponent * >::iterator end ()`
- `ModelComponent * front ()`
- `ModelComponent * next ()`
- `bool hasChanged () const`
- `void setHasChanged (bool _hasChanged)`

8.14.1 Detailed Description

Definition at line 22 of file [ComponentManager.h](#).

8.14.2 Constructor & Destructor Documentation

8.14.2.1 ComponentManager()

```
ComponentManager::ComponentManager (
    Model * model )
```

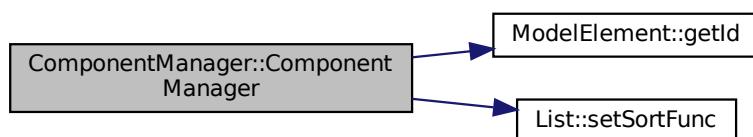
Components are sorted by ID

Definition at line 20 of file [ComponentManager.cpp](#).

```
00020
00021     _parentModel = model;
00022     _components = new List<ModelComponent*>();
00023     _components->setSortFunc([](const ModelComponent* a, const ModelComponent * b) {
00024         return a->getId() < b->getId();
00025     });
00026 }
```

References [ModelElement::getId\(\)](#), and [List< T >::setSortFunc\(\)](#).

Here is the call graph for this function:



8.14.2.2 ~ComponentManager() virtual ComponentManager::~ComponentManager () [virtual], [default]

8.14.3 Member Function Documentation

8.14.3.1 begin() std::list< ModelComponent * >::iterator ComponentManager::begin ()

Definition at line 60 of file [ComponentManager.cpp](#).

```
00060
00061     return _components->list ()->begin ();
00062 }
```

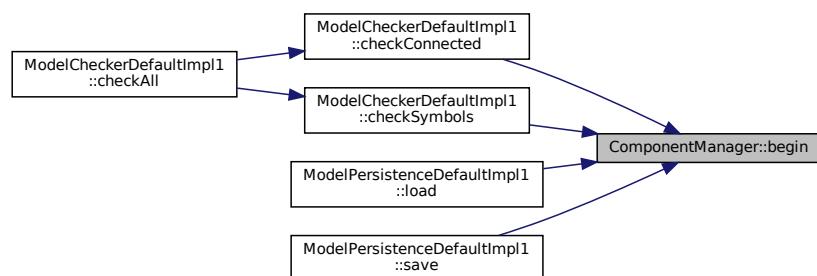
References [List< T >::list\(\)](#).

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), and [ModelPersistenceDefaultImpl1::save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.2 `clear()` void ComponentManager::clear ()

Definition at line 45 of file [ComponentManager.cpp](#).

```
00045     {
00046         this->_components->clear\(\);
00047         _hasChanged = true;
00048     }
```

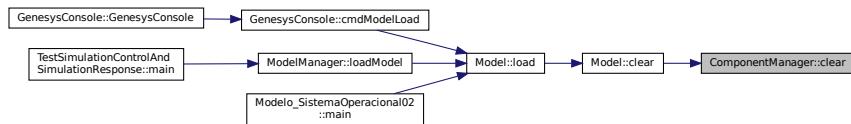
References [List< T >::clear\(\)](#).

Referenced by [Model::clear\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.3 `end()` std::list< [ModelComponent](#) * >::iterator ComponentManager::end ()

Definition at line 64 of file [ComponentManager.cpp](#).

```
00064     {
00065         return _components->list\(\)->end\(\);
00066     }
```

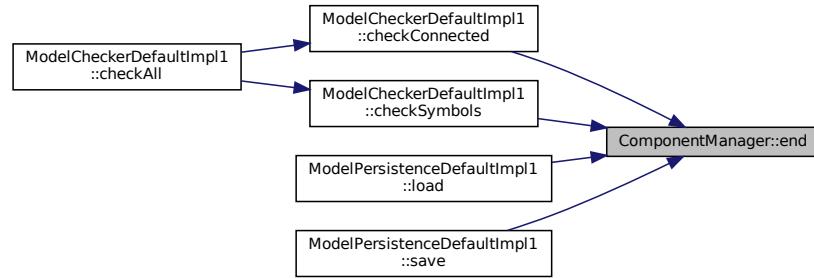
References [List< T >::list\(\)](#).

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), and [ModelPersistenceDefaultImpl1::save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.4 front() `ModelComponent * ComponentManager::front ()`

Definition at line 68 of file [ComponentManager.cpp](#).

```
00068
00069     return _components->front ();
00070 }
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



8.14.3.5 getNumberOfComponents() `unsigned int ComponentManager::getNumberOfComponents ()`

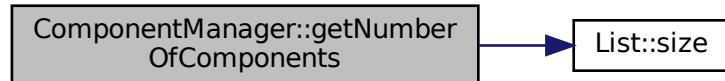
Definition at line 56 of file [ComponentManager.cpp](#).

```
00056
00057     return _components->size ();
00058 }
```

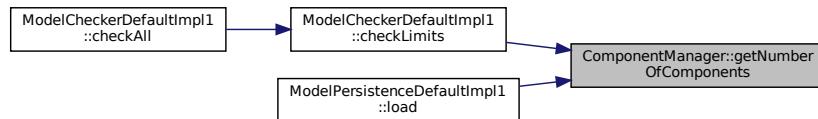
References [List< T >::size\(\)](#).

Referenced by [ModelCheckerDefaultImpl1::checkLimits\(\)](#), and [ModelPersistenceDefaultImpl1::load\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.6 `hasChanged()` `bool ComponentManager::hasChanged() const`

Definition at line 76 of file [ComponentManager.cpp](#).

```
00076 {  
00077     return _hasChanged;  
00078 }
```

Referenced by [Model::hasChanged\(\)](#).

Here is the caller graph for this function:



```
8.14.3.7 insert() bool ComponentManager::insert (
    ModelComponent * comp )
```

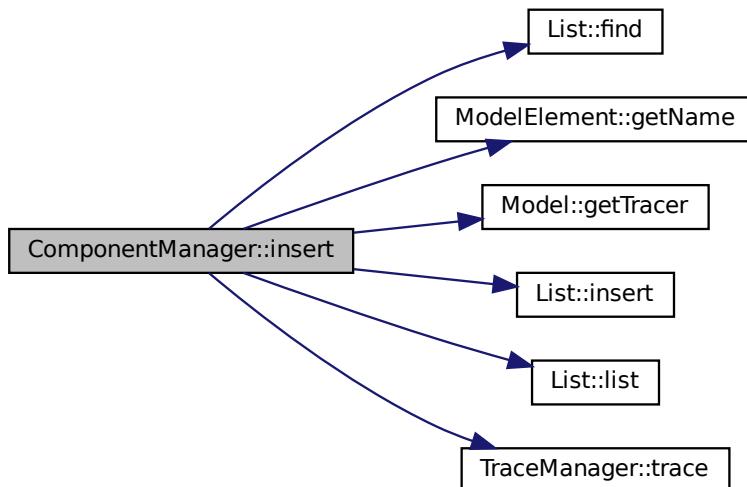
Definition at line 28 of file [ComponentManager.cpp](#).

```
00028     if (_components->find(comp) == _components->list()->end()) {
00029         _components->insert(comp);
00030         _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \""
00031             + comp->getName() + "\" successfully inserted");
00032         _hasChanged = true;
00033     }
00034     return true;
00035     _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \""
00036         + comp->getName() + "\" could not be inserted");
00037     return false;
00037 }
```

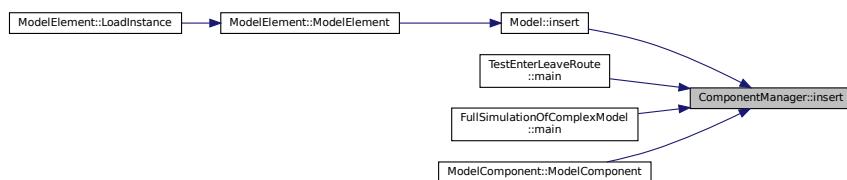
References [Util::componentResult](#), [List< T >::find\(\)](#), [ModelElement::getName\(\)](#), [Model::getTracer\(\)](#), [List< T >::insert\(\)](#), [List< T >::list\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [Model::insert\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), and [ModelComponent::ModelComponent\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.8 `next()`

Definition at line 72 of file [ComponentManager.cpp](#).

```
00072     {
00073         return _components->next();
00074     }
```

References [List< T >::next\(\)](#).

Here is the call graph for this function:



8.14.3.9 `remove()`

void ComponentManager::remove (

 ModelComponent * comp)

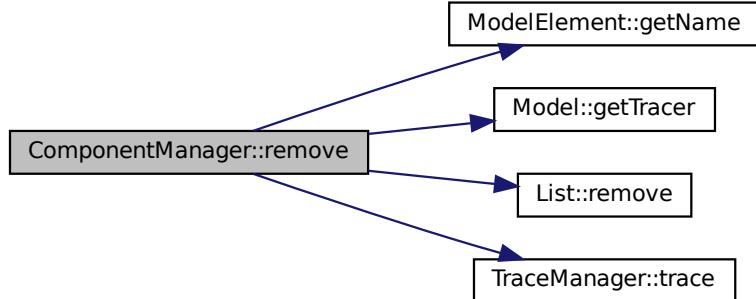
Definition at line 39 of file [ComponentManager.cpp](#).

```
00039
00040     _components->remove(comp);
00041     _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \""
00042         + comp->getName() + "\" successfully removed");
00043     _hasChanged = true;
00043 }
```

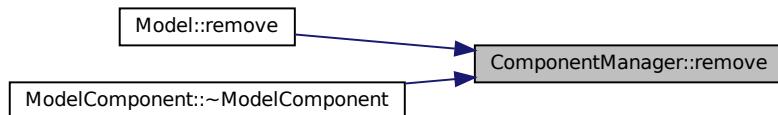
References [Util::componentResult](#), [ModelElement::getName\(\)](#), [Model::getTracer\(\)](#), [List< T >::remove\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [Model::remove\(\)](#), and [ModelComponent::~ModelComponent\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.10 setHasChanged() void ComponentManager::setHasChanged (bool _hasChanged)

Definition at line 80 of file [ComponentManager.cpp](#).

```
00080
00081     this->_hasChanged = _hasChanged;
00082 }
```

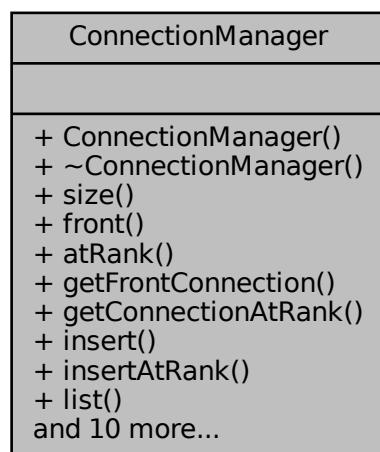
The documentation for this class was generated from the following files:

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

8.15 ConnectionManager Class Reference

```
#include <ConnectionManager.h>
```

Collaboration diagram for ConnectionManager:



Public Member Functions

- `ConnectionManager ()`
- `virtual ~ConnectionManager ()=default`
- `unsigned int size ()`
- `ModelComponent * front ()`
- `ModelComponent * atRank (unsigned int rank)`
- `Connection * getFrontConnection ()`
- `Connection * getConnectionAtRank (unsigned int rank)`
- `void insert (ModelComponent *component, unsigned int inputNumber=0)`
- `void insertAtRank (unsigned int rank, Connection *connection)`
- `std::list< Connection * > * list () const`
- `unsigned int getCurrentOutputConnections () const`
- `void setMaxOutputConnections (unsigned int _maxOutputConnections)`
- `unsigned int getMaxOutputConnections () const`
- `void setMinOutputConnections (unsigned int _minOutputConnections)`
- `unsigned int getMinOutputConnections () const`
- `unsigned int getCurrentInputConnections () const`
- `void setMaxInputConnections (unsigned int _maxInputConnections)`
- `unsigned int getMaxInputConnections () const`
- `void setMinInputConnections (unsigned int _minInputConnections)`
- `unsigned int getMinInputConnections () const`

8.15.1 Detailed Description

Definition at line 26 of file [ConnectionManager.h](#).

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `ConnectionManager()` `ConnectionManager::ConnectionManager ()`

Definition at line 18 of file [ConnectionManager.cpp](#).

```
00018 {  
00019 }
```

8.15.2.2 `~ConnectionManager()` `virtual ConnectionManager::~ConnectionManager () [virtual], [default]`

8.15.3 Member Function Documentation

```
8.15.3.1 atRank() ModelComponent * ConnectionManager::atRank (
    unsigned int rank )
```

DEPRECATED. Use getConnectionAtRank instead

Definition at line 29 of file [ConnectionManager.cpp](#).

```
00029
00030     return _nextConnections->getAtRank(rank)->first;
00031 }
```

References [List< T >::getAtRank\(\)](#).

Here is the call graph for this function:



```
8.15.3.2 front() ModelComponent * ConnectionManager::front ( )
```

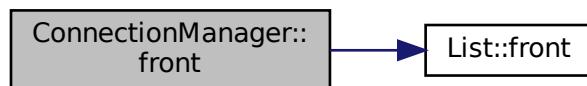
DEPRECATED. Use frontConnection instead

Definition at line 25 of file [ConnectionManager.cpp](#).

```
00025
00026     return _nextConnections->front ()->first;
00027 }
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



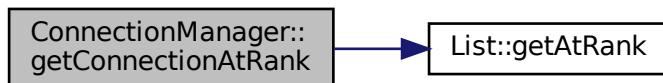
8.15.3.3 getConnectionAtRank() `Connection * ConnectionManager::getConnectionAtRank (`
 `unsigned int rank)`

Definition at line 37 of file [ConnectionManager.cpp](#).

```
00037
00038     return _nextConnections->getAtRank(rank);
00039 }
```

References [List< T >::getAtRank\(\)](#).

Here is the call graph for this function:



8.15.3.4 getCurrentInputConnections() `unsigned int ConnectionManager::getCurrentInputConnections`
 `() const`

Definition at line 83 of file [ConnectionManager.cpp](#).

```
00083
00084     return _currentInputConnections;
00085 }
```

8.15.3.5 getCurrentOutputConnections() `unsigned int ConnectionManager::getCurrentOutputConnections`
 `() const`

Definition at line 59 of file [ConnectionManager.cpp](#).

```
00059
00060     return _currentOutputConnections;
00061 }
```

8.15.3.6 getFrontConnection() `Connection * ConnectionManager::getFrontConnection ()`

Definition at line 33 of file [ConnectionManager.cpp](#).

```
00033
00034     return _nextConnections->front();
00035 }
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



8.15.3.7 getMaxInputConnections() `unsigned int ConnectionManager::getMaxInputConnections () const`

Definition at line 91 of file [ConnectionManager.cpp](#).

```
00091 {  
00092     return _maxInputConnections;  
00093 }
```

8.15.3.8 getMaxOutputConnections() `unsigned int ConnectionManager::getMaxOutputConnections () const`

Definition at line 67 of file [ConnectionManager.cpp](#).

```
00067 {  
00068     return _maxOutputConnections;  
00069 }
```

8.15.3.9 getMinInputConnections() `unsigned int ConnectionManager::getMinInputConnections () const`

Definition at line 99 of file [ConnectionManager.cpp](#).

```
00099 {  
00100     return _minInputConnections;  
00101 }
```

8.15.3.10 getMinOutputConnections() `unsigned int ConnectionManager::getMinOutputConnections () const`

Definition at line 75 of file [ConnectionManager.cpp](#).

```
00075 {  
00076     return _minOutputConnections;  
00077 }
```

```
8.15.3.11 insert() void ConnectionManager::insert (
    ModelComponent * component,
    unsigned int inputNumber = 0 )
```

Definition at line 41 of file [ConnectionManager.cpp](#).

```
00041
00042     _nextConnections->insert(new Connection(component, inputNumber));
00043     _currentOutputConnections++;
00044 }
```

{

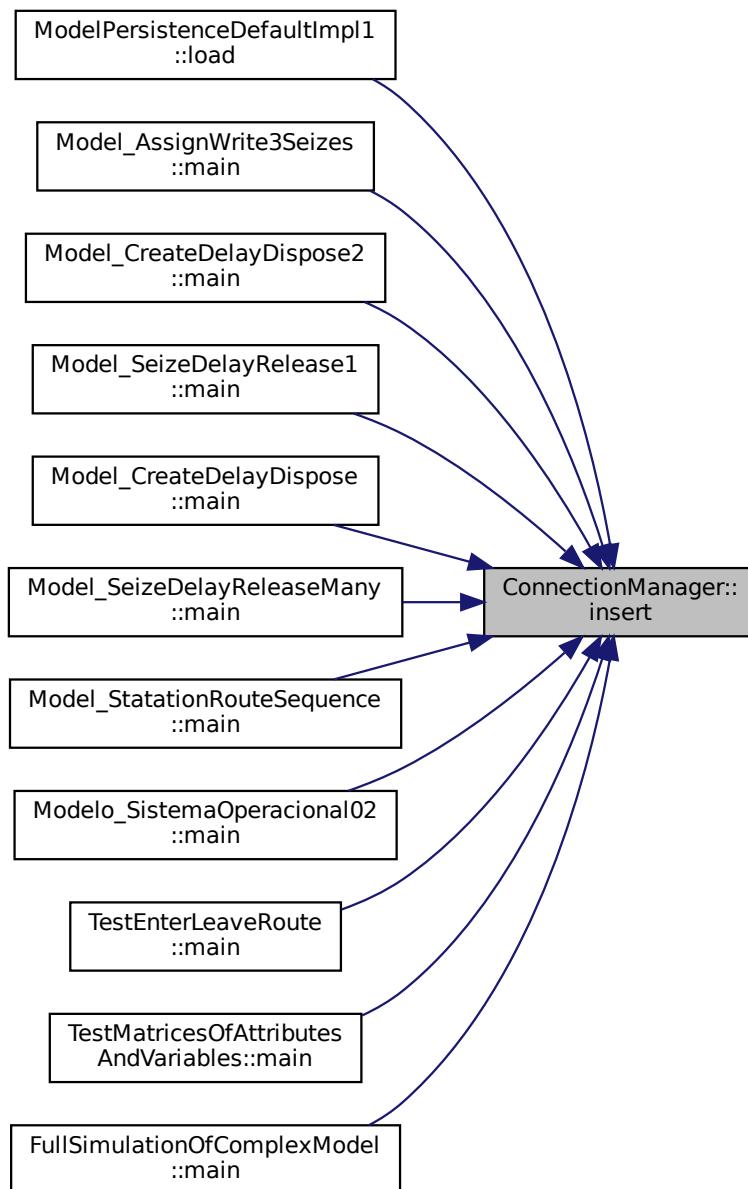
References [List< T >::insert\(\)](#).

Referenced by [ModelPersistenceDefaultImpl1::load\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.12 `insertAtRank()` void ConnectionManager::insertAtRank (unsigned int rank, Connection * connection)

Definition at line 46 of file [ConnectionManager.cpp](#).

```

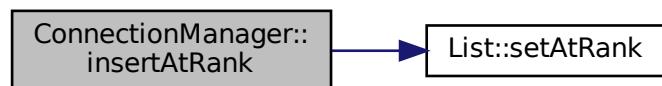
00046
00047     _nextConnections->setAtRank(rank, connection); // \TODO: it does not work if there is less than
00048     _currentOutputConnections++;
}

```

```
00049 }
```

References [List< T >::setAtRank\(\)](#).

Here is the call graph for this function:



8.15.3.13 list() std::list< Connection * > * ConnectionManager::list () const

Definition at line 51 of file [ConnectionManager.cpp](#).

```
00051
00052     return _nextConnections->list();
00053 }
```

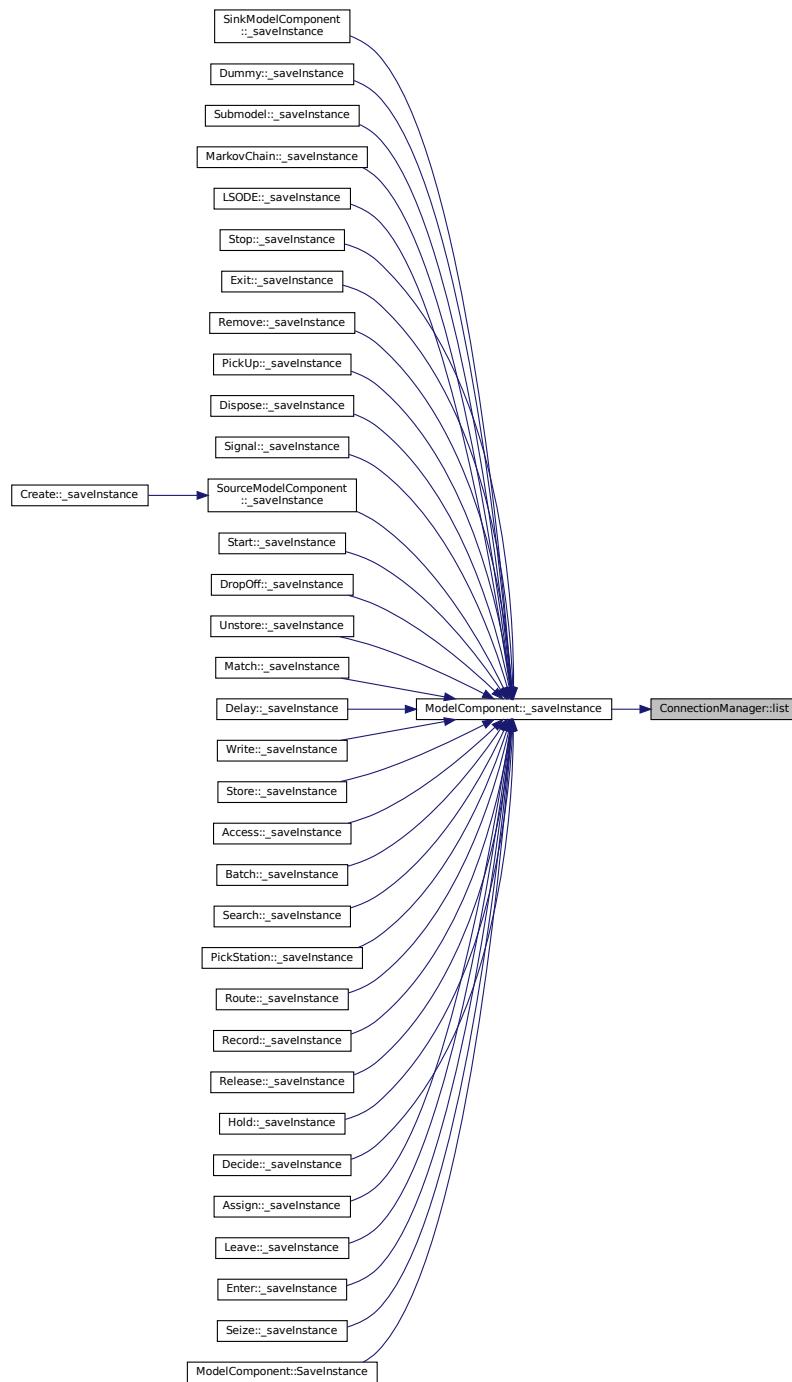
References [List< T >::list\(\)](#).

Referenced by [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.14 `setMaxInputConnections()` `void ConnectionManager::setMaxInputConnections (`
`unsigned int _maxInputConnections)`

Definition at line 87 of file [ConnectionManager.cpp](#).
00087

{

```
00088     this->_maxInputConnections = _maxInputConnections;
00089 }
```

Referenced by [Create::Create\(\)](#).

Here is the caller graph for this function:



8.15.3.15 setMaxOutputConnections() void ConnectionManager::setMaxOutputConnections (unsigned int *_maxOutputConnections*)

Definition at line 63 of file [ConnectionManager.cpp](#).

```
00063
00064     this->_maxOutputConnections = _maxOutputConnections;
00065 }
```

Referenced by [Dispose::Dispose\(\)](#).

Here is the caller graph for this function:



8.15.3.16 setMinInputConnections() void ConnectionManager::setMinInputConnections (unsigned int *_minInputConnections*)

Definition at line 95 of file [ConnectionManager.cpp](#).

```
00095
00096     this->_minInputConnections = _minInputConnections;
00097 }
```

Referenced by [Create::Create\(\)](#).

Here is the caller graph for this function:



8.15.3.17 setMinOutputConnections() void ConnectionManager::setMinOutputConnections (unsigned int _minOutputConnections)

Definition at line 71 of file [ConnectionManager.cpp](#).

```
00071
00072     this->_minOutputConnections = _minOutputConnections;
00073 }
```

Referenced by [Dispose::Dispose\(\)](#).

Here is the caller graph for this function:



8.15.3.18 size() unsigned int ConnectionManager::size ()

Definition at line 21 of file [ConnectionManager.cpp](#).

```
00021
00022     return _nextConnections->size();
00023 }
```

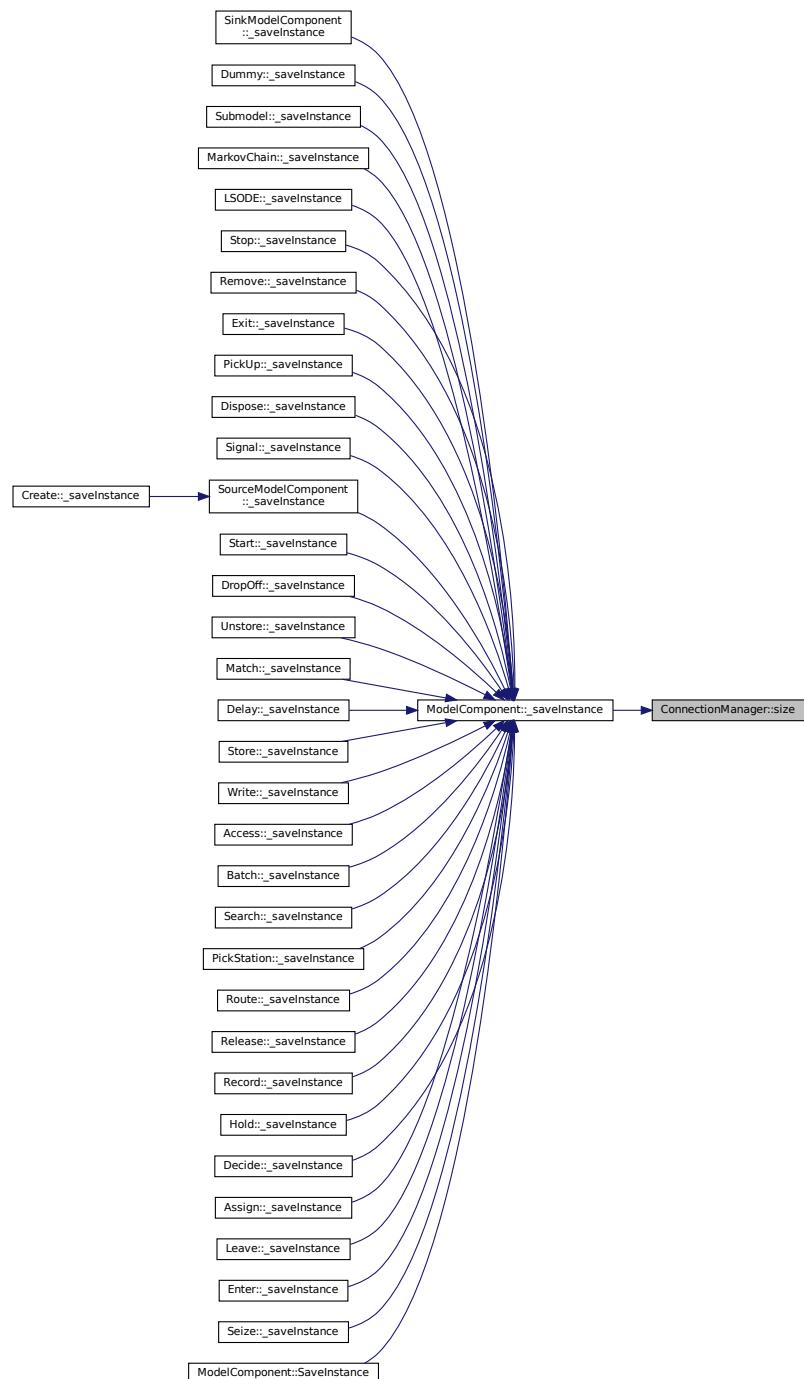
References [List< T >::size\(\)](#).

Referenced by [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



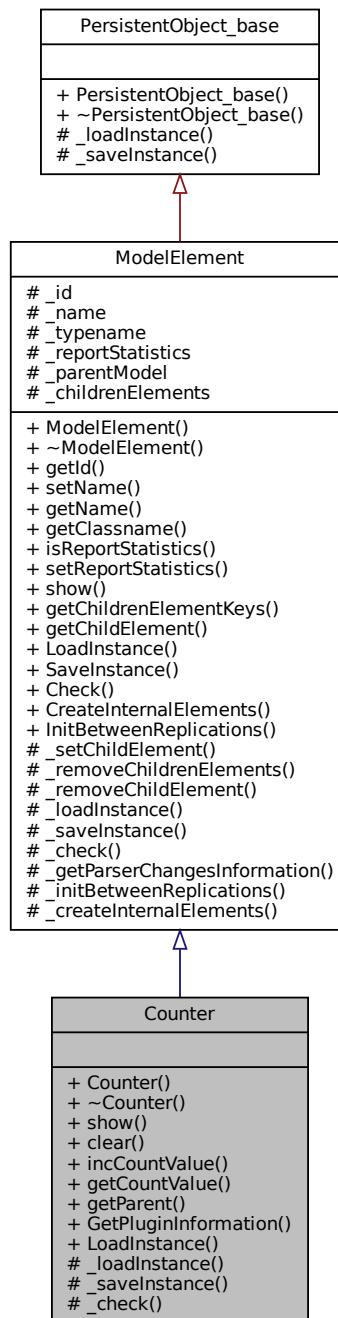
The documentation for this class was generated from the following files:

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

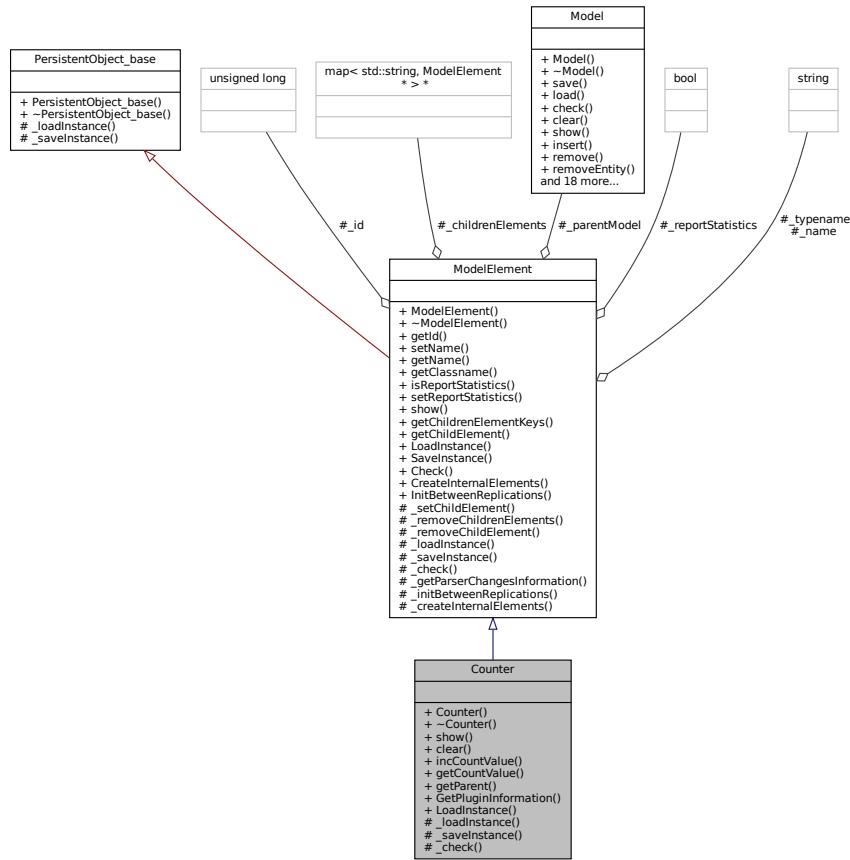
8.16 Counter Class Reference

```
#include <Counter.h>
```

Inheritance diagram for Counter:



Collaboration diagram for Counter:



Public Member Functions

- `Counter (Model *model, std::string name="", ModelElement *parent=nullptr)`
- virtual `~Counter ()=default`
- virtual `std::string show ()`
- void `clear ()`
- void `incCountValue (int value=1)`
- unsigned long `getCountValue () const`
- `ModelElement * getParent () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.16.1 Detailed Description

The [Counter](#) element is used to count events, and its internal count value is added by a configurable amount, usually incremented by one.

Definition at line 25 of file [Counter.h](#).

8.16.2 Constructor & Destructor Documentation

```
8.16.2.1 Counter() Counter::Counter (
    Model * model,
    std::string name = "",
    ModelElement * parent = nullptr )
```

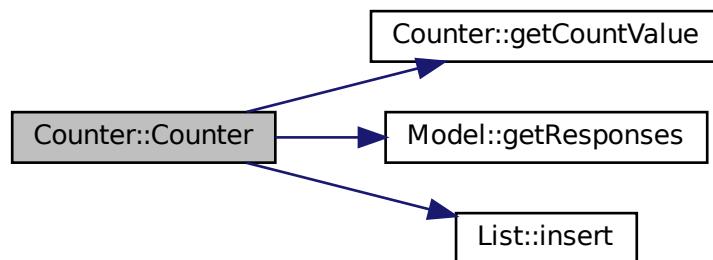
Definition at line 20 of file [Counter.cpp](#).

```
00020     Util::TypeOf<Counter>(), name) {
00021     _parent = parent;
00022     GetterMember getterMember = DefineGetterMember<Counter>(this, &Counter::getCountValue);
00023     //std::string parentName = "";
00024     //if (_parent != nullptr)
00025     //    parentName = _parent->name();
00026     SimulationResponse* resp = new SimulationResponse(Util::TypeOf<Counter>(),
00027             /*parentName + ":" + */
00028             _parentModel->getResponses()->insert(resp);
00029 }
```

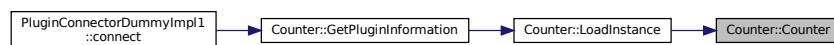
References [ModelElement::_name](#), [ModelElement::_parentModel](#), [getCountValue\(\)](#), [Model::getResponses\(\)](#), and [List< T >::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.2.2 ~Counter() virtual Counter::~Counter () [virtual], [default]

8.16.3 Member Function Documentation

8.16.3.1 _check() bool Counter::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 79 of file [Counter.cpp](#).

```
00079
00080     return true;
00081 }
```

8.16.3.2 _loadInstance() bool Counter::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 70 of file [Counter.cpp](#).

```
00070
00071     return ModelElement::_loadInstance(fields);
00072 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.3.3 `_saveInstance()` `std::map< std::string, std::string > * Counter::_saveInstance ()`
 [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 74 of file [Counter.cpp](#).

```
00074
00075     return ModelElement::_saveInstance();
00076 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



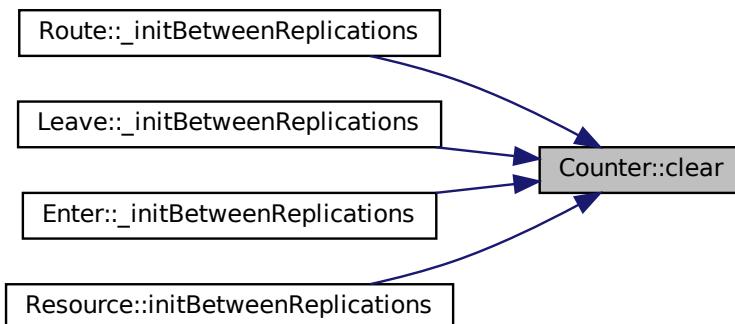
8.16.3.4 `clear()` `void Counter::clear ()`

Definition at line 38 of file [Counter.cpp](#).

```
00038
00039     _count = 0;
00040 }
```

Referenced by [Route::_initBetweenReplications\(\)](#), [Leave::_initBetweenReplications\(\)](#), [Enter::_initBetweenReplications\(\)](#), and [Resource::initBetweenReplications\(\)](#).

Here is the caller graph for this function:



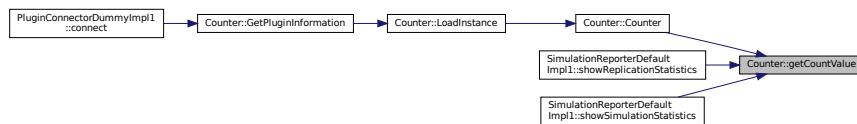
8.16.3.5 getCountValue() `unsigned long Counter::getCountValue() const`

Definition at line 46 of file [Counter.cpp](#).

```
00046
00047     return _count;
00048 }
```

Referenced by [Counter\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



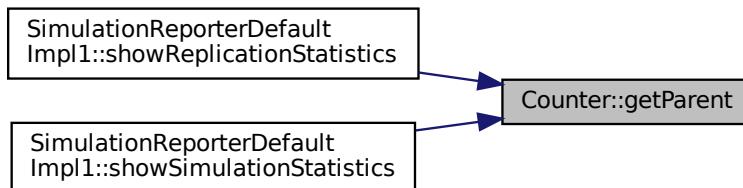
8.16.3.6 getParent() `ModelElement * Counter::getParent() const`

Definition at line 50 of file [Counter.cpp](#).

```
00050
00051     return _parent;
00052 }
```

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.16.3.7 GetPluginInformation() `PluginInformation * Counter::GetPluginInformation () [static]`

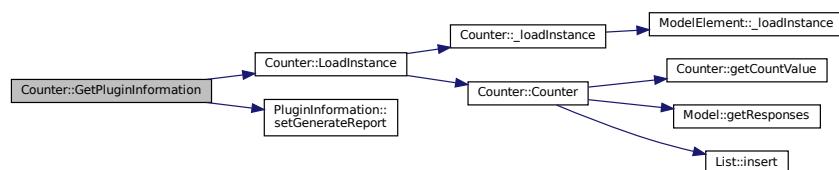
Definition at line 54 of file [Counter.cpp](#).

```
00054     {
00055         PluginInformation* info = new PluginInformation(Util::TypeOf<Counter>(), &Counter::LoadInstance);
00056         info->setGenerateReport(true);
00057         return info;
00058     }
```

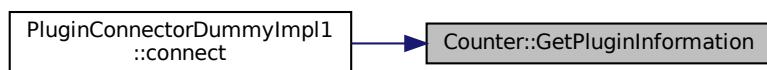
References [LoadInstance\(\)](#), and [PluginInformation::setGenerateReport\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



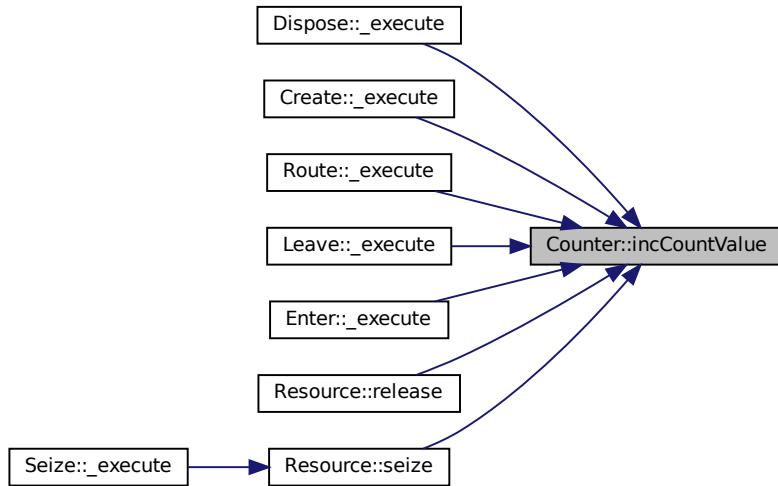
8.16.3.8 incCountValue() `void Counter::incCountValue (int value = 1)`

Definition at line 42 of file [Counter.cpp](#).

```
00042     {
00043         _count += value;
00044     }
```

Referenced by [Dispose::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Leave::_execute\(\)](#), [Enter::_execute\(\)](#), [Resource::release\(\)](#), and [Resource::seize\(\)](#).

Here is the caller graph for this function:



8.16.3.9 LoadInstance() `ModelElement * Counter::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

Definition at line 60 of file `Counter.cpp`.

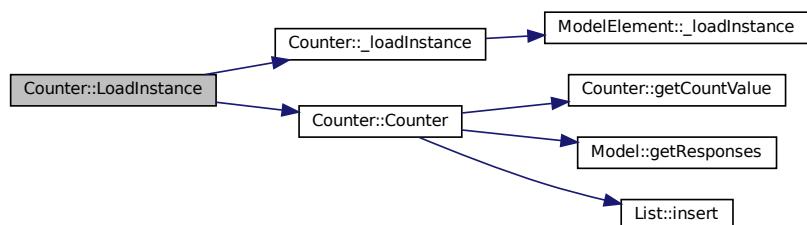
```

00060
00061     Counter* newElement = new Counter(model);
00062     try {
00063         newElement->_loadInstance(fields);
00064     } catch (const std::exception& e) {
00065     }
00066
00067     return newElement;
00068 }
```

References [_loadInstance\(\)](#), and [Counter\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.16.3.10 show() std::string Counter::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 32 of file [Counter.cpp](#).

```
00032     {
00033         return ModelElement::show() +
00034             ", count=" + std::to_string(this->_count);
00035     }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



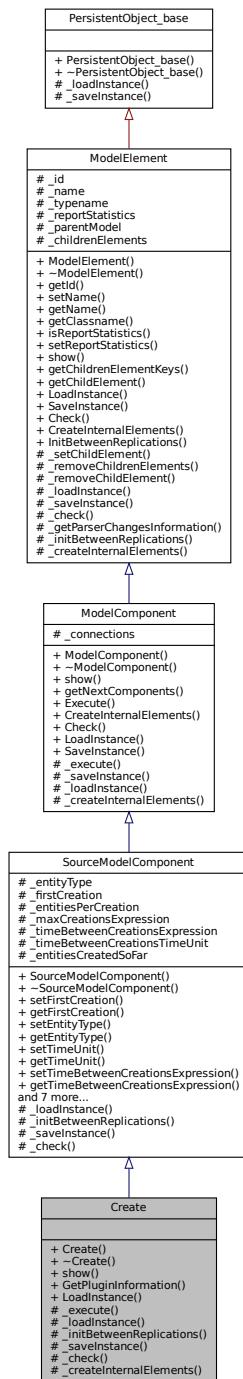
The documentation for this class was generated from the following files:

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

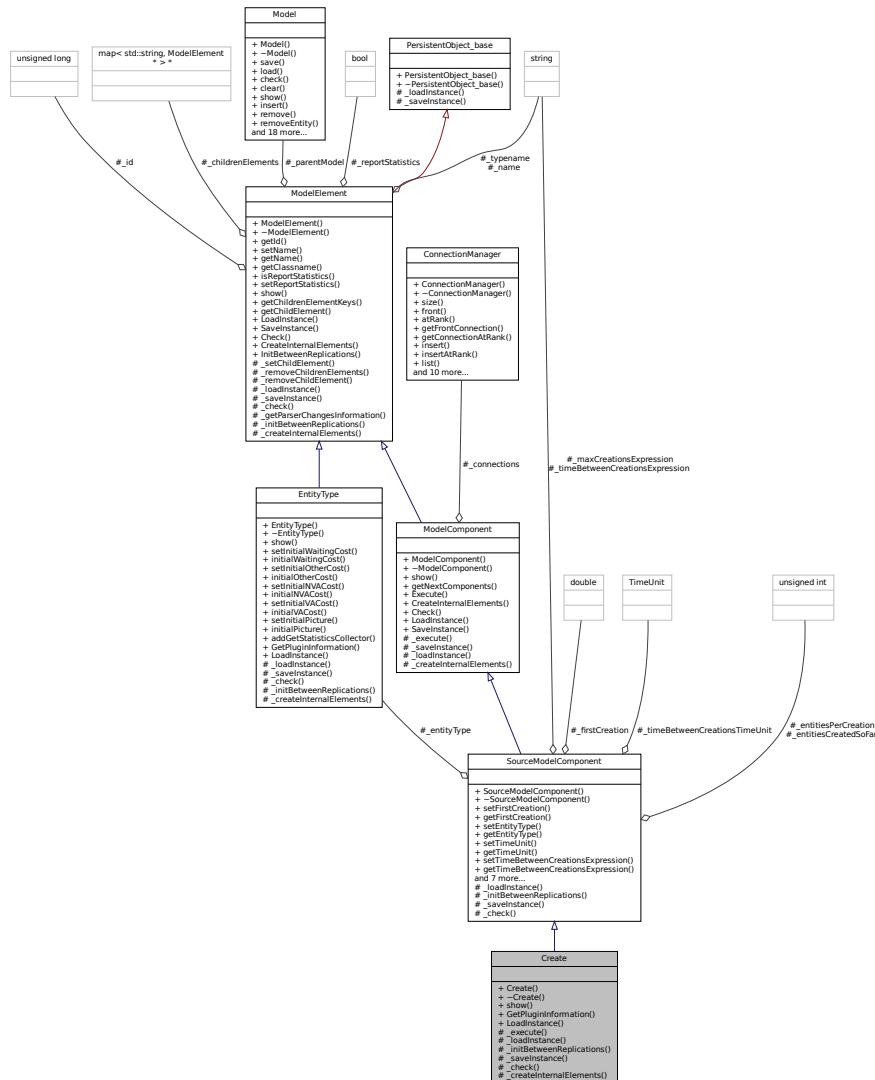
8.17 Create Class Reference

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

Inheritance diagram for Create:



Collaboration diagram for Create:



Public Member Functions

- `Create (Model *model, std::string name="")`
- virtual `~Create ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.17.1 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**) **Create** module DESCRIPTION This module is intended as the starting point for entities in a simulation model. Entities are created using a schedule or based on a time between arrivals. Entities then leave the module to begin processing through the system. The entity type is specified in this module. TYPIC← AL USES The start of a part's production in a manufacturing line A document's arrival (for example, order, check, application) into a business process A customer's arrival at a service process (for example, retail store, restaurant, information desk) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Entity Type Name of the entity type to be generated. Type Type of arrival stream to be generated. Types include Random (uses an exponential distribution, user specifies mean), Schedule (uses an exponential distribution, mean determined from the specified **Schedule** module), Constant (user specifies constant value; for example, 100), or Expression (drop-down list of various distributions). Value Determines the mean of the exponential distribution (if Random is used) or the constant value (if Constant is used) for the time between arrivals. Applies only when Type is Random or Constant. Schedule Name Identifies the name of the schedule to be used. The schedule defines the arrival pattern for entities arriving to the system. Applies only when Type is **Schedule**. Expression Any distribution or value specifying the time between arrivals. Applies only when Type is Expression. Units Time units used for interarrival and first creation times. Does not apply when Type is **Schedule**. Entities per Arrival Number of entities that will enter the system at a given time with each arrival. Max Arrivals Maximum number of entities that this module will generate. When this value is reached, the creation of new entities by this module ceases. First Creation Starting time for the first entity to arrive into the system. Does not apply when Type is **Schedule**.

Definition at line 67 of file [Create.h](#).

8.17.2 Constructor & Destructor Documentation

```
8.17.2.1 Create() Create::Create (
    Model * model,
    std::string name = "")
```

Definition at line 21 of file [Create.cpp](#).

```
00021 : SourceModelComponent(model, Util::TypeOf<Create>(), name)
00022 {
00023     //_numberOut = new Counter(_parentModel, _name + "." + "Count_number_in", this);
00024     //\todo Check if element has already been inserted and this is not needed:
00025     _parentModel->elements()->insert(_numberOut);
00026     _connections->setMinInputConnections(0);
00027     _connections->setMaxInputConnections(0);
```

```

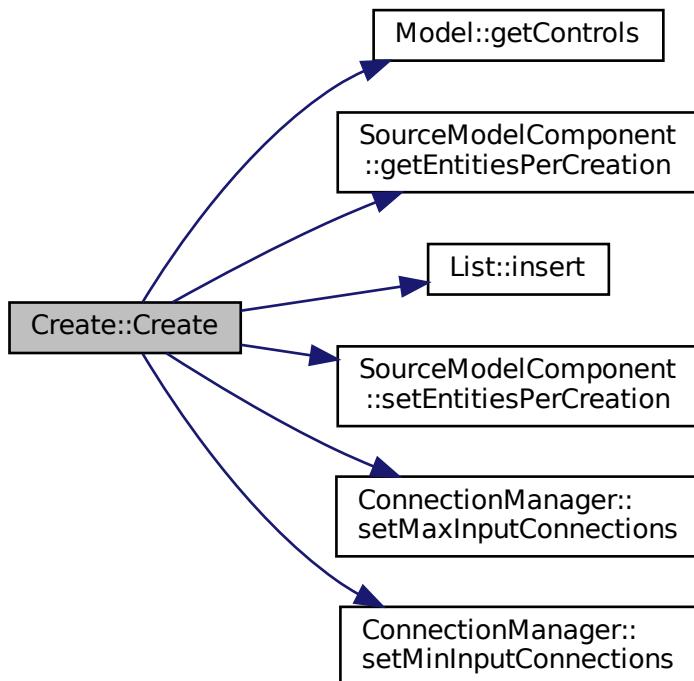
00026     GetterMember getter = DefineGetterMember<SourceModelComponent>(this,
00027     &Create::getEntitiesPerCreation);
00028     SetterMember setter = DefineSetterMember<SourceModelComponent>(this,
00029     &Create::setEntitiesPerCreation);
00030     model->getControls()->insert(new SimulationControl(Util::TypeOf<Create>(), _name +
00031         ".EntitiesPerCreation", getter, setter));
00032     /* \todo:
00033     model->getControls()->insert(new SimulationControl(Util::TypeOf<Create>(), "Time Between
00034     Creations",
00035         DefineGetterMember<SourceModelComponent>(this, &Create::getTimeBetweenCreationsExpression),
00036         DefineSetterMember<SourceModelComponent>(this, &Create::setTimeBetweenCreationsExpression))
00037     );
00038     */
00039 }
00040

```

References [ModelComponent::_connections](#), [ModelElement::_name](#), [Model::getControls\(\)](#), [SourceModelComponent::getEntitiesPerCreation\(\)](#), [List< T >::insert\(\)](#), [SourceModelComponent::setEntitiesPerCreation\(\)](#), [ConnectionManager::setMaxInputConnections\(\)](#), and [ConnectionManager::setMinInputConnections\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.17.2.2 ~Create() virtual Create::~Create () [virtual], [default]

8.17.3 Member Function Documentation

8.17.3.1 _check() bool Create::_check (std::string * errorMessage) [protected], [virtual]

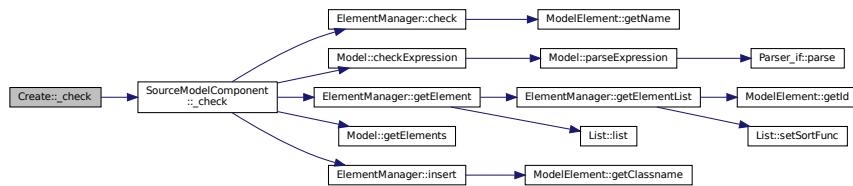
Reimplemented from [SourceModelComponent](#).

Definition at line 99 of file [Create.cpp](#).

```
00099           {
00100     bool resultAll = SourceModelComponent::_check(errorMessage);
00101     return resultAll;
00102 }
```

References [SourceModelComponent::_check\(\)](#).

Here is the call graph for this function:



8.17.3.2 _createInternalElements() void Create::_createInternalElements () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

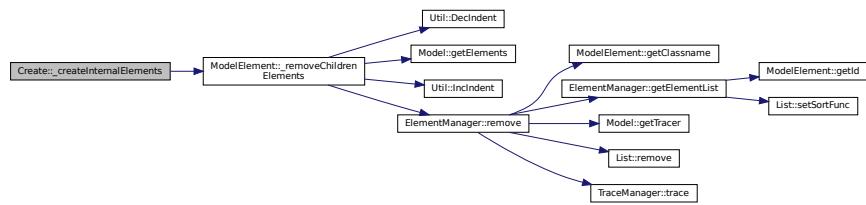
Reimplemented from [ModelComponent](#).

Definition at line 104 of file [Create.cpp](#).

```
00104 {
00105   if (_reportStatistics && _numberOut == nullptr) {
00106     _numberOut = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00107     _childrenElements->insert("CountNumberIn", _numberOut);
00108     // \todo _childrenElements->insert("Count_number_in", _numberOut);
00109   } else if (!_reportStatistics && _numberOut != nullptr) {
00110     this->_removeChildrenElements();
00111     // \todo _childrenElements->remove("Count_number_in");
00112     //_numberOut->~Counter();
00113     _numberOut = nullptr;
00114   }
00115 }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildren](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



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

Implements [ModelComponent](#).

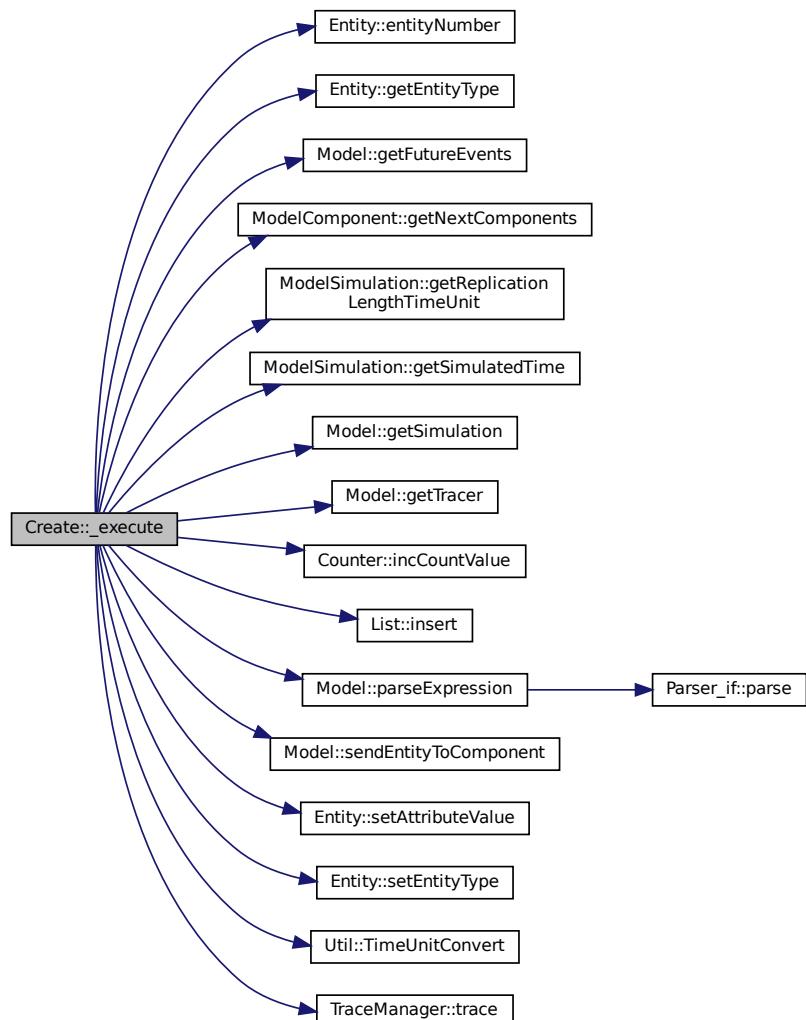
Definition at line 41 of file [Create.cpp](#).

```

00041 {
00042     double tnow = _parentModel->getSimulation()->getSimulatedTime();
00043     entity->setAttributeValue("Entity.ArrivalTime", tnow); // 
00044     ->find("Entity.ArrivalTime")->second->setValue(tnow);
00045     //entity->setAttributeValue("Entity.Picture", 1); // 
00046     ->find("Entity.ArrivalTime")->second->setValue(tnow);
00047     double timeBetweenCreations, timeScale, newArrivalTime;
00048     unsigned int _maxCreations = _parentModel->parseExpression(this->_maxCreationsExpression);
00049     for (unsigned int i = 0; i<this->_entitiesPerCreation; i++) {
00050         if (_entitiesCreatedSoFar < _maxCreations) {
00051             _entitiesCreatedSoFar++;
00052             Entity* newEntity = new Entity(_parentModel);
00053             newEntity->setEntityType(entity->getEntityType());
00054             //_parentModel->elements()->insert(newEntity); // ->getEntities()->insert(newEntity);
00055             timeBetweenCreations =
00056               _parentModel->parseExpression(this->_timeBetweenCreationsExpression);
00057             timeScale = Util::TimeUnitConvert(this->_timeBetweenCreationsTimeUnit,
00058               _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00059             newArrivalTime = tnow + timeBetweenCreations*timeScale;
00060             Event* newEvent = new Event(newArrivalTime, newEntity, this);
00061             _parentModel->getFutureEvents()->insert(newEvent);
00062             _parentModel->getTracer()->trace("Arrival of entity " +
00063               std::to_string(newEntity->entityNumber()) + " schedule for time " + std::to_string(newArrivalTime));
00064             //_model->getTrace()->trace("Arrival of entity "+std::to_string(entity->getId()) + " 
00065             schedule for time " +std::to_string(newArrivalTime));
00066         }
00067     }
00068     if (_reportStatistics)
00069         _numberOut->incCountValue();
00070     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00071 }
  
```

References [SourceModelComponent::_entitiesCreatedSoFar](#), [SourceModelComponent::_entitiesPerCreation](#), [SourceModelComponent::_maxCreationsExpression](#), [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [SourceModelComponent::_timeBetweenCreationsExpression](#), [SourceModelComponent::_timeBetweenCreationsTimeUnit](#), [Entity::entityNumber\(\)](#), [Entity::getEntityType\(\)](#), [Model::getFutureEvents\(\)](#), [ModelComponent::getNextComponents\(\)](#), [ModelSimulation::getReplicationLengthTimeUnit\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [Counter::incCountValue\(\)](#), [List< T >::insert\(\)](#), [Model::parseExpression\(\)](#), [Model::sendEntityToComponent\(\)](#), [Entity::setAttributeValue\(\)](#), [Entity::setEntityType\(\)](#), [Util::TimeUnitConvert\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.17.3.4 `_initBetweenReplications()` void Create::_initBetweenReplications () [protected], [virtual]

Reimplemented from [SourceModelComponent](#).

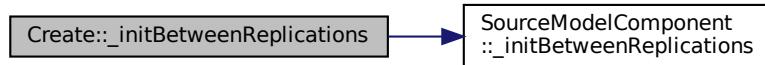
Definition at line 90 of file [Create.cpp](#).

```

00090
00091     SourceModelComponent::_initBetweenReplications ();
00092 }
```

References [SourceModelComponent::_initBetweenReplications\(\)](#).

Here is the call graph for this function:



8.17.3.5 `_loadInstance()` `bool Create::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [SourceModelComponent](#).

Definition at line 86 of file [Create.cpp](#).

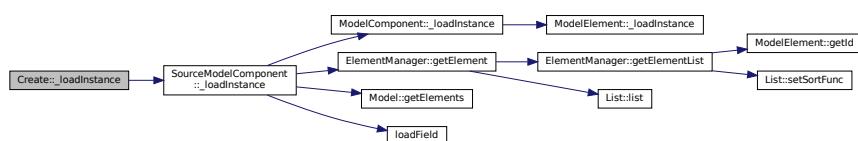
```

00086
00087     return SourceModelComponent::_loadInstance(fields);
00088 }
```

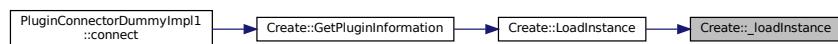
References [SourceModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

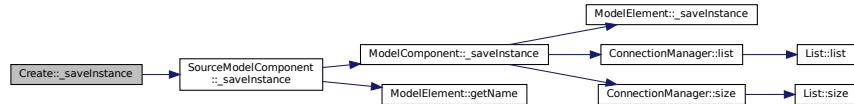
Reimplemented from [SourceModelComponent](#).

Definition at line 94 of file [Create.cpp](#).

```
00094     {
00095         std::map<std::string, std::string>* fields = SourceModelComponent::\_saveInstance\(\);
00096         return fields;
00097     }
```

References [SourceModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.17.3.7 `GetPluginInformation()` `PluginInformation * Create::GetPluginInformation ()` [static]

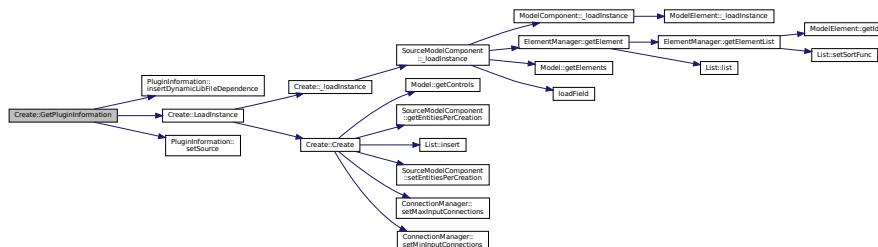
Definition at line 67 of file [Create.cpp](#).

```
00067     {
00068         PluginInformation* info = new PluginInformation(Util::TypeOf<Create>(), &Create::LoadInstance);
00069         info->setSource(true);
00070         info->insertDynamicLibFileDependence("attribute.so");
00071         info->insertDynamicLibFileDependence("entitytype.so");
00072         info->insertDynamicLibFileDependence("statisticscollector.so");
00073         return info;
00074     }
```

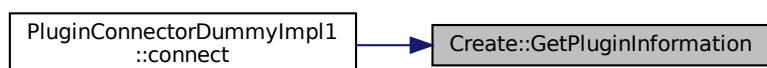
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), [LoadInstance\(\)](#), and [PluginInformation::setSource\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.17.3.8 LoadInstance() ModelComponent * Create::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

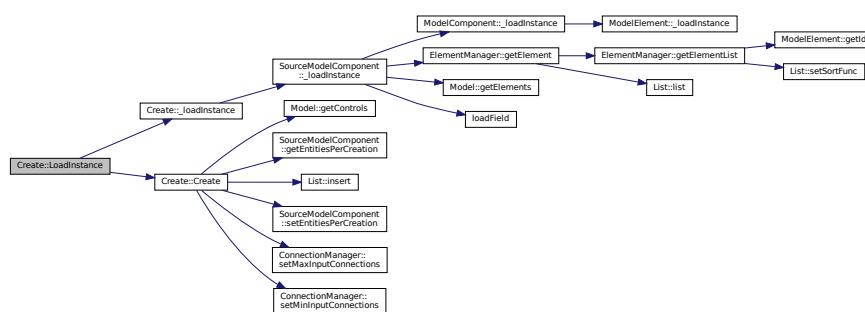
Definition at line 76 of file [Create.cpp](#).

```
00076
00077     Create* newComponent = new Create(model);
00078     try {
00079         newComponent->_loadInstance(fields);
00080     } catch (const std::exception& e) {
00081     }
00082 }
00083     return newComponent;
00084 }
```

References [_loadInstance\(\)](#), and [Create\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

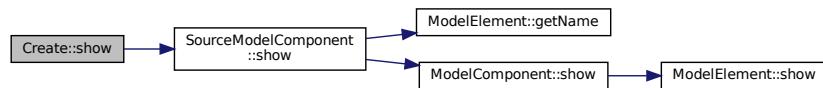
Reimplemented from [SourceModelComponent](#).

Definition at line 37 of file [Create.cpp](#).

```
00037
00038     {
00039     return SourceModelComponent::show();
00040 }
```

References [SourceModelComponent::show\(\)](#).

Here is the call graph for this function:



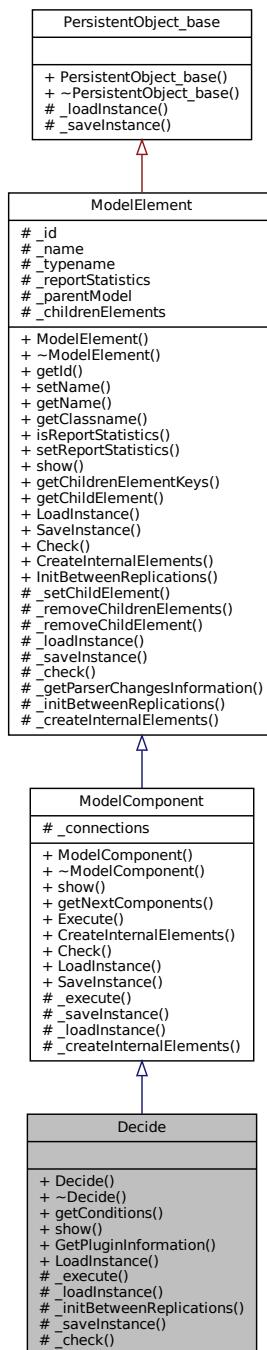
The documentation for this class was generated from the following files:

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

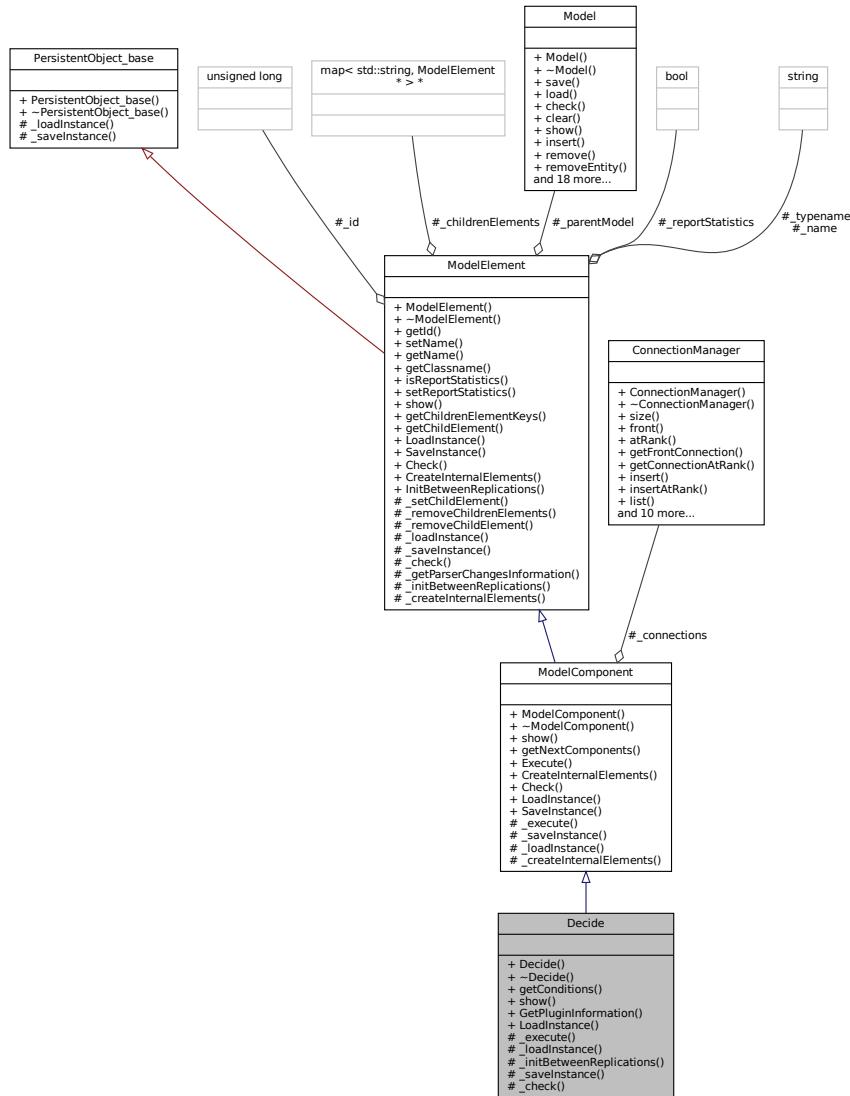
8.18 Decide Class Reference

```
#include <Decide.h>
```

Inheritance diagram for Decide:



Collaboration diagram for Decide:



Public Member Functions

- `Decide (Model *model, std::string name="")`
- virtual `~Decide ()=default`
- `List< std::string > * getConditions () const`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void [_execute](#) (Entity *entity)
- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual void [_initBetweenReplications](#) ()
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)

Additional Inherited Members

8.18.1 Detailed Description

Decide module DESCRIPTION This module allows for decision-making processes in the system. It includes options to make decisions based on one or more conditions (for example, if entity type is Gold Card) or based on one or more probabilities (for example, 75%, true; 25%, false). Conditions can be based on attribute values (for example, Priority), variable values (for example, Number Denied), the entity type, or an expression (for example, NQ(ProcessA.Queue)). There are two exit points out of the **Decide** module when its specified type is either 2-way by Chance or 2-way by Condition. There is one exit point for “true” entities and one for “false” entities. When the N-way by Chance or by Condition type is specified, multiple exit points are shown for each condition or probability and a single “else” exit. The number of entities that exit from each type (true/false) is displayed for 2-way by Chance or by Condition modules only. TYPICAL USES Dispatching a faulty part for rework Branching accepted vs. rejected checks Sending priority customers to a dedicated process Prompt Description Name Unique module identifier displayed on the module shape. Type Indicates whether the decision is based on a condition (if X>Y) or by chance/percentage (for example, 60%, yes; 40%, no). The type can be specified as either 2-way or N-way. 2-way allows for one condition or probability (plus the “false” exit). N-way allows for any number of conditions or probabilities to be specified as well as an “else” exit. Conditions Defines one or more conditions used to direct entities to different modules. Applies only when Type is N-way by Condition. Percentages Defines one or more percentages used to direct entities to different modules. Applies only when Type is N-way by Chance. Percent True Value that will be checked to determine the percentage of entities sent out a given True exit. If Types of conditions that are available for evaluation: **Variable**, **Variable Array (1D)**, **Variable Array (2D)**, **Attribute**, **Entity Type**, **Expression**. Named Specifies the name of the variable, attribute, or entity type that will be evaluated when an entity enters the module. Does not apply when Type is Expression. Is Evaluator for the condition. Applies only to **Attribute** and **Variable** conditions. Row Specifies the row index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and **Variable** is Array 1-D or Array 2-D. Column Specifies the column index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and **Variable** is Array 1-D or Array 2-D. Value Expression that will be either compared to an attribute or variable or that will be evaluated as a single expression to determine if it is true or false. Does not apply to **Entity Type** condition. If Type is Expression, this value must also include the evaluator (for example, Color<>Red).

Definition at line 73 of file [Decide.h](#).

8.18.2 Constructor & Destructor Documentation

```
8.18.2.1 Decide() Decide::Decide (
    Model * model,
    std::string name = "" )
```

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

```
00017 : ModelComponent(model, Util::TypeOf<Decide>(), name) {
00018 }
```

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

Here is the caller graph for this function:



```
8.18.2.2 ~Decide() virtual Decide::~Decide () [virtual], [default]
```

8.18.3 Member Function Documentation

```
8.18.3.1 _check() bool Decide::_check (
    std::string * errorMessage ) [protected], [virtual]
```

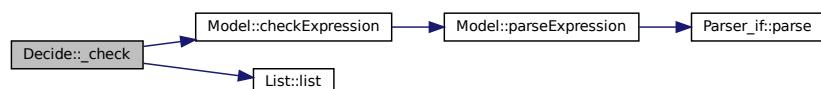
Reimplemented from [ModelElement](#).

Definition at line 68 of file [Decide.cpp](#).

```
00068 {
00069     bool allResult = true;
00070     std::string condition;
00071     for (std::list<std::string>::iterator it = _conditions->list()>begin(); it != _conditions->list()>end(); it++) {
00072         condition = (*it);
00073         allResult &= _parentModel->checkExpression(condition, "condition", errorMessage);
00074     }
00075     return allResult;
00076 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



```
8.18.3.2 _execute() void Decide::_execute (
    Entity * entity) [protected], [virtual]
```

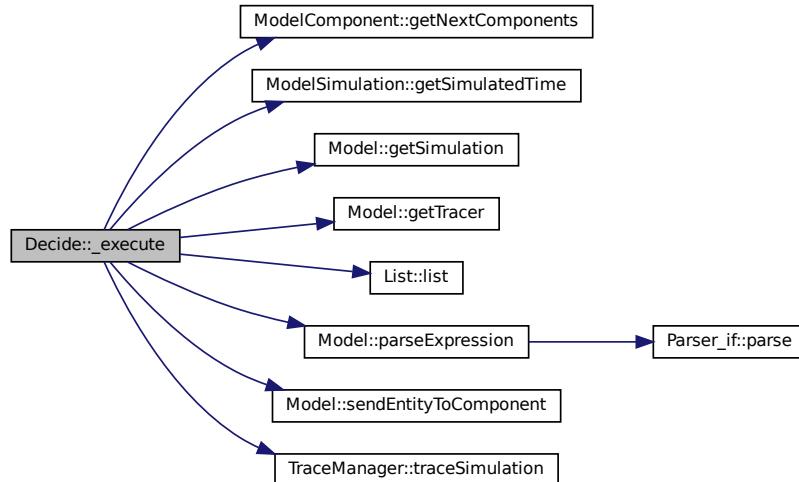
Implements [ModelComponent](#).

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

```
00028     {
00029         double value;
00030         unsigned short i = 0;
00031         for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->list()->end(); it++) {
00032             value = \_parentModel->parseExpression((*it));
00033             \_parentModel->getTracer()->traceSimulation(\_parentModel->getSimulation()->getSimulatedTime(),
00034                 entity, this, std::to_string(i + 1) + "th condition evaluated to " + std::to_string(value) + " // "
00035                 + (*it));
00036             if (value) {
00037                 \_parentModel->sendEntityToComponent(entity,
00038                     this->getNextComponents()->getConnectionAtRank(i), 0.0);
00039             }
00040             \_parentModel->getTracer()->traceSimulation(\_parentModel->getSimulation()->getSimulatedTime(),
00041                 entity, this, "No condition has been evaluated true");
00042             \_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getConnectionAtRank(i),
00043                 0.0);
00042 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [List< T >::list\(\)](#), [Model::parseExpression\(\)](#), [Model::sendEntityToComponent\(\)](#), and [TraceManager::traceSimulation\(\)](#).

Here is the call graph for this function:



```
8.18.3.3 _initBetweenReplications() void Decide::_initBetweenReplications () [protected],
[virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 44 of file [Decide.cpp](#).

```
00044     {
00045 }
```

8.18.3.4 `_loadInstance()` `bool Decide::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

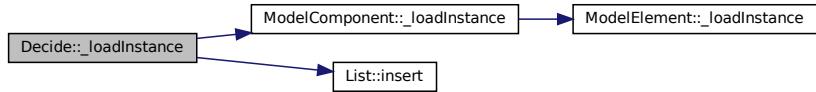
Definition at line 47 of file [Decide.cpp](#).

```
00047                                         {
00048     bool res = ModelComponent::_loadInstance(fields);
00049     if (res) {
00050         unsigned int nv = std::stoi((*(fields->find("conditions"))).second);
00051         for (unsigned int i = 0; i < nv; i++) {
00052             this->_conditions->insert((*(fields->find("condition" + std::to_string(i)))).second);
00053         }
00054     }
00055     return res;
00056 }
```

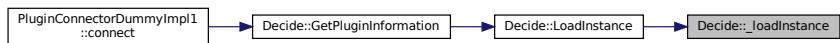
References [ModelComponent::_loadInstance\(\)](#), and [List< T >::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.3.5 `_saveInstance()` `std::map< std::string, std::string > * Decide::_saveInstance () [protected], [virtual]`

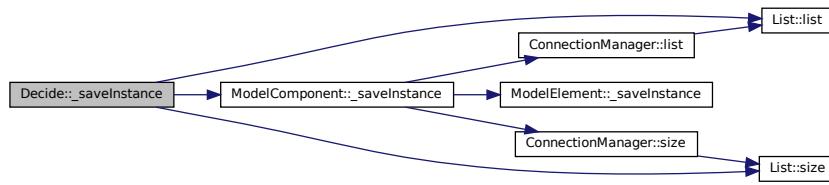
Reimplemented from [ModelComponent](#).

Definition at line 58 of file [Decide.cpp](#).

```
00058                                         {
00059     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00060     //Util::TypeOf<Decide>();
00061     unsigned short i = 0;
00062     fields->emplace("conditions", std::to_string(_conditions->size()));
00063     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->list()->end(); it++) {
00064         fields->emplace("condition" + std::to_string(i++), "\"" + (*it) + "\"");
00065     }
00066 }
```

References [ModelComponent::_saveInstance\(\)](#), [List< T >::list\(\)](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.18.3.6 getConditions() `List< std::string > * Decide::getConditions () const`

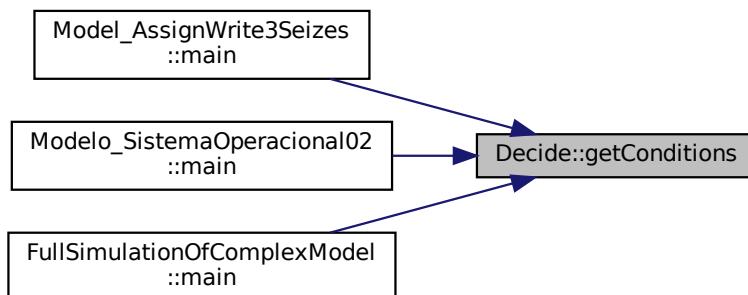
Definition at line 20 of file [Decide.cpp](#).

```

00020
00021     return _conditions;
00022 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.18.3.7 GetPluginInformation() `PluginInformation * Decide::GetPluginInformation () [static]`

Definition at line 78 of file [Decide.cpp](#).

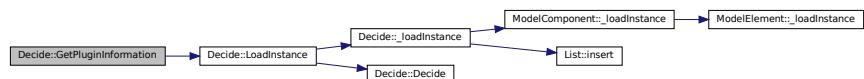
```

00078
00079     PluginInformation* info = new PluginInformation(Util::TypeOf<Decide>(), &Decide::LoadInstance);
00080     return info;
00081 }
```

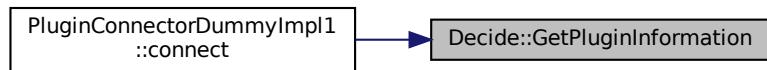
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.3.8 LoadInstance() ModelComponent * Decide::LoadInstance (

```

        Model * model,
        std::map< std::string, std::string > * fields ) [static]
```

Definition at line 83 of file [Decide.cpp](#).

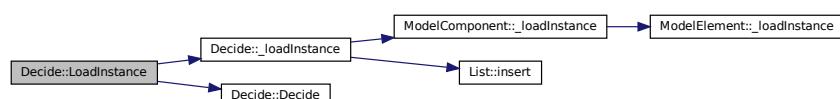
```

00083
00084     Decide* newComponent = new Decide(model);
00085     try {
00086         newComponent->_loadInstance(fields);
00087     } catch (const std::exception& e) {
00088     }
00089 }
00090     return newComponent;
00091 }
```

References [_loadInstance\(\)](#), and [Decide\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.3.9 show() std::string Decide::show () [virtual]

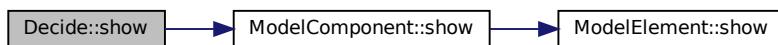
Reimplemented from [ModelComponent](#).

Definition at line 24 of file [Decide.cpp](#).

```
00024     {
00025     return ModelComponent::show() + "";
00026 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



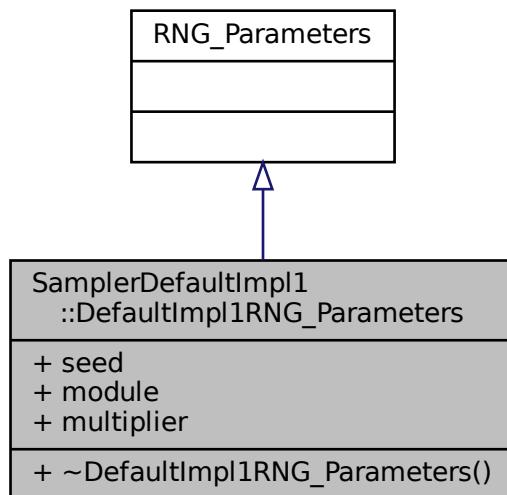
The documentation for this class was generated from the following files:

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

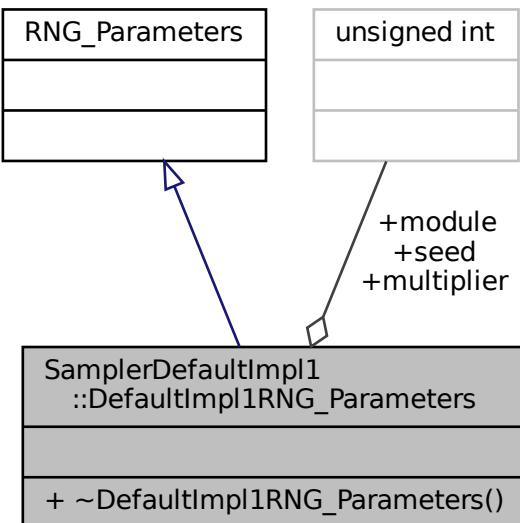
8.19 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference

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

Inheritance diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Collaboration diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Public Member Functions

- `~DefaultImpl1RNG_Parameters ()=default`

Public Attributes

- `unsigned int seed = 666`
- `unsigned int module = 2147483647`
- `unsigned int multiplier = 950706376`

8.19.1 Detailed Description

Definition at line 23 of file [SamplerDefaultImpl1.h](#).

8.19.2 Constructor & Destructor Documentation

8.19.2.1 `~DefaultImpl1RNG_Parameters()` `SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::~DefaultImpl1RNG_Parameters () [default]`

8.19.3 Member Data Documentation

8.19.3.1 module `unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::module = 2147483647`

Definition at line 25 of file [SamplerDefaultImpl1.h](#).

Referenced by [SamplerDefaultImpl1::random\(\)](#).

8.19.3.2 multiplier `unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::multiplier = 950706376`

Definition at line 26 of file [SamplerDefaultImpl1.h](#).

8.19.3.3 seed `unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::seed = 666`

Definition at line 24 of file [SamplerDefaultImpl1.h](#).

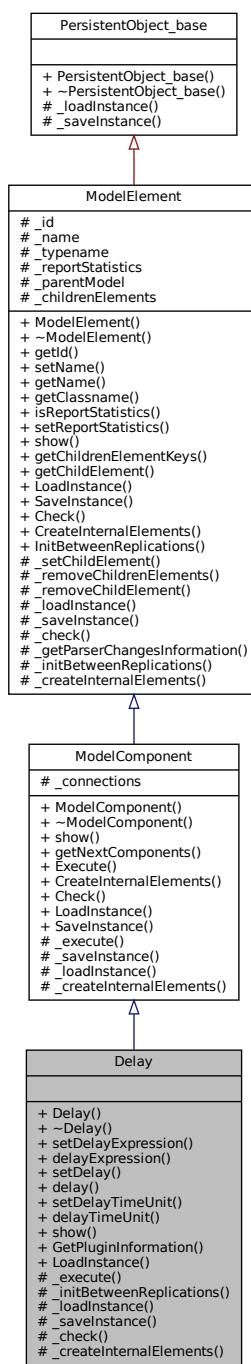
The documentation for this struct was generated from the following file:

- [SamplerDefaultImpl1.h](#)

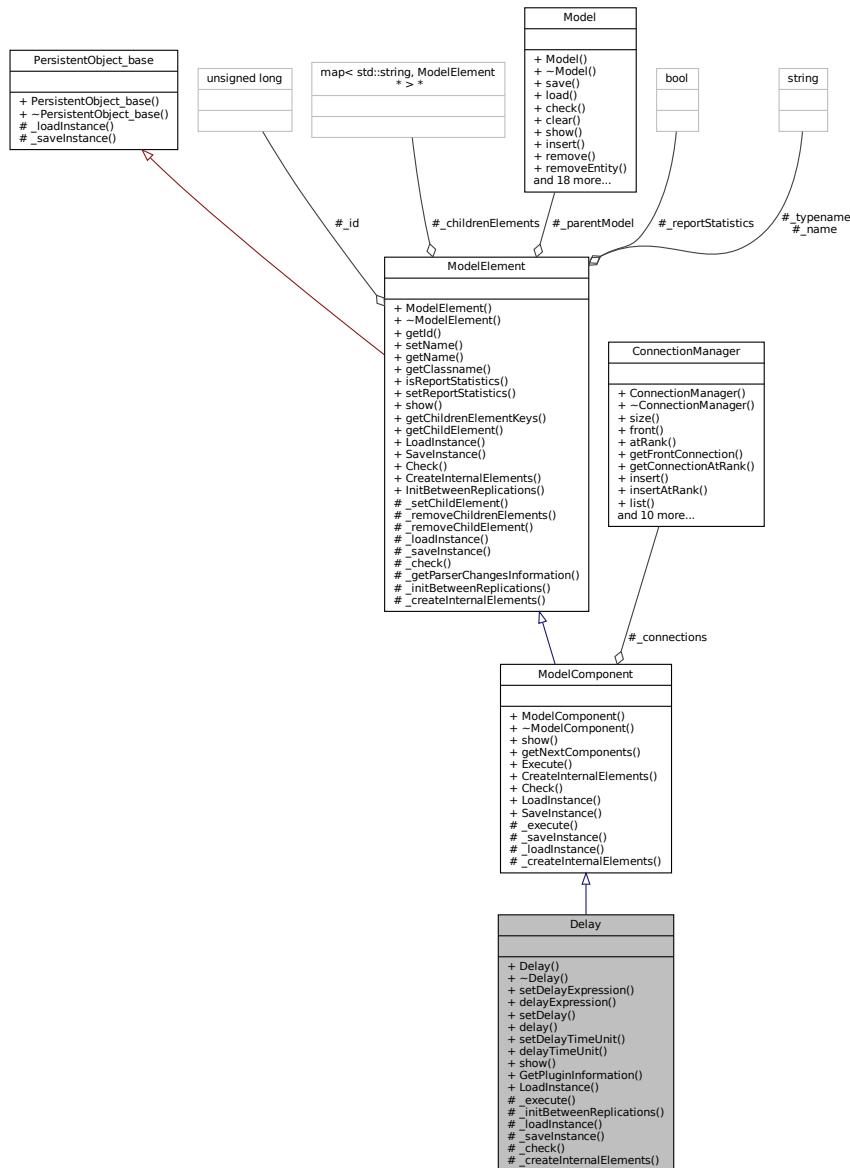
8.20 Delay Class Reference

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

Inheritance diagram for Delay:



Collaboration diagram for Delay:



Public Member Functions

- `Delay (Model *model, std::string name="")`
- virtual `~Delay ()=default`
- void `setDelayExpression (std::string _delayExpression)`
- `std::string delayExpression () const`
- void `setDelay (double delay)`
- `double delay () const`
- void `setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)`
- `Util::TimeUnit delayTimeUnit () const`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.20.1 Detailed Description

Delay module DESCRIPTION The **Delay** module delays an entity by a specified amount of time. When an entity arrives at a **Delay** module, the time delay expression is evaluated and the entity remains in the module for the resulting time period. The time is then allocated to the entity's value-added, non-value added, transfer, wait, or other time. Associated costs are calculated and allocated as well. TYPICAL USES Processing a check at a bank Performing a setup on a machine Transferring a document to another department PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Allocation Type of category to which the entity's incurred delay time and cost will be added. **Delay** Time Determines the value of the delay for the entity. Units Time units used for the delay time.

Definition at line 41 of file `Delay.h`.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 Delay() `Delay::Delay (`

```
00018     Model * model,
00019     std::string name = "" )
```

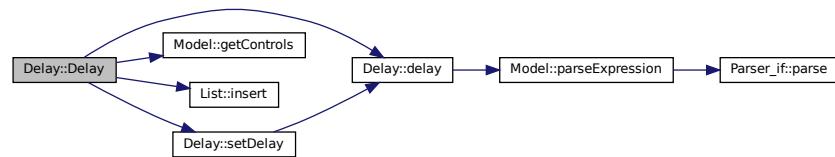
Definition at line 18 of file `Delay.cpp`.

```
00018     : ModelComponent(model, Util::TypeOf<Delay>(), name) {
00019
00020     GetterMember getter = DefineGetterMember<Delay>(this, &Delay::delay);
00021     SetterMember setter = DefineSetterMember<Delay>(this, &Delay::setDelay);
00022     model->getControls()->insert(new SimulationControl(Util::TypeOf<Delay>(), _name + ".Delay",
00023     getter, setter));
00024     //GetterMember getter2 = DefineGetterMember<Delay>(this, &Delay::delayTimeUnit);
00025     //SetterMember setter2 = DefineSetterMember<Delay>(this, &Delay::setDelayTimeUnit);
00026     //model->controls()->insert(new SimulationControl(Util::TypeOf<Delay>(), _name + ".DelayTimeUnit",
00027     getter2, setter2));
00028
00029 }
```

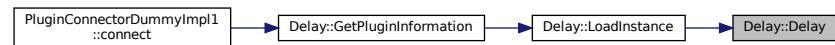
References [ModelElement::_name](#), [delay\(\)](#), [Model::getControls\(\)](#), [List< T >::insert\(\)](#), and [setDelay\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.2 ~Delay() virtual Delay::~Delay () [virtual], [default]

8.20.3 Member Function Documentation

8.20.3.1 _check() bool Delay::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

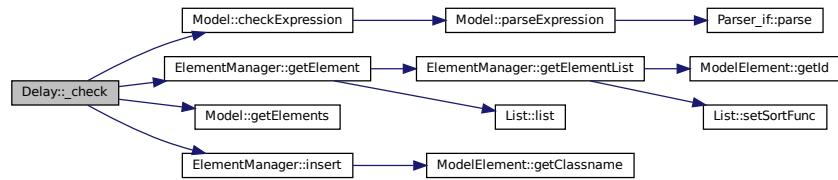
Definition at line 105 of file [Delay.cpp](#).

```

00105
00106     //include attributes needed
00107     ElementManager* elements = _parentModel->getElements();
00108     std::vector<std::string> neededNames = {"Entity.TotalWaitTime"};
00109     std::string neededName;
00110     for (unsigned int i = 0; i < neededNames.size(); i++) {
00111         neededName = neededNames[i];
00112         if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00113             Attribute* attr = new Attribute(_parentModel, neededName);
00114             elements->insert(attr);
00115         }
00116     }
00117     return _parentModel->checkExpression(_delayExpression, "Delay expression", errorMessage);
00118 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), and [ElementManager::insert\(\)](#).

Here is the call graph for this function:



8.20.3.2 `_createInternalElements()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelComponent](#).

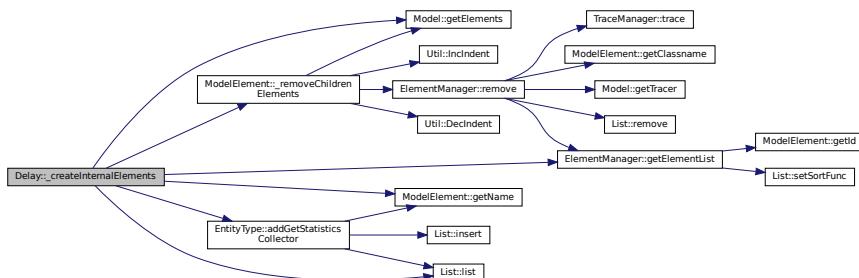
Definition at line 120 of file [Delay.cpp](#).

```

00120
00121     if (_reportStatistics && _cstatWaitTime == nullptr) {
00122         _cstatWaitTime = new StatisticsCollector(_parentModel, _name + "." + "WaitTime", this);
00123         _childrenElements->insert({ "WaitTime", _cstatWaitTime });
00124         // include StatisticsCollector needed in EntityType
00125         ElementManager* elements = _parentModel->getElements();
00126         std::list<ModelElement*>* enttypes =
00127             elements->getElementTypeList(Util::TypeOf<EntityType>())->list();
00128         for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00129             EntityType* enttype = static_cast<EntityType*>((*it));
00130             if ((*it)->isReportStatistics())
00131                 enttype->addGetStatisticsCollector(enttype->getName() + ".WaitTime"); // force create
00132             this CStat before simulation starts
00133         }
00134     } else {
00135         _removeChildrenElements();
00136     }
  
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#), [ModelElement::_reportStatistics](#), [EntityType::addGetStatisticsCollector\(\)](#), [ElementManager::getElementList\(\)](#), [Model::getElements\(\)](#), [ModelElement::getName\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



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

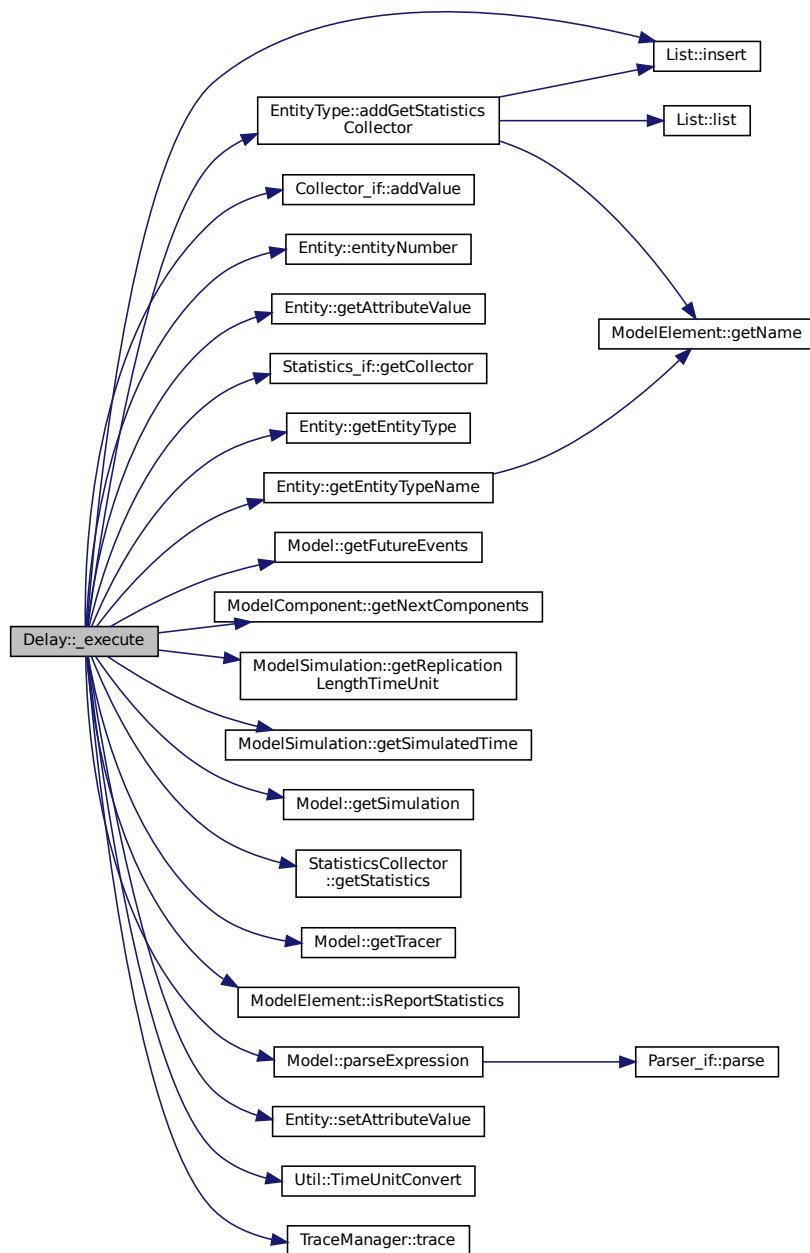
Implements [ModelComponent](#).

Definition at line 72 of file [Delay.cpp](#).

```
00072         {
00073     double waitTime = _parentModel->parseExpression(_delayExpression) *
00074     Util::TimeUnitConvert(_delayTimeUnit, _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00075     if (_reportStatistics) {
00076         _cstatWaitTime->getStatistics()->getCollector()->addValue(waitTime);
00077         if (entity->getEntityType()->isReportStatistics())
00078             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityType() +
00079 ".WaitTime")->getStatistics()->getCollector()->addValue(waitTime);
00080     }
00081     entity->setAttributeValue("Entity.TotalWaitTime",
00082         entity->getAttributeValue("Entity.TotalWaitTime") + waitTime);
00083     double delayEndTime = _parentModel->getSimulation()->getSimulatedTime() + waitTime;
00084     Event* newEvent = new Event(delayEndTime, entity,
00085         this->getNextComponents() ->getFrontConnection());
00086     _parentModel->getFutureEvents() ->insert(newEvent);
00087     _parentModel->getTracer()->trace("End of delay of entity " +
00088         std::to_string(entity->entityNumber()) + " scheduled to time " + std::to_string(delayEndTime));
00089 }
```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [EntityType::addGetStatisticsCollector\(\)](#), [Collector_if::addValue\(\)](#), [Entity::entityNumber\(\)](#), [Entity::getAttributeValue\(\)](#), [Statistics_if::getCollector\(\)](#), [Entity::getEntityType\(\)](#), [Entity::getEntityTypeName\(\)](#), [Model::getFutureEvents\(\)](#), [ModelComponent::getNextComponents\(\)](#), [ModelSimulation::getReplicationLengthTimeUnit\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [Model::getTracer\(\)](#), [List< T >::insert\(\)](#), [ModelElement::isReportStatistics\(\)](#), [Model::parseExpression\(\)](#), [Entity::setAttributeValue\(\)](#), [Util::TimeUnitConvert\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.20.3.4 `_initBetweenReplications()`

`void Delay::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 95 of file [Delay.cpp](#).

```

00095
00096 }
```

```
8.20.3.5 _loadInstance() bool Delay::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

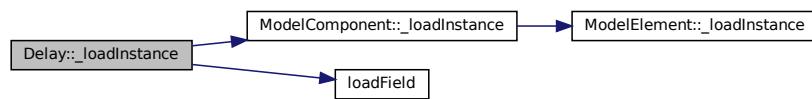
Definition at line 86 of file [Delay.cpp](#).

```
00086
00087     bool res = ModelComponent::_loadInstance(fields);
00088     if (res) {
00089         this->_delayExpression = (*fields->find("delayExpression")).second;
00090         this->_delayTimeUnit = static_cast<Util::TimeUnit>(std::stoi(loadField(fields,
00091             "delayExpressionTimeUnit", std::to_string(static_cast<int>(Util::TimeUnit::second)))));
00092     }
00093     return res;
00093 }
```

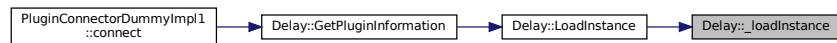
References [ModelComponent::_loadInstance\(\)](#), [loadField\(\)](#), and [Util::second](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

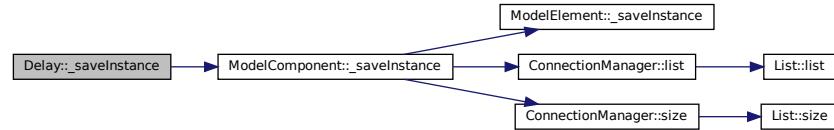
Reimplemented from [ModelComponent](#).

Definition at line 98 of file [Delay.cpp](#).

```
00098
00099     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00100     //Util::TypeOf<Delay>());
00101     fields->emplace("delayExpression", "\"" + this->_delayExpression + "\"");
00102     if (_delayTimeUnit != Util::TimeUnit::second) fields->emplace("delayExpressionTimeUnit",
00103         std::to_string(static_cast<int>(this->_delayTimeUnit)));
00102     return fields;
00103 }
```

References [ModelComponent::_saveInstance\(\)](#), and [Util::second](#).

Here is the call graph for this function:



8.20.3.7 delay() double Delay::delay () const

Definition at line 35 of file [Delay.cpp](#).

```

00035 {
00036     return _parentModel->parseExpression(_delayExpression);
00037 }
  
```

References [ModelElement::_parentModel](#), and [Model::parseExpression\(\)](#).

Referenced by [Delay\(\)](#), and [setDelay\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.3.8 delayExpression() std::string Delay::delayExpression () const

Definition at line 60 of file [Delay.cpp](#).

```

00060 {
00061     return _delayExpression;
00062 }
  
```

8.20.3.9 delayTimeUnit() `Util::TimeUnit Delay::delayTimeUnit () const`Definition at line 68 of file [Delay.cpp](#).

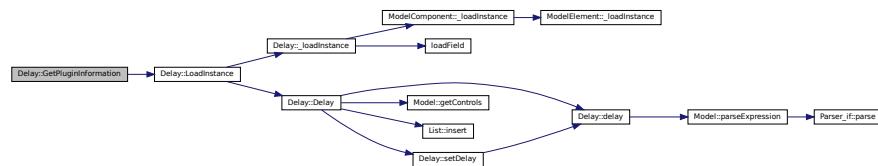
```
00068     {
00069         return _delayTimeUnit;
00070     }
```

8.20.3.10 GetPluginInformation() `PluginInformation * Delay::GetPluginInformation () [static]`Definition at line 138 of file [Delay.cpp](#).

```
00138     {
00139         PluginInformation* info = new PluginInformation(Util::TypeOf<Delay>(), &Delay::LoadInstance);
00140         return info;
00141     }
```

References [LoadInstance\(\)](#).Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.20.3.11 LoadInstance()** `ModelComponent * Delay::LoadInstance (`

```
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

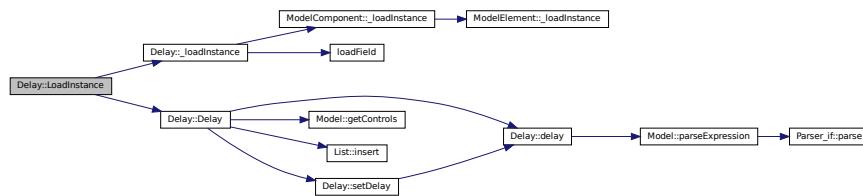
Definition at line 46 of file [Delay.cpp](#).

```
00046     {
00047         Delay* newComponent = new Delay(model);
00048         try {
00049             newComponent->_loadInstance(fields);
00050         } catch (const std::exception& e) {
00051         }
00052         return newComponent;
00053     }
```

References [_loadInstance\(\)](#), and [Delay\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.3.12 setDelay() void Delay::setDelay (double delay)

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

```

00031
00032     _delayExpression = std::to_string(delay);
00033 }
```

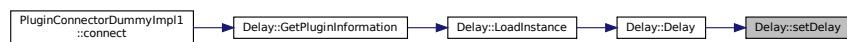
References [delay\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



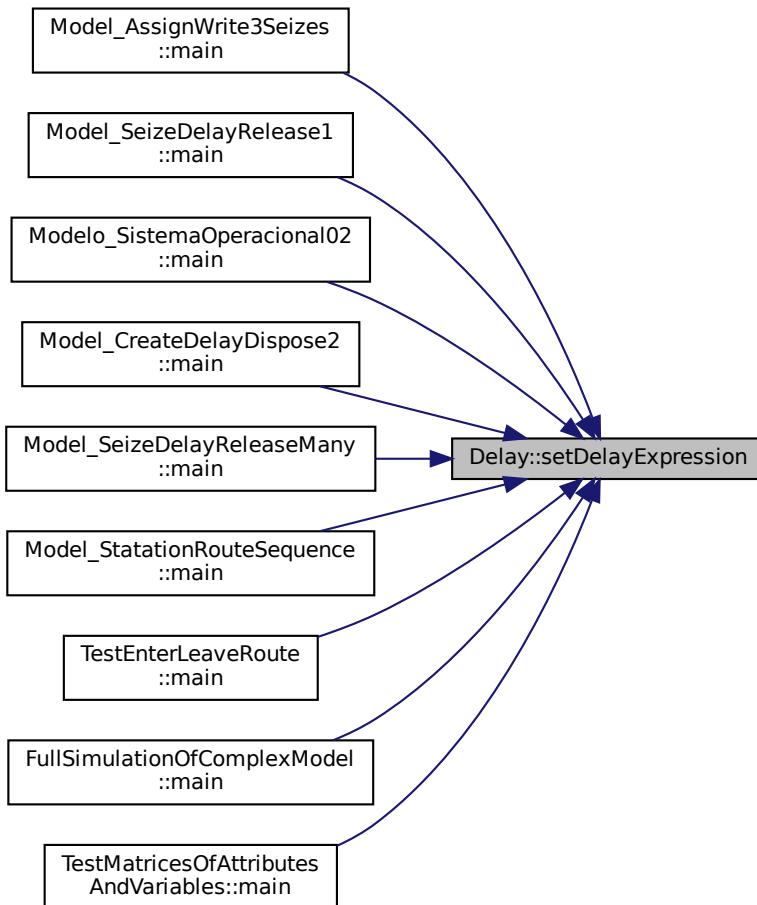
8.20.3.13 setDelayExpression() void Delay::setDelayExpression (std::string _delayExpression)

Definition at line 56 of file [Delay.cpp](#).

```
00056     this->_delayExpression = _delayExpression;
00057 }
00058 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_StationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



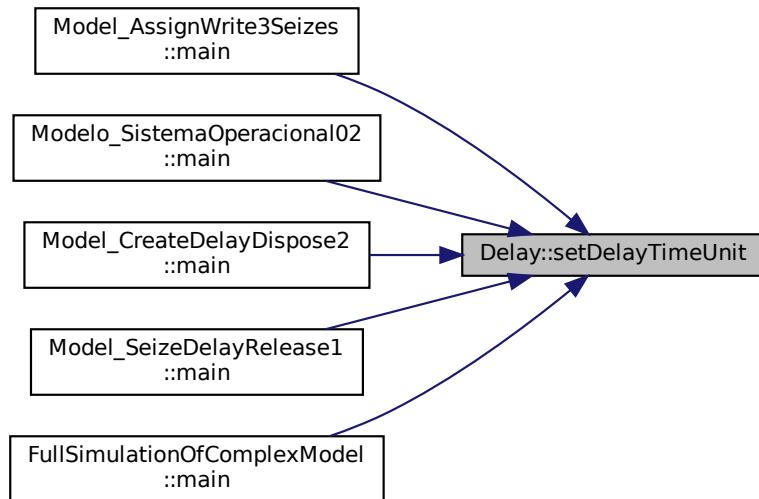
8.20.3.14 setDelayTimeUnit() void Delay::setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)

Definition at line 64 of file [Delay.cpp](#).

```
00064
00065     this->_delayTimeUnit = _delayTimeUnit;
00066 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.20.3.15 show() std::string Delay::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Delay.cpp](#).

```

00040
00041     return ModelComponent::show() +
00042         ",delayExpression=" + this->_delayExpression +
00043         ",timeUnit=" + std::to_string(static_cast<int> (this->_delayTimeUnit));
00044 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



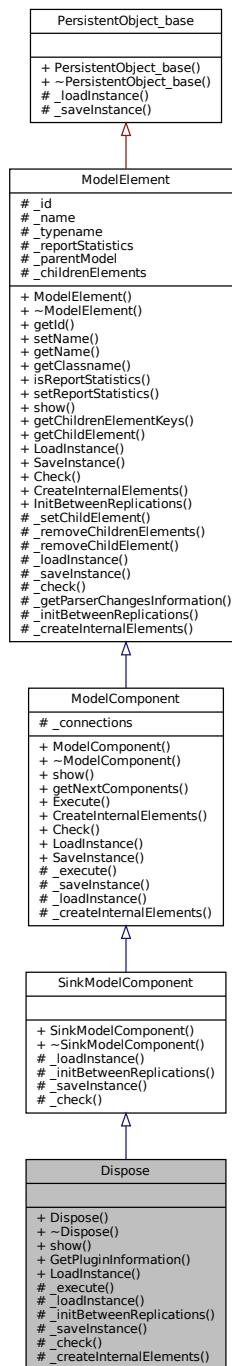
The documentation for this class was generated from the following files:

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

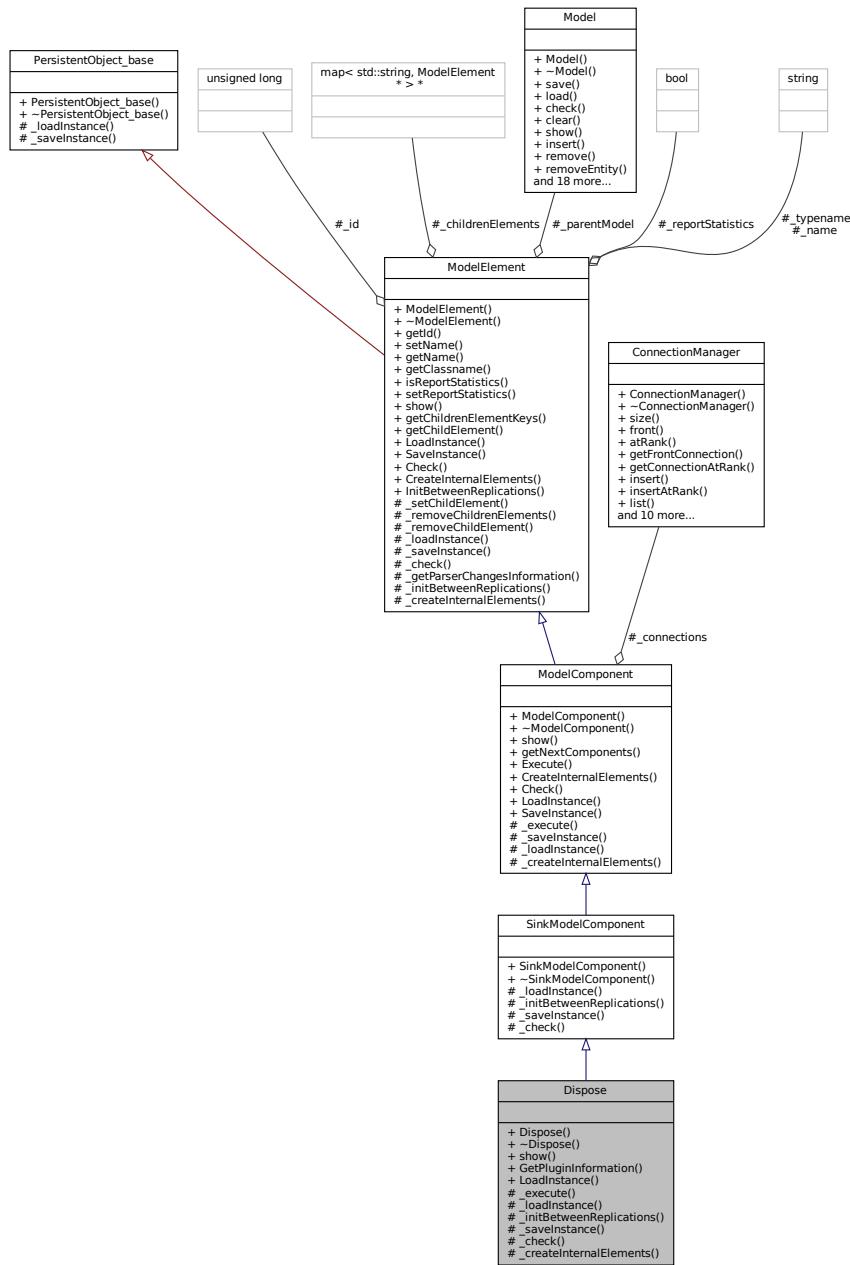
8.21 Dispose Class Reference

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

Inheritance diagram for Dispose:



Collaboration diagram for Dispose:



Public Member Functions

- **Dispose** (**Model** *model, std::string name="")
- virtual **~Dispose** ()=default
- virtual std::string **show** ()

Static Public Member Functions

- static **PluginInformation** * **GetPluginInformation** ()
- static **ModelComponent** * **LoadInstance** (**Model** *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void `_execute` (`Entity` *entity)
- virtual bool `_loadInstance` (`std::map< std::string, std::string >` *fields)
- virtual void `_initBetweenReplications` ()
- virtual `std::map< std::string, std::string >` * `_saveInstance` ()
- virtual bool `_check` (`std::string` *errorMessage)
- virtual void `_createInternalElements` ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.21.1 Detailed Description

Dispose module DESCRIPTION This module is intended as the ending point for entities in a simulation model. Entity statistics may be recorded before the entity is disposed of. TYPICAL USES Parts leaving the modeled facility The termination of a business process Customers departing from the store Prompt Description Name Unique module identifier displayed on the module shape. **Record Entity Statistics** Determines whether or not the incoming entity's statistics will be recorded. Statistics include value-added time, non-value-added time, wait time, transfer time, other time, total time, value-added cost, non-value-added cost, wait cost, transfer cost, other cost, and total cost.

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

8.21.2 Constructor & Destructor Documentation

8.21.2.1 `Dispose()`

```
Dispose::Dispose (
    Model * model,
    std::string name = "" )
```

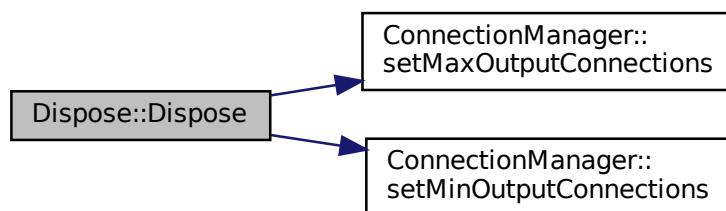
Definition at line 17 of file `Dispose.cpp`.

```
00017     name) {
00018         //_numberOut = new Counter(_parentModel, _name + "." + "Count_number_out", this);
00019         _connections->setMinOutputConnections(0);
00020         _connections->setMaxOutputConnections(0);
00021 }
```

References `ModelComponent::_connections`, `ConnectionManager::setMaxOutputConnections()`, and `ConnectionManager::setMinOutputConnections()`.

Referenced by `LoadInstance()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.21.2.2 ~Dispose() `virtual Dispose::~Dispose () [virtual], [default]`

8.21.3 Member Function Documentation

8.21.3.1 _check() `bool Dispose::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [SinkModelComponent](#).

Definition at line 52 of file [Dispose.cpp](#).

```

00052
00053     // 
00054     return true;
00055 }
```

8.21.3.2 _createInternalElements() `void Dispose::_createInternalElements () [protected], [virtual]`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelComponent](#).

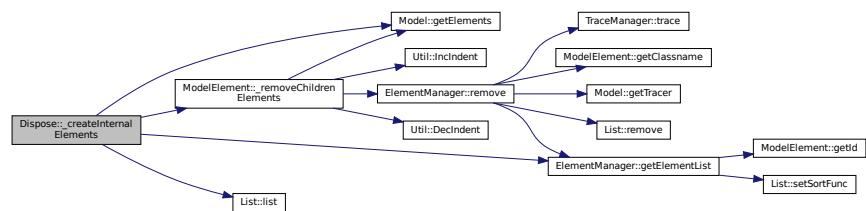
Definition at line 57 of file [Dispose.cpp](#).

```

00057
00058     if (_reportStatistics && _numberOut == nullptr) {
00059         // creates the counter (and then the CStats)
00060         _numberOut = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00061         _childrenElements->insert({"CountNumberIn", _numberOut});
00062         // include StatisticsCollector needed for each EntityType
00063         std::list<ModelElement*>* enttypes =
00064             _parentModel->getElements()->getEntityList(Util::TypeOf<EntityType>())->list();
00065             for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00066                 if ((*it)->isReportStatistics())
00067                     static_cast<EntityType*> ((*it))->addGetStatisticsCollector((*it)->getName() + "." +
00068 "TotalTimeInSystem"); // force create this CStat before model checking
00069             }
00070         } else if (! _reportStatistics && _numberOut != nullptr) {
00071             // _numberOut->~Counter();
00072             //_numberOut = nullptr;
00073         }
00074 }
```

References ModelElement::_childrenElements, ModelElement::_name, ModelElement::_parentModel, ModelElement::_removeChild, ModelElement::_reportStatistics, ElementManager::getElementList(), Model::getElements(), and List<T>::list().

Here is the call graph for this function:



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

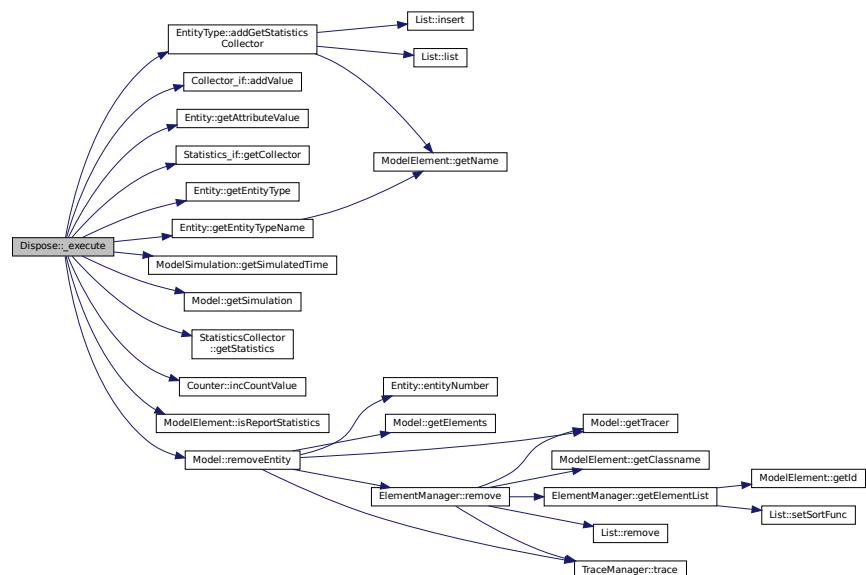
Implements [ModelComponent](#).

Definition at line 27 of file [Dispose.cpp](#).

```
00027                                     {
00028     if (_reportStatistics) {
00029         _numberOut->incCountValue();
00030         if (entity->getEntityType()->isReportStatistics()) {
00031             double timeInSystem = _parentModel->getSimulation()->getSimulatedTime() -
00032             entity->getAttributeValue("Entity.ArrivalTime");
00033             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityTypeName() + "." +
00034             "TotalTimeInSystem")->getStatistics()->getCollector()->addValue(timeInSystem);
00035         }
00036     }
00037     _parentModel->removeEntity(entity, _reportStatistics);
00038 }
```

References ModelElement::_parentModel, ModelElement::_reportStatistics, EntityType::addGetStatisticsCollector(), Collector_if::addValue(), Entity::getAttributeValue(), Statistics_if::getCollector(), Entity::getEntityType(), Entity::getEntityTypeName(), ModelSimulation::getSimulatedTime(), Model::getSimulation(), StatisticsCollector::getStatistics(), Counter::incCountValue(), ModelElement::isReportStatistics(), and Model::removeEntity().

Here is the call graph for this function:



8.21.3.4 `_initBetweenReplications()` void Dispose::`_initBetweenReplications ()` [protected], [virtual]

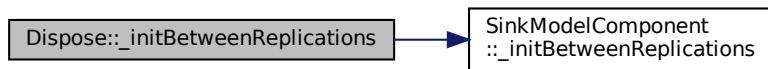
Reimplemented from [SinkModelComponent](#).

Definition at line 42 of file [Dispose.cpp](#).

```
00042 {  
00043     SinkModelComponent::_initBetweenReplications ();  
00044 }
```

References [SinkModelComponent::`_initBetweenReplications\(\)`](#).

Here is the call graph for this function:



8.21.3.5 `_loadInstance()` bool Dispose::`_loadInstance (`

`std::map< std::string, std::string > * fields)` [protected], [virtual]

Reimplemented from [SinkModelComponent](#).

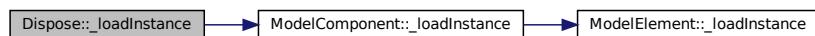
Definition at line 38 of file [Dispose.cpp](#).

```
00038 {  
00039     return ModelComponent::_loadInstance (fields);  
00040 }
```

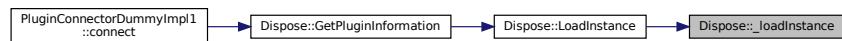
References [ModelComponent::`_loadInstance\(\)`](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

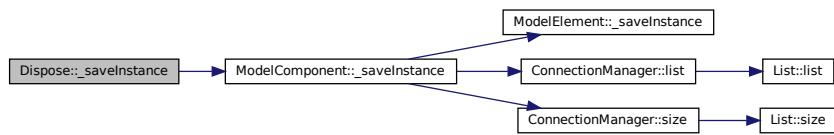
Reimplemented from [SinkModelComponent](#).

Definition at line 46 of file [Dispose.cpp](#).

```
00046     {
00047         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00048         return fields;
00049     }
00050 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.21.3.7 `GetPluginInformation()` `PluginInformation * Dispose::GetPluginInformation ()` [static]

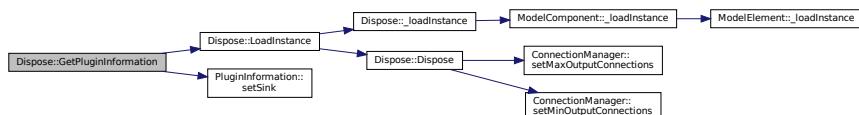
Definition at line 76 of file [Dispose.cpp](#).

```
00076     {
00077         PluginInformation* info = new PluginInformation(Util::TypeOf<Dispose>(), &Dispose::LoadInstance);
00078         info->setSink(true);
00079         return info;
00080     }
```

References [LoadInstance\(\)](#), and [PluginInformation::setSink\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.21.3.8 LoadInstance() ModelComponent * Dispose::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

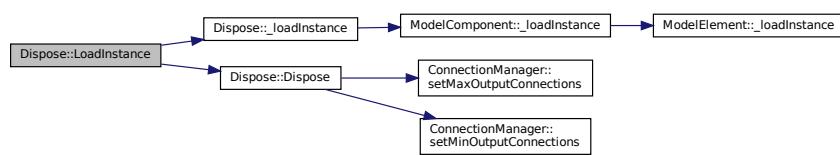
Definition at line 82 of file [Dispose.cpp](#).

```
00082
00083     Dispose* newComponent = new Dispose(model);
00084     try {
00085         newComponent->_loadInstance(fields);
00086     } catch (const std::exception& e) {
00087     }
00088 }
00089     return newComponent;
00090 }
```

References [_loadInstance\(\)](#), and [Dispose\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

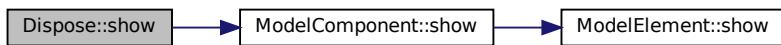
Reimplemented from [ModelComponent](#).

Definition at line 23 of file [Dispose.cpp](#).

```
00023
00024     {
00025     return SinkModelComponent::show();
  }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



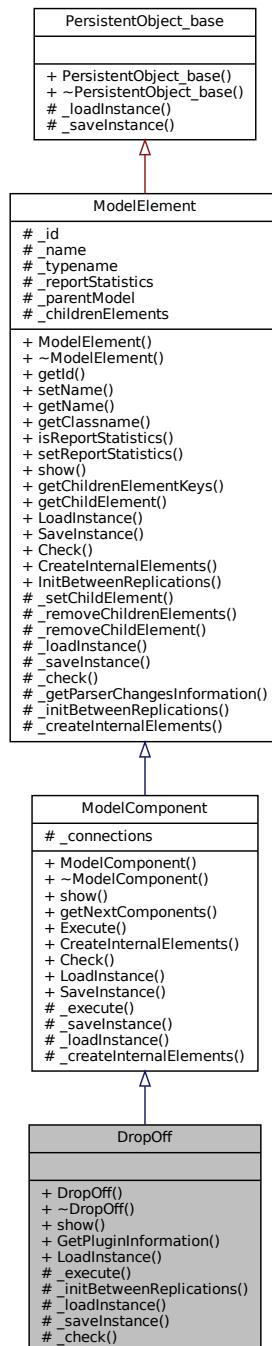
The documentation for this class was generated from the following files:

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

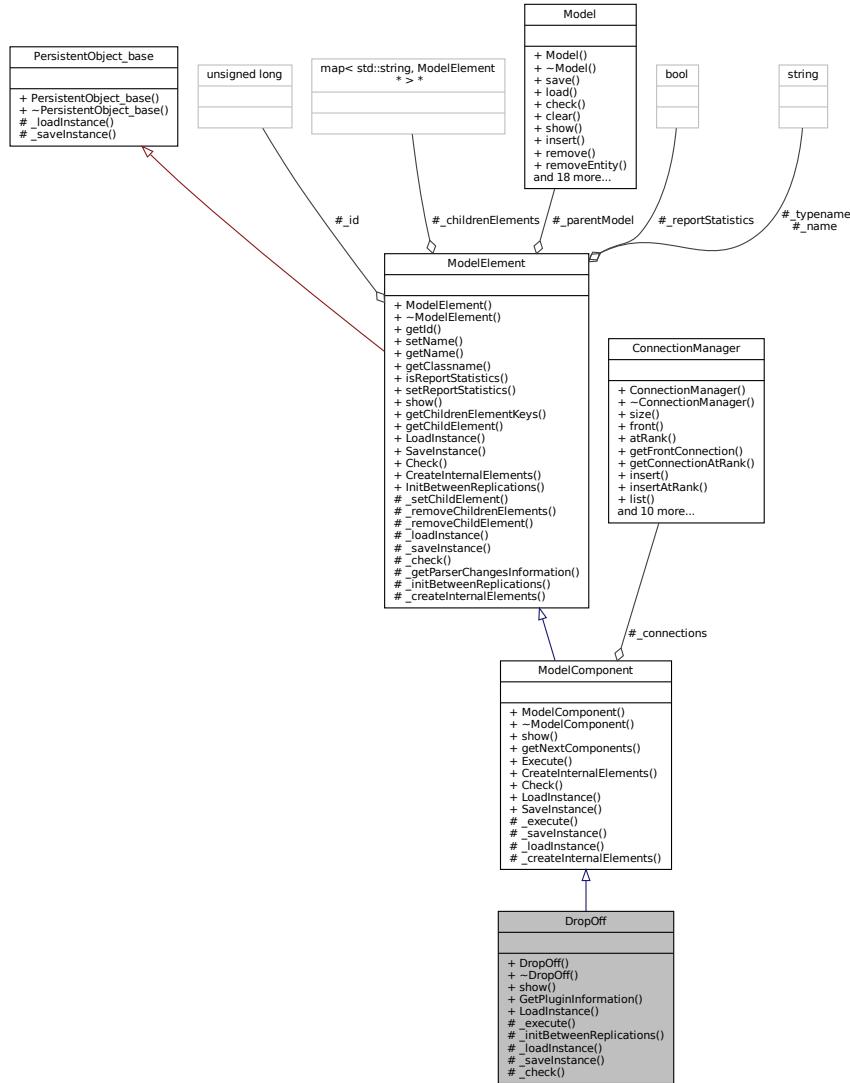
8.22 DropOff Class Reference

```
#include <DropOff.h>
```

Inheritance diagram for DropOff:



Collaboration diagram for DropOff:



Public Member Functions

- `DropOff (Model *model, std::string name="")`
- virtual `~DropOff ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.22.1 Detailed Description

Dropoff module DESCRIPTION The Dropoff module removes a specified number of entities from the entity's group and sends them to another module, as specified by a graphical connection. Group user-defined attribute value and internal attributes may be given to the dropped-off entities based on a specified rule. TYPICAL USES Loading shelves with product Separating a form for use in various departments PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Quantity Number of entities that will be dropped off from an incoming representative grouped entity. Starting Rank Starting rank of the entities to be dropped off, based on the entities in the group. Member Attributes Method of determining how to assign the representative entity attribute values (other than costs/times) to the dropped-off original entities. [Attribute Name](#) Name of representative entity attribute(s) assigned to droppedoff original entities of the group

Definition at line 41 of file [DropOff.h](#).

8.22.2 Constructor & Destructor Documentation

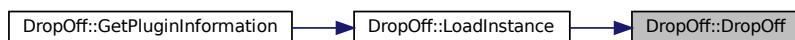
8.22.2.1 DropOff() `DropOff::DropOff (`
 `Model * model,`
 `std::string name = "")`

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

```
00017 : ModelComponent(model, Util::TypeOf<DropOff>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.22.2.2 ~DropOff() `virtual DropOff::~DropOff () [virtual], [default]`

8.22.3 Member Function Documentation

```
8.22.3.1 _check() bool DropOff::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 56 of file [DropOff.cpp](#).

```
00056
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
```

```
8.22.3.2 _execute() void DropOff::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 34 of file [DropOff.cpp](#).

```
00034
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



```
8.22.3.3 _initBetweenReplications() void DropOff::_initBetweenReplications ( ) [protected],
[virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 47 of file [DropOff.cpp](#).

```
00047
00048 }
```

```
8.22.3.4 _loadInstance() bool DropOff::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

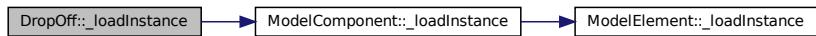
Definition at line 39 of file [DropOff.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

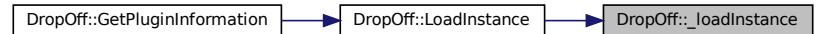
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.22.3.5 _saveInstance() std::map< std::string, std::string > * DropOff::_saveInstance ( )
[protected], [virtual]
```

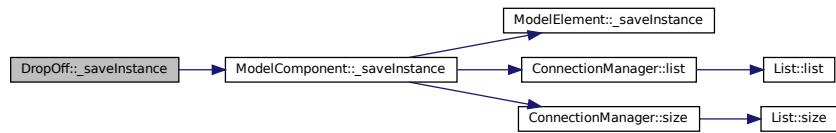
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [DropOff.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



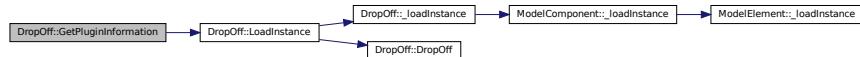
8.22.3.6 GetPluginInformation() `PluginInformation * DropOff::GetPluginInformation () [static]`

Definition at line 62 of file `DropOff.cpp`.

```
00062 {  
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<DropOff>(), &DropOff::LoadInstance);  
00064 // ...  
00065     return info;  
00066 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.22.3.7 LoadInstance() `ModelComponent * DropOff::LoadInstance (`

```
Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

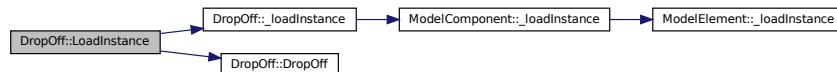
Definition at line 24 of file `DropOff.cpp`.

```
00024 {  
00025     DropOff* newComponent = new DropOff(model);  
00026     try {  
00027         newComponent->_loadInstance(fields);  
00028     } catch (const std::exception& e) {  
00029     }  
00031     return newComponent;  
00032 }
```

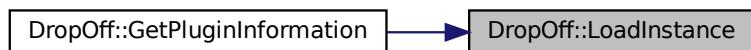
References `_loadInstance()`, and `DropOff()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.3.8 show() std::string DropOff::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [DropOff.cpp](#).

```
00020           {  
00021     return ModelComponent::show() + "";  
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



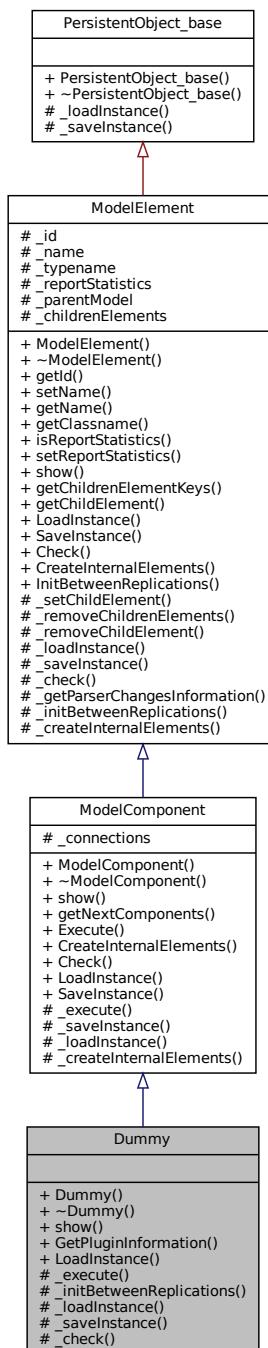
The documentation for this class was generated from the following files:

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

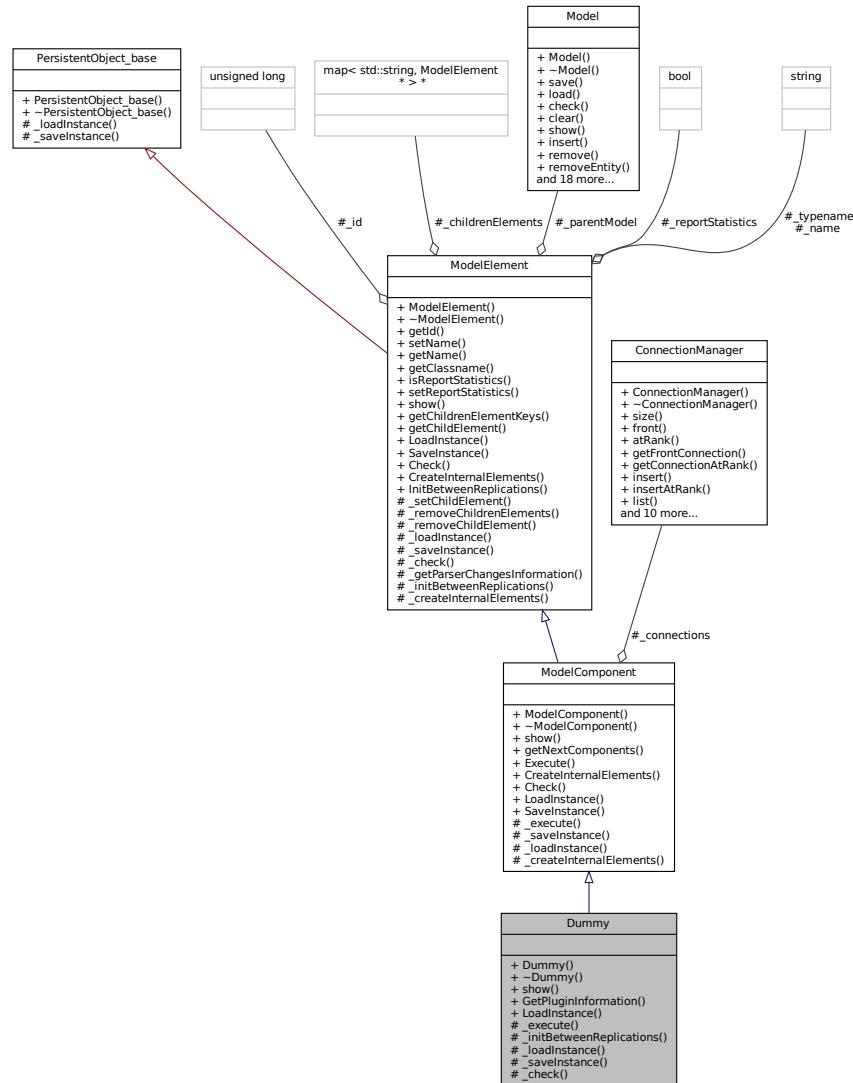
8.23 Dummy Class Reference

```
#include <Dummy.h>
```

Inheritance diagram for Dummy:



Collaboration diagram for Dummy:



Public Member Functions

- **Dummy (Model *model, std::string name="")**
- virtual **~Dummy ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.23.1 Detailed Description

This component ...

Definition at line 22 of file [Dummy.h](#).

8.23.2 Constructor & Destructor Documentation

8.23.2.1 Dummy() [Dummy::Dummy](#) (

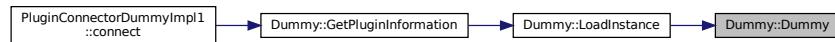
```
    Model * model,
    std::string name = "" )
```

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

```
00017 : ModelComponent(model, Util::TypeOf<Dummy>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.23.2.2 ~Dummy() [virtual Dummy::~Dummy](#) () [virtual], [default]

8.23.3 Member Function Documentation

8.23.3.1 _check() [bool Dummy::_check](#) (

```
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Dummy.cpp](#).

```
00056 {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
```

```
8.23.3.2 _execute() void Dummy::_execute (
    Entity * entity) [protected], [virtual]
```

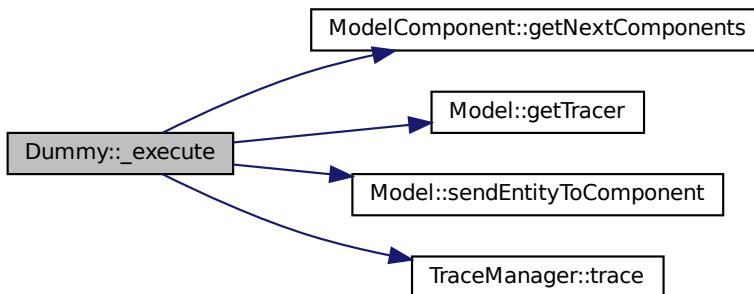
Implements [ModelComponent](#).

Definition at line 34 of file [Dummy.cpp](#).

```
00034     {
00035         _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036         this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00037         0.0);
00037 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



```
8.23.3.3 _initBetweenReplications() void Dummy::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Dummy.cpp](#).

```
00047
00048 }
```

```
8.23.3.4 _loadInstance() bool Dummy::_loadInstance (
```

```
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

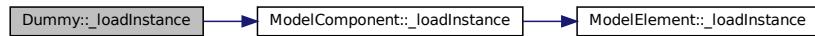
Definition at line 39 of file [Dummy.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

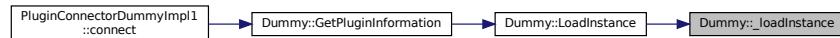
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.5 [_saveInstance\(\)](#) std::map< std::string, std::string > * Dummy::_saveInstance () [protected], [virtual]

Reimplemented from [ModelComponent](#).

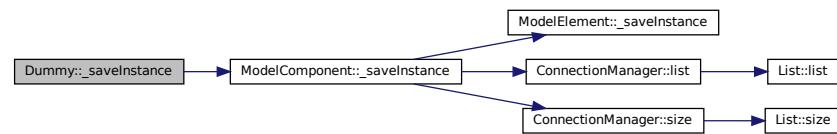
Definition at line 50 of file [Dummy.cpp](#).

```

00050
00051     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);
00052     //...
00053     return fields;
00054 }
  
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.23.3.6 GetPluginInformation() `PluginInformation * Dummy::GetPluginInformation () [static]`

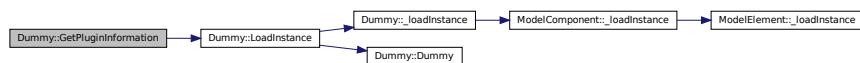
Definition at line 62 of file [Dummy.cpp](#).

```
00062 {  
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Dummy>(), &Dummy::LoadInstance);  
00064     // ...  
00065     return info;  
00066 }
```

References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.7 LoadInstance() `ModelComponent * Dummy::LoadInstance (`

```
    Model * model,  
    std::map< std::string, std::string > * fields ) [static]
```

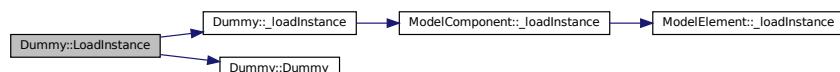
Definition at line 24 of file [Dummy.cpp](#).

```
00024 {  
00025     Dummy* newComponent = new Dummy(model);  
00026     try {  
00027         newComponent->_loadInstance(fields);  
00028     } catch (const std::exception& e) {  
00029     }  
00030     return newComponent;  
00031 }  
00032 }
```

References [_loadInstance\(\)](#), and [Dummy\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.8 `show()` `std::string Dummy::show() [virtual]`

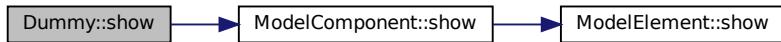
Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Dummy.cpp](#).

```
00020     {
00021     return ModelComponent::show() + "";
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



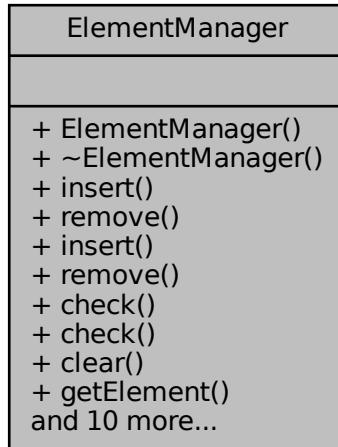
The documentation for this class was generated from the following files:

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

8.24 ElementManager Class Reference

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

Collaboration diagram for ElementManager:



Public Member Functions

- `ElementManager (Model *model)`
- virtual `~ElementManager ()=default`
- `bool insert (ModelElement *anElement)`
- `void remove (ModelElement *anElement)`

Deprecated.
- `bool insert (std::string elementTypename, ModelElement *anElement)`

Deprecated.
- `void remove (std::string elementTypename, ModelElement *anElement)`
- `bool check (std::string elementTypename, ModelElement *anElement, std::string expressionName, std::string *errorMessage)`
- `bool check (std::string elementTypename, std::string elementName, std::string expressionName, bool mandatory, std::string *errorMessage)`
- `void clear ()`
- `ModelElement * getElement (std::string elementTypename, Util::identification id)`
- `ModelElement * getElement (std::string elementTypename, std::string name)`
- `unsigned int getNumberOfElements (std::string elementTypename)`
- `unsigned int getNumberOfElements ()`
- `int getRankOf (std::string elementTypename, std::string name)`

returns the position (1st position=0) of the element if found, or negative value if not found
- `std::list< std::string > * getElementClassnames () const`
- `List< ModelElement * > * getElementList (std::string elementTypename) const`
- `void show ()`
- `Model * getParentModel () const`
- `bool hasChanged () const`
- `void setHasChanged (bool _hasChanged)`

8.24.1 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 allowed.

Definition at line 30 of file [ElementManager.h](#).

8.24.2 Constructor & Destructor Documentation

8.24.2.1 ElementManager()

```
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.

Definition at line 19 of file [ElementManager.cpp](#).

```
00019     _parentModel = model;
00020     _elements = new std::map<std::string, List<ModelElement*>>();
00021     /* \todo: -- Sort methods for elements should be a decorator */
00022     //_elements->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00023     //    return a->getId() < b->getId();
00024     //});
00025 };
00026
00027 }
```

8.24.2.2 ~ElementManager()

```
virtual ElementManager::~ElementManager ( ) [virtual], [default]
```

8.24.3 Member Function Documentation

8.24.3.1 check() [1/2]

```
bool ElementManager::check (
    std::string elementTypename,
    ModelElement * anElement,
    std::string expressionName,
    std::string * errorMessage )
```

Definition at line 77 of file [ElementManager.cpp](#).

```
00077     {
00078     bool result = anElement != nullptr;
00079     if (!result) {
00080         std::string msg = elementTypename + " for '" + expressionName + "' is null.";
00081         errorMessage->append(msg);
00082     } else {
00083         result = check(elementTypename, anElement->getName(), expressionName, true, errorMessage);
00084     }
00085     return result;
00086 }
```

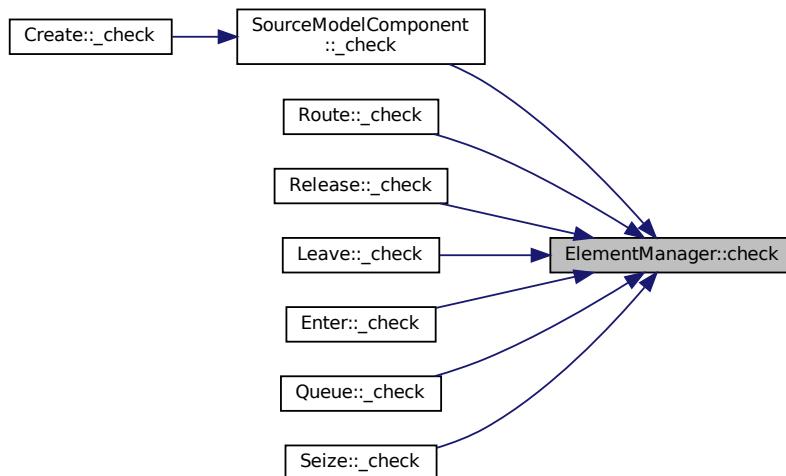
References [ModelElement::getName\(\)](#).

Referenced by [SourceModelComponent::_check\(\)](#), [Route::_check\(\)](#), [Release::_check\(\)](#), [Leave::_check\(\)](#), [Enter::_check\(\)](#), [Queue::_check\(\)](#), and [Seize::_check\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.2 check() [2/2] `bool ElementManager::check (`
 `std::string elementTypename,`
 `std::string elementName,`
 `std::string expressionName,`
 `bool mandatory,`
 `std::string * errorMessage)`

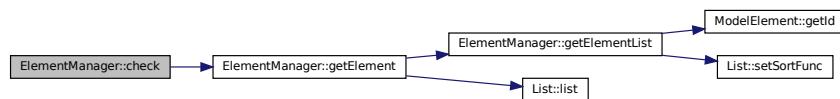
Definition at line 65 of file [ElementManager.cpp](#).

```

00065
00066     if (elementName == "" && !mandatory) {
00067         return true;
00068     }
00069     bool result = getElement(elementTypename, elementName) != nullptr;
00070     if (!result) {
00071         std::string msg = elementTypename + " \" " + elementName + " \" for '\" " + expressionName + "' is
not in the model.'";
00072         errorMessage->append(msg);
00073     }
00074     return result;
00075 }
```

References [getElement\(\)](#).

Here is the call graph for this function:



8.24.3.3 clear() void ElementManager::clear ()

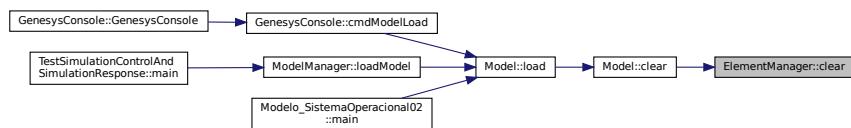
Definition at line 88 of file [ElementManager.cpp](#).

```

00088     {
00089         this->_elements->clear();
00090     }
  
```

Referenced by [Model::clear\(\)](#).

Here is the caller graph for this function:



8.24.3.4 getElement() [1/2] ModelElement * ElementManager::getElement (

```

        std::string elementTypename,
        std::string name )
  
```

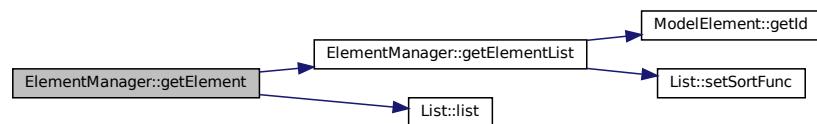
Definition at line 186 of file [ElementManager.cpp](#).

```

00186
00187     List<ModelElement*>* list = getElementList(elementTypename);
00188     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00189         it++) {
00190         if ((*it)->getName() == name) { // found
00191             return (*it);
00192         }
00193     }
00194 }
  
```

References [getElementList\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.24.3.5 getElement() [2/2] `ModelElement * ElementManager::getElement (std::string elementTypename, Util::identification id)`

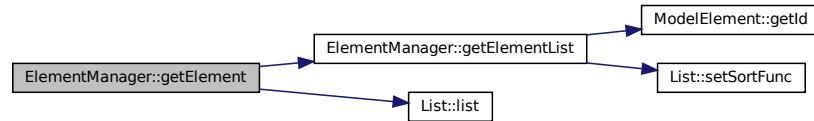
Definition at line 155 of file `ElementManager.cpp`.

```
00155
00156     List<ModelElement*>* list = getElementList(elementTypename);
00157     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00158         it++) {
00159         if ((*it)->getId() == id) { // found
00160             return (*it);
00161         }
00162     }
00163 }
```

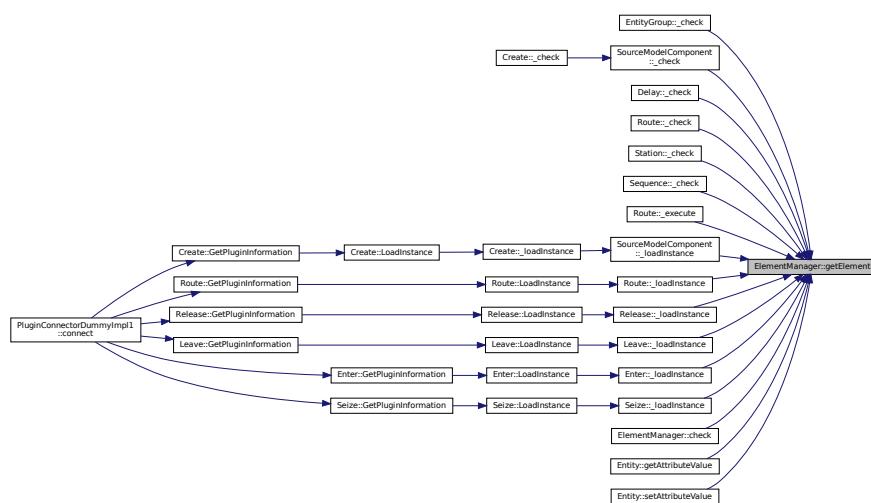
References `getElementList()`, and `List< T >::list()`.

Referenced by `EntityGroup::_check()`, `SourceModelComponent::_check()`, `Delay::_check()`, `Route::_check()`, `Station::_check()`, `Sequence::_check()`, `Route::_execute()`, `SourceModelComponent::_loadInstance()`, `Route::_loadInstance()`, `Release::_loadInstance()`, `Leave::_loadInstance()`, `Enter::_loadInstance()`, `Seize::_loadInstance()`, `check()`, `Entity::getAttributeValue()`, and `Entity::setAttributeValue()`.

Here is the call graph for this function:



Here is the caller graph for this function:



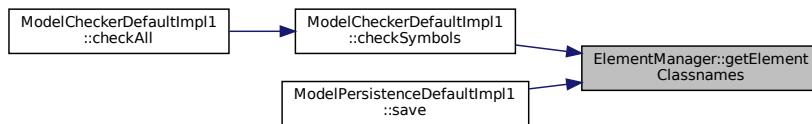
8.24.3.6 getElementExceptions() `std::list< std::string > * ElementManager::getElementExceptions() const`

Definition at line 178 of file [ElementManager.cpp](#).

```
00178     std::list<std::string>* keys = new std::list<std::string>();
00179     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin(); it != _elements->end(); it++) {
00180         keys->insert(keys->end(), (*it).first);
00181     }
00182     return keys;
00183 }
```

Referenced by [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), and [ModelPersistenceDefaultImpl1::save\(\)](#).

Here is the caller graph for this function:



8.24.3.7 getElementList() `List< ModelElement * > * ElementManager::getElementList(std::string elementTypename) const`

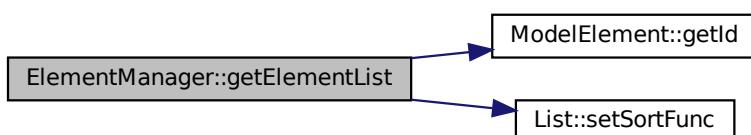
Definition at line 140 of file [ElementManager.cpp](#).

```
00140
00141     std::map<std::string, List<ModelElement*>>::iterator it = this->_elements->find(elementTypename);
00142     if (it == this->_elements->end()) {
00143         // list does not exists yet. Create it and set a valid iterator
00144         List<ModelElement*>* newList = new List<ModelElement*>();
00145         newList->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00146             return a->getId() < b->getId();
00147         });
00148         _elements->insert(std::pair<std::string, List<ModelElement*>>(elementTypename, newList));
00149         it = this->_elements->find(elementTypename);
00150     }
00151     List<ModelElement*>* infras = it->second;
00152     return infras;
00153 }
```

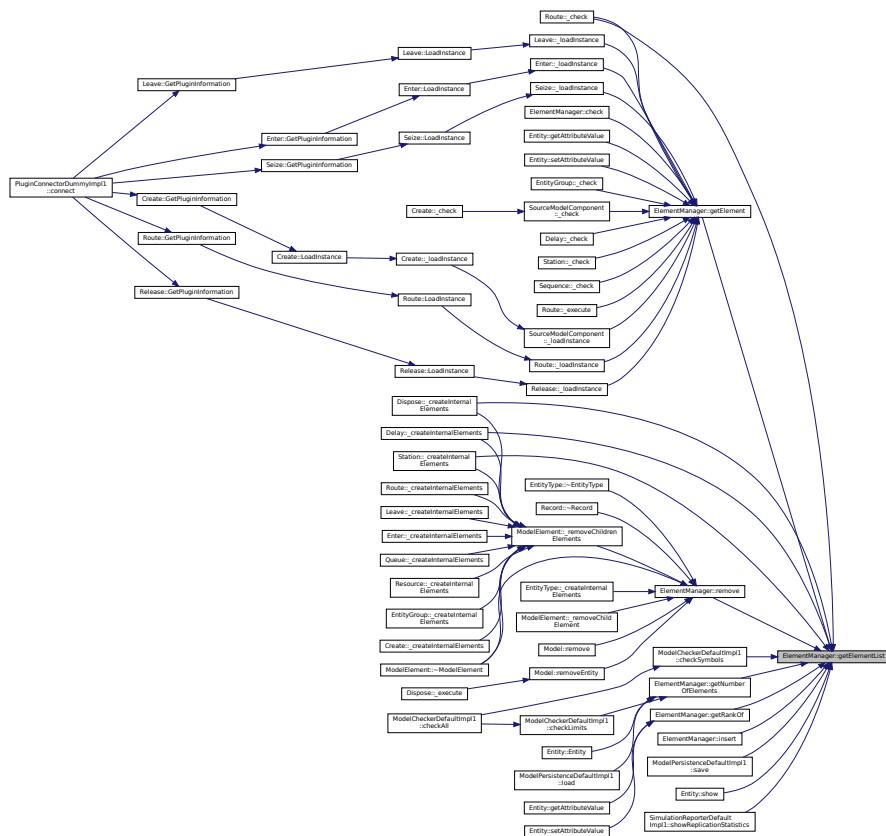
References [ModelElement::getId\(\)](#), and [List< T >::setSortFunc\(\)](#).

Referenced by [Route::_check\(\)](#), [Dispose::_createInternalElements\(\)](#), [Delay::_createInternalElements\(\)](#), [Station::_createInternalElements\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [getElement\(\)](#), [getNumberOfElements\(\)](#), [getRankOf\(\)](#), [insert\(\)](#), [remove\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [Entity::show\(\)](#), and [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.8 getNumberOfElements() [1/2] `unsigned int ElementManager::getNumberOfElements ()`

Definition at line 97 of file [ElementManager.cpp](#).

```
00097
00098     unsigned int total = 0;
00099     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin(); it != _elements->end(); it++) {
00100         total += (*it).second->size();
00101     }
00102     return total;
00103 }
```

8.24.3.9 getNumberOfElements() [2/2] `unsigned int ElementManager::getNumberOfElements (std::string elementTypename)`

Definition at line 92 of file [ElementManager.cpp](#).

```
00092
00093     List<ModelElement*>* listElements = getElementList(elementTypename);
00094     return listElements->size();
00095 }
```

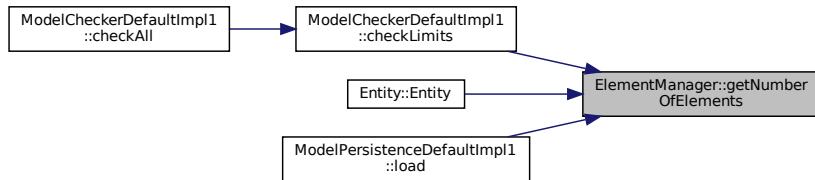
References [getElementList\(\)](#), and [List< T >::size\(\)](#).

Referenced by [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [Entity::Entity\(\)](#), and [ModelPersistenceDefaultImpl1::load\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.10 `getParentModel()` `Model * ElementManager::getParentModel () const`

Definition at line 128 of file [ElementManager.cpp](#).

```

00128
00129     return _parentModel;
00130 }
```

8.24.3.11 `getRankOf()` `int ElementManager::getRankOf (` `std::string elementTypename,` `std::string name)`

returns the position (1st position=0) of the element if found, or negative value if not found

Definition at line 165 of file [ElementManager.cpp](#).

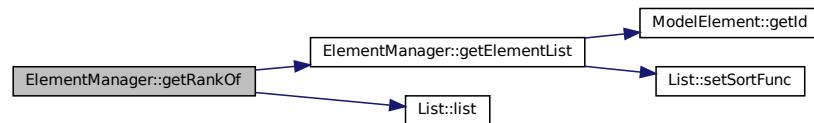
```

00165
00166     int rank = 0;
00167     List<ModelElement*>* list = getElementList(elementTypename);
00168     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00169         it++) {
00170         if ((*it)->getName() == name) { // found
00171             return rank;
00172         } else {
00173             rank++;
00174         }
00175     }
00176 }
```

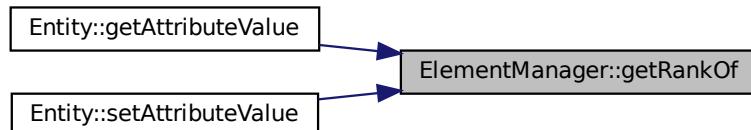
References [getElementList\(\)](#), and [List< T >::list\(\)](#).

Referenced by [Entity::getAttributeValue\(\)](#), and [Entity::setAttributeValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.12 hasChanged() bool ElementManager::hasChanged () const

Definition at line 132 of file [ElementManager.cpp](#).

```
00132 {  
00133     return _hasChanged;  
00134 }
```

Referenced by [Model::hasChanged\(\)](#).

Here is the caller graph for this function:



```
8.24.3.13 insert() [1/2] bool ElementManager::insert (
    ModelElement * anElement )
```

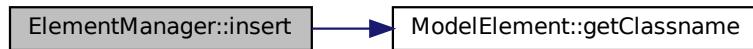
Definition at line 29 of file [ElementManager.cpp](#).

```
00029     {
00030         std::string elementTypename = anElement->getclassname();
00031         bool res = insert(elementTypename, anElement);
00032         /*
00033             if (res)
00034                 _parentModel->tracer()->trace(Util::TraceLevel::elementResult, "Element " + anElement->name()
00035             + " successfully inserted");
00036             else
00037                 _parentModel->tracer()->trace(Util::TraceLevel::elementResult, "Element " + anElement->name()
00038             + " could not be inserted");
00039         */
00039     return res;
00039 }
```

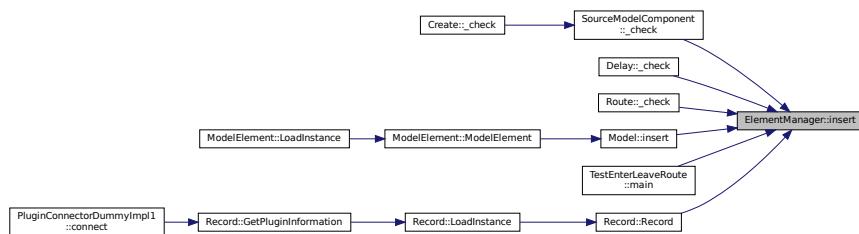
References [ModelElement::getclassname\(\)](#).

Referenced by [SourceModelComponent::_check\(\)](#), [Delay::_check\(\)](#), [Route::_check\(\)](#), [Model::insert\(\)](#), [TestEnterLeaveRoute::main\(\)](#), and [Record::Record\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.24.3.14 insert() [2/2] bool ElementManager::insert (
    std::string elementTypename,
    ModelElement * anElement )
```

Deprecated.

Definition at line 41 of file [ElementManager.cpp](#).

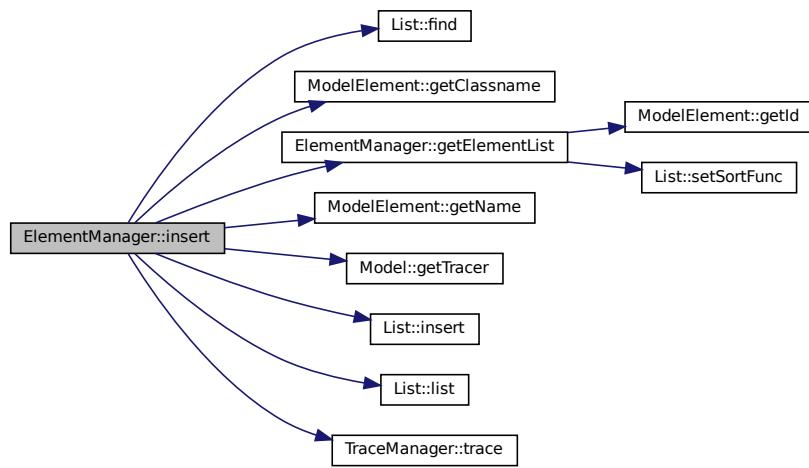
```
00041     {
00042         List<ModelElement*>* listElements = getElementList(elementTypename);
00043         if (listElements->find(anElement) == listElements->list()->end()) { //not found
00044             listElements->insert(anElement);
```

```

00045     this->_parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Element " +
00046     anElement->getclassname() + " \" " + anElement->getName() + "\" successfully inserted.");
00047     return true;
00048 }
00049 this->_parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Element \" " +
00050     anElement->getName() + "\" could not be inserted.");
00051     return false;
00052 }
```

References [List< T >::find\(\)](#), [ModelElement::getclassname\(\)](#), [getElementList\(\)](#), [ModelElement::getName\(\)](#), [Model::getTracer\(\)](#), [List< T >::insert\(\)](#), [List< T >::list\(\)](#), [Util::toolDetailed](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.24.3.15 remove() [1/2] void ElementManager::remove (
 ModelElement * anElement)

Deprecated.

Definition at line 52 of file [ElementManager.cpp](#).

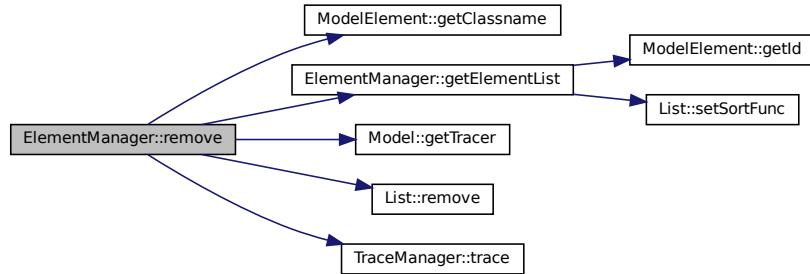
```

00052
00053     std::string elementTypename = anElement->getclassname();
00054     List<ModelElement*>* listElements = getElementList(elementTypename);
00055     listElements->remove(anElement);
00056     _parentModel->getTracer()->trace(Util::TraceLevel::elementResult, "Element successfully removed");
00057
00058 }
```

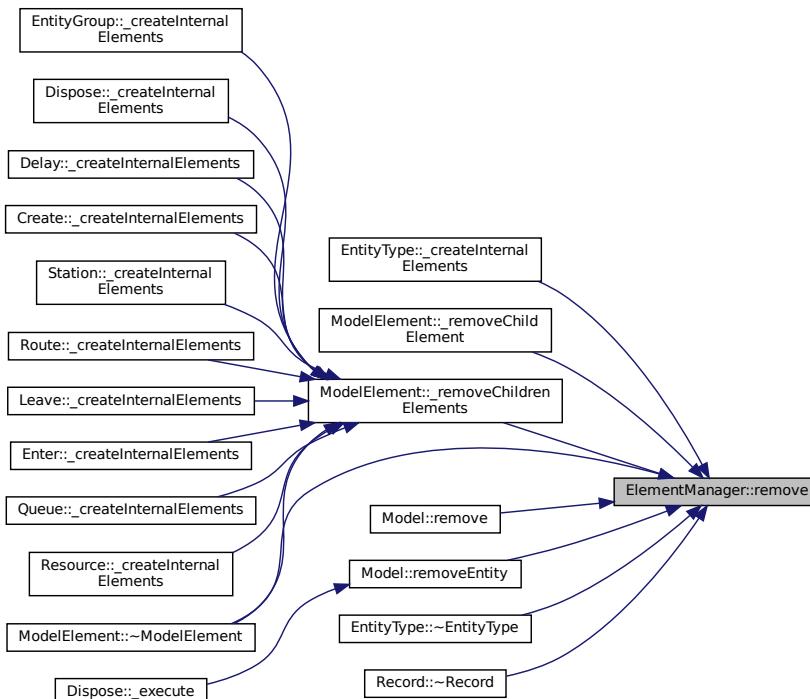
References [Util::elementResult](#), [ModelElement::getclassname\(\)](#), [getElementList\(\)](#), [Model::getTracer\(\)](#), [List< T >::remove\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [EntityType::_createInternalElements\(\)](#), [ModelElement::_removeChildElement\(\)](#), [ModelElement::_removeChildrenElements\(\)](#), [Model::remove\(\)](#), [Model::removeEntity\(\)](#), [EntityType::~EntityType\(\)](#), [ModelElement::~ModelElement\(\)](#), and [Record::~Record\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.16 remove() [2/2] void ElementManager::remove (std::string elementTypename, ModelElement * anElement)

Definition at line 60 of file [ElementManager.cpp](#).

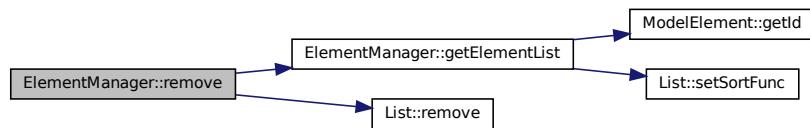
```

00060
00061     List<ModelElement*>* listElements = getElementList(elementTypename);
00062     listElements->remove(anElement);
  
```

```
00063 }
```

References [getElementList\(\)](#), and [List< T >::remove\(\)](#).

Here is the call graph for this function:



8.24.3.17 setHasChanged() void ElementManager::setHasChanged (bool _hasChanged)

Definition at line 136 of file [ElementManager.cpp](#).

```
00136     {
00137         this->_hasChanged = _hasChanged;
00138     }
```

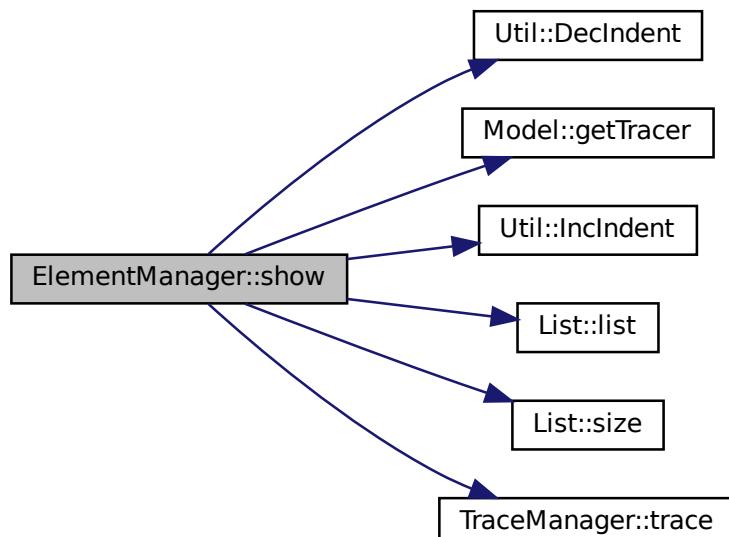
8.24.3.18 show() void ElementManager::show ()

Definition at line 105 of file [ElementManager.cpp](#).

```
00105     {
00106         _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Model Elements:");
00107         //std::map<std::string, List<ModelElement*>>* _elements;
00108         std::string key;
00109         List<ModelElement*>* list;
00110         Util::IncIndent();
00111         {
00112             for (std::map<std::string, List<ModelElement*>>::iterator infraIt = _elements->begin();
00113                 infraIt != _elements->end(); infraIt++) {
00114                 key = (*infraIt).first;
00115                 list = (*infraIt).second;
00116                 _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, key + ": (" +
00117                     std::to_string(list->size()) + ")");
00118                 Util::IncIndent();
00119                 {
00120                     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it !=
00121                         list->list()->end(); it++) {
00122                         _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, (*it)->show());
00123                     }
00124                 }
00125             }
00126         }
00127     }
```

References [Util::DecIndent\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [List< T >::list\(\)](#), [List< T >::size\(\)](#), [Util::toolDetailed](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



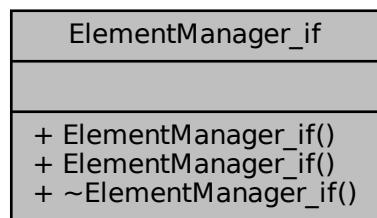
The documentation for this class was generated from the following files:

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

8.25 ElementManager_if Class Reference

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

Collaboration diagram for `ElementManager_if`:



Public Member Functions

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

8.25.1 Detailed Description

Definition at line 17 of file [ElementManager_if.h](#).

8.25.2 Constructor & Destructor Documentation

8.25.2.1 [ElementManager_if\(\)](#) [1/2] ElementManager_if::ElementManager_if ()

```
ElementManager_if::ElementManager_if (
    const ElementManager_if & orig )
```

8.25.2.3 [~ElementManager_if\(\)](#) virtual ElementManager_if::~ElementManager_if () [virtual]

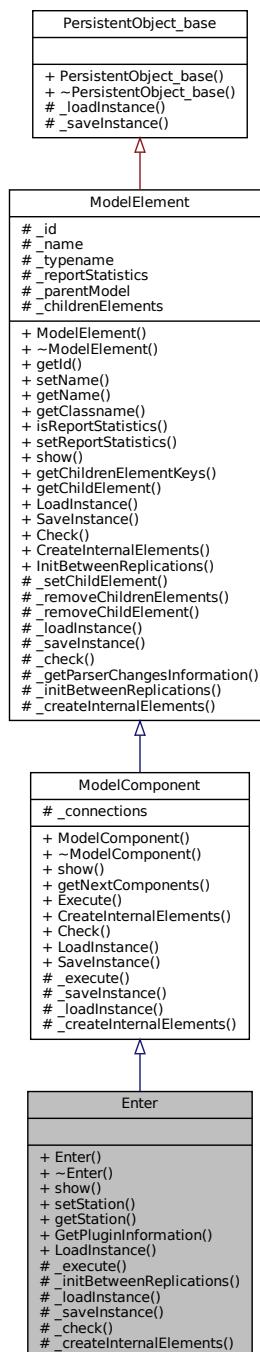
The documentation for this class was generated from the following file:

- [ElementManager_if.h](#)

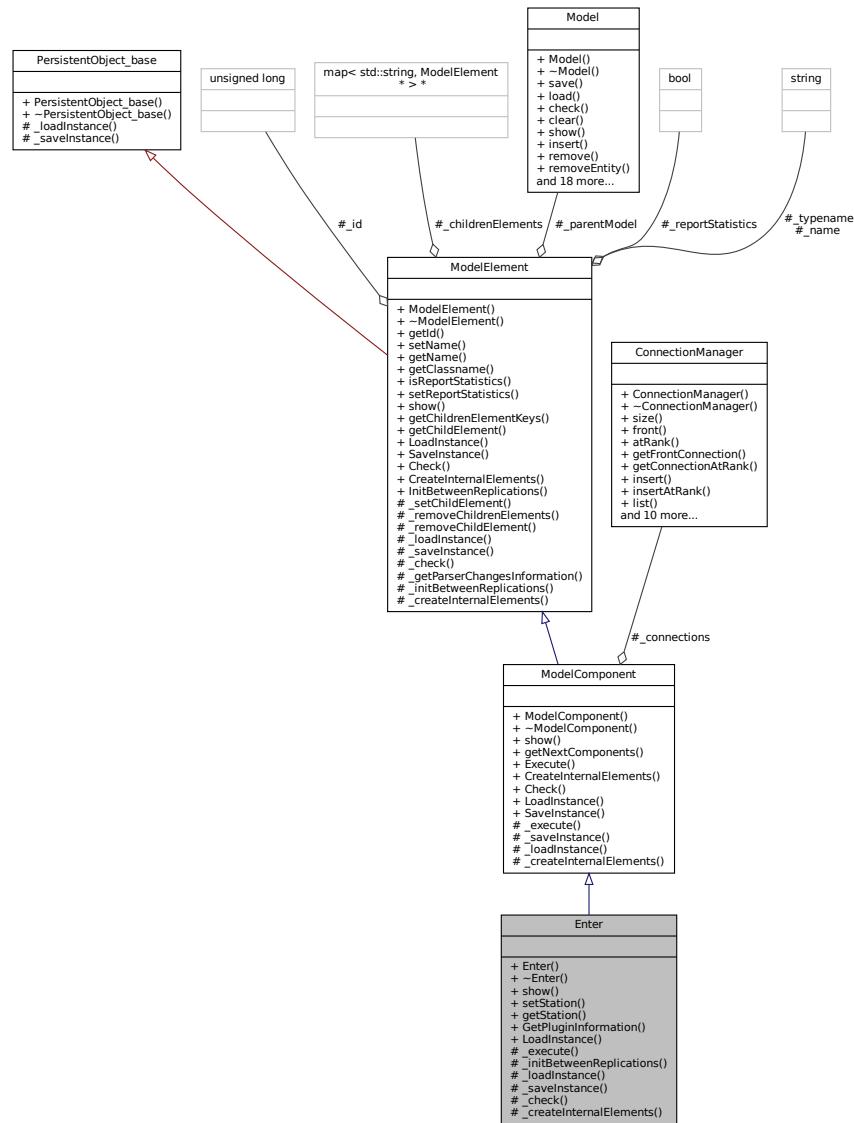
8.26 Enter Class Reference

```
#include <Enter.h>
```

Inheritance diagram for Enter:



Collaboration diagram for Enter:



Public Member Functions

- `Enter (Model *model, std::string name="")`
- virtual `~Enter ()=default`
- virtual std::string `show ()`
- void `setStation (Station *_station)`
- Station * `getStation () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.26.1 Detailed Description

Enter module DESCRIPTION The **Enter** module defines a station (or a set of stations) corresponding to a physical or logical location where processing occurs. When an entity arrives at an **Enter** module, an unloading delay may occur and any transfer device used to transfer the entity to the **Enter** module's station may be released. The station (or each station within the defined set) has a matching Activity Area that is used to report all times and costs accrued by the entities in this station. This Activity Area's name is the same as the station. If a parent Activity Area is defined, then it also accrues any times and costs by the entities in this station. TYPICAL USES The start of a part's production in a series of parallel processes where the part's forklift needs to be released. The start of a document's processing after the document has been created where the mail clerk resource needs to be released. PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Station Type Type of station, either a single **Station** or station **Set**. Station Name Name of the individual station. A given station can only exist once within a model. Parent Activity Area Name of the Activity Area's parent. Associated Intersection Name of the intersection associated with this station in a guided transporter network. Report Statistics Specifies whether or not statistics will automatically be collected and stored in the report database for this station and its corresponding activity area. Set Name Name of the station set. A given station set can only exist once within a model. Save Attribute Specifies the attribute to be used to store the index into the station set for an entity entering this module. Set Members This repeat group permits you to define the individual stations that are to be members of the specified station set. A station set must have at least one member station. Active when Station Type is **Set**. Station Name This field indicates the name of a station that is to be a member of this station set. A given station can only exist within a model once. Therefore, an individual station can only be the member of one station set, and that individual station may not be the name of a station in another module. Parent Activity Area Name of the Activity Area's parent for the station set member. Associated Intersection Name of the intersection associated with this station set in a guided transporter network. Report Statistics Specifies whether or not statistics will automatically be collected and stored in the report database for this station set member and its corresponding activity area. Allocation Type of category to which the entity's incurred delay time and cost will be added. Delay This field defines the delay that will be experienced by entities immediately upon arrival at the station. Units Time units used for the delay time. Transfer In If a resource, transporter, or conveyor was used to transfer the entity to this station, this can be used to release, free, or exit the device. If **Release Resource** is selected, the specified resource is released. If **Free Transporter** is selected, the specified transporter is freed. If **Exit Conveyor** is selected, the specified conveyor is exited. Transporter Name Name of the transporter to be freed upon arrival to the station. Active when Transfer Name is **Free Transporter**. Unit Number Unit number of the transporter if the transporter is multicapacity. Conveyor Name Name of the conveyor to exit upon arrival to the station. Resource Type Type of allocation, either single **Resource** or resource **Set**. Resource Name Name of the resource to release. Active when Transfer Name is **Release Resource**. Set Name Name of the resource set from which the resource is to be released. Release Rule Determines which member of the set is to be released, either the Last Member Seized, First Member Seized, or Specific Member. Set Index Index into the set that determines which member of the set is to be released. Attribute Name Name of the attribute that determines the instance number of the resource to release. Expression Expression value that determines the instance number of the resource to release.

Definition at line 98 of file [Enter.h](#).

8.26.2 Constructor & Destructor Documentation

8.26.2.1 Enter() Enter::Enter (

```
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Enter.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Enter>(), name) {  
00019 }
```

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

Here is the caller graph for this function:



8.26.2.2 ~Enter() virtual Enter::~Enter () [virtual], [default]

8.26.3 Member Function Documentation

8.26.3.1 _check() bool Enter::_check (

```
    std::string * errorMessage ) [protected], [virtual]
```

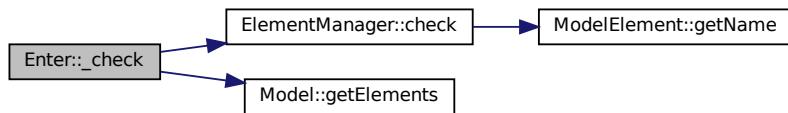
Reimplemented from [ModelElement](#).

Definition at line 71 of file [Enter.cpp](#).

```
00071 {
00072     bool resultAll = true;
00073     resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",
00074         errorMessage);
00075 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



8.26.3.2 `_createInternalElements()` void Enter::_createInternalElements () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

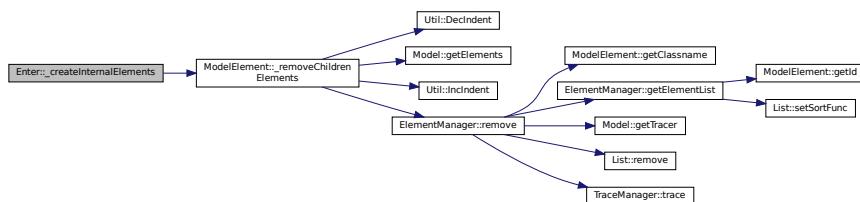
Reimplemented from [ModelComponent](#).

Definition at line 84 of file [Enter.cpp](#).

```
00084
00085     if (_reportStatistics) {
00086         if (_numberIn == nullptr) {
00087             _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00088             _childrenElements->insert({"CountNumberIn", _numberIn});
00089         }
00090     } else
00091         if (_numberIn != nullptr) {
00092             _removeChildrenElements();
00093         }
00094 }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



8.26.3.3 `_execute()` void Enter::_execute (Entity * entity) [protected], [virtual]

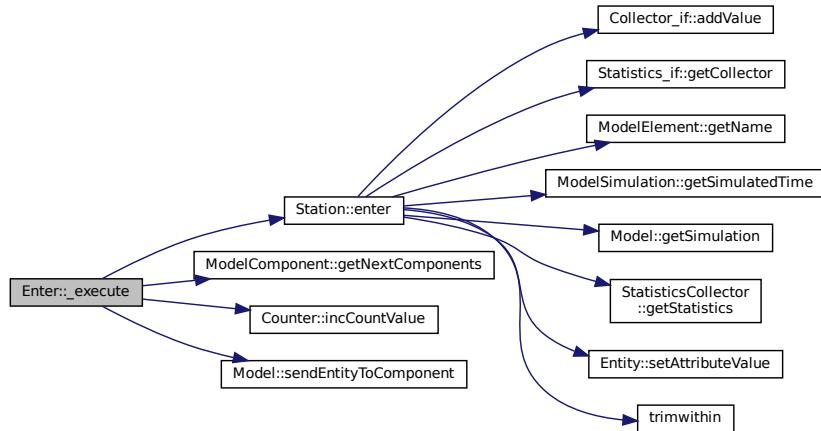
Implements [ModelComponent](#).

Definition at line 44 of file [Enter.cpp](#).

```
00044
00045     if (_reportStatistics)
00046         _numberIn->incCountValue();
00047     _station->enter(entity);
00048     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00049 }
```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [Station::enter\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Counter::incCountValue\(\)](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.26.3.4 `_initBetweenReplications()` void Enter::__initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 61 of file [Enter.cpp](#).

```

00061
00062     _numberIn->clear();
00063 }
```

References [Counter::clear\(\)](#).

Here is the call graph for this function:



8.26.3.5 `_loadInstance()` bool Enter::__loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Enter.cpp](#).

```

00051
00052     bool res = ModelComponent::__loadInstance(fields);
```

```

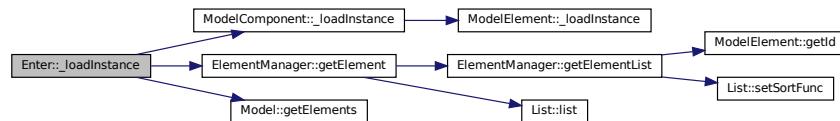
00053     if (res) {
00054         std::string stationName = ((*fields->find("stationName"))).second;
00055         Station* station = dynamic_cast<Station*>
00056             (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00057         this->_station = station;
00058     }
00059     return res;
00059 }

```

References [ModelComponent::_loadInstance\(\)](#), [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), and [Model::getElements\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.26.3.6 `_saveInstance()`

`std::map< std::string, std::string > * Enter::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelComponent](#).

Definition at line 65 of file [Enter.cpp](#).

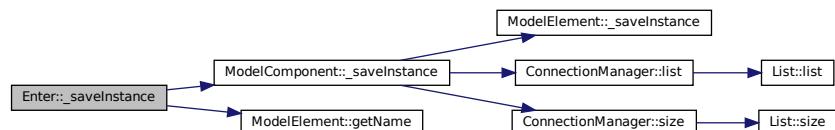
```

00065
00066     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);
00067     fields->emplace("stationName", "\"" + (this->_station->getName()) + "\"");
00068     return fields;
00069 }

```

References [ModelComponent::_saveInstance\(\)](#), and [ModelElement::getName\(\)](#).

Here is the call graph for this function:



8.26.3.7 GetPluginInformation() `PluginInformation * Enter::GetPluginInformation () [static]`

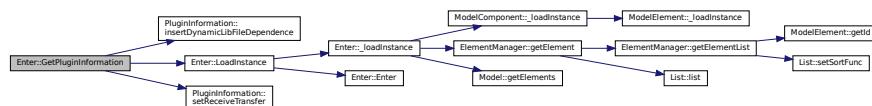
Definition at line 77 of file [Enter.cpp](#).

```
00077     {
00078         PluginInformation* info = new PluginInformation(Util::TypeOf<Enter>(), &Enter::LoadInstance);
00079         info->setReceiveTransfer(true);
00080         info->insertDynamicLibFileDependence("station.so");
00081         return info;
00082     }
```

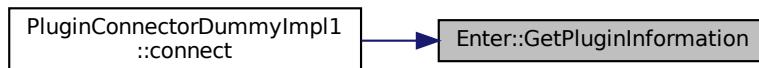
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), [LoadInstance\(\)](#), and [PluginInformation::setReceiveTransfer\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.26.3.8 getStation() `Station * Enter::getStation () const`

Definition at line 40 of file [Enter.cpp](#).

```
00040     {
00041         return _station;
00042     }
```

8.26.3.9 LoadInstance() `ModelComponent * Enter::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Enter.cpp](#).

```
00025     {
00026         Enter* newComponent = new Enter(model);
00027         try {
00028             newComponent->_loadInstance(fields);
00029         } catch (const std::exception& e) {
00030         }
00031         return newComponent;
00032     }
```

References `_loadInstance()`, and `Enter()`.

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.26.3.10 setStation() void Enter::setStation (Station * _station)

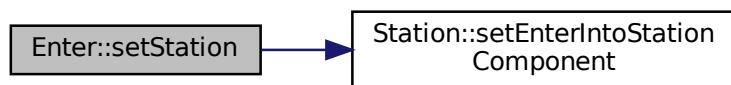
Definition at line 35 of file [Enter.cpp](#).

```
00035     ...
00036     this->_station = _station;
00037     _station->setEnterIntoStationComponent (this);
00038 }
```

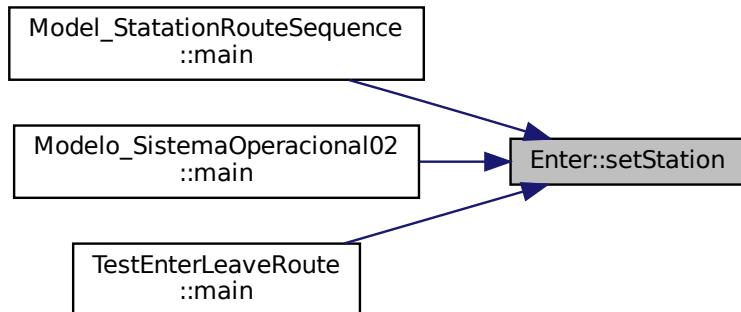
References Station::setEnterIntoStationComponent().

Referenced by [Model_StationRouteSequence::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), and [Modelo_SistemaOperacional02::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.26.3.11 show() std::string Enter::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Enter.cpp](#).

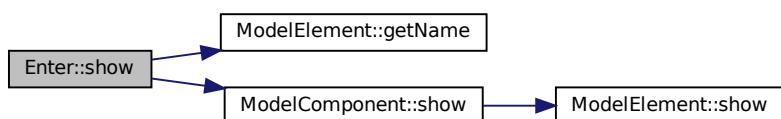
```

00021     {
00022         return ModelComponent::show() + ",station=" + this->_station->getName();
00023     }

```

References [ModelElement::getName\(\)](#), and [ModelComponent::show\(\)](#).

Here is the call graph for this function:



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

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

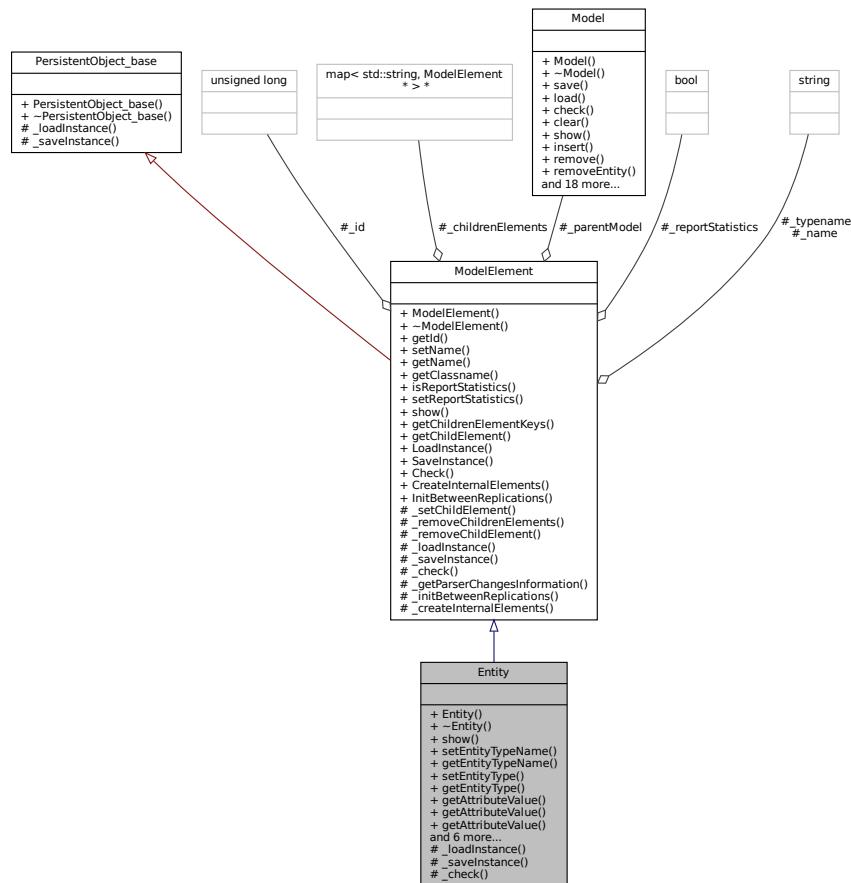
8.27 Entity Class Reference

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

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- Entity (Model *model)
 - virtual ~Entity ()=default
 - virtual std::string show ()
 - void setEntityType (std::string entityTypeName) throw ()
 - std::string getEntityType () const
 - void setEntityType (EntityType *entityType)
 - EntityType * getEntityType () const
 - double getAttributeValue (std::string attributeName)
 - double getAttributeValue (std::string index, std::string attributeName)
 - double getAttributeValue (Util::identification attributeID)
 - double getAttributeValue (std::string index, Util::identification attributeID)
 - void setAttributeValue (std::string attributeName, double value)
 - void setAttributeValue (std::string index, std::string attributeName, double value)
 - void setAttributeValue (Util::identification attributeID, double value)
 - void setAttributeValue (std::string index, Util::identification attributeID, double value)
 - Util::identification entityNumber () const

Protected Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)

Additional Inherited Members

8.27.1 Detailed Description

Entity module DESCRIPTION This data module defines the various entity types and their initial picture values in a simulation. Initial costing information and holding costs are also defined for the entity. TYPICAL USES Items being produced or assembled (parts, pallets) Documents (forms, e-mails, faxes, reports) People moving through a process (customers, callers) PROMPTS Prompt Description Name The unique name of the attribute being defined. Rows Number of rows in a one- or two-dimensional attribute. Columns Number of columns in a two-dimensional attribute. Data Type The data type of the values stored in the attribute. Valid types are Real and String. The default type is Real. Initial Values Lists the initial value or values of the attribute. You can assign new values to the attribute by using the [Assign](#) module. Initial Value Entity attribute value when entity is created and enters the system. Prompt Description Entity Type The name of the entity type being defined. This name must be unique. Initial Picture Graphical representation of the entity at the start of the simulation. This value can be changed during the simulation using the [Assign](#) module. Holding Cost/Hour Hourly cost of processing the entity through the system. This cost is incurred when the entity is anywhere in the system. Initial VA Cost Initial cost value that will be assigned to the value-added cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a value-added activity. Initial NVA Cost Initial cost value that will be assigned to the non-value-added cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a non-value-added activity. Initial Waiting Cost Initial cost value that will be assigned to the waiting-cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a wait activity; for example, waiting to be batched or waiting for resource(s) at a Process module. Initial Transfer Cost Initial cost value that will be assigned to the transfer cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a transfer activity. Initial Other Cost Initial cost value that will be assigned to the other cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in another activity. Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this entity type.

Definition at line 76 of file [Entity.h](#).

8.27.2 Constructor & Destructor Documentation

8.27.2.1 Entity() Entity::Entity (

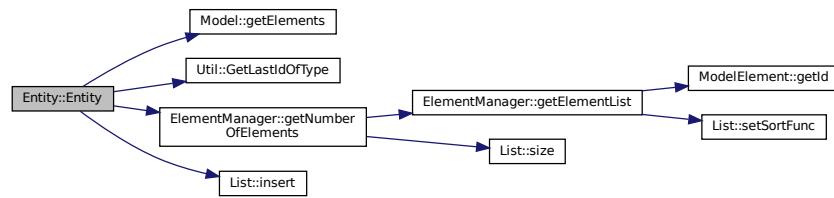
`Model * model`)

Definition at line 21 of file [Entity.cpp](#).

```
00021     : ModelElement(model, Util::TypeOf<Entity>()) {  
00022     //elements = elements;  
00023     _entityNumber = Util::GetLastIdOfType(Util::TypeOf<Entity>());  
00024     unsigned int numAttributes =  
     _parentModel->getElements()->getNumberOfElements(Util::TypeOf<Attribute>());  
00025     for (unsigned i = 0; i < numAttributes; i++) {  
00026         std::map<std::string, double>* map = new std::map<std::string, double>();  
00027         _attributeValues->insert(map);  
00028     }  
00029 }
```

References [ModelElement::_parentModel](#), [Model::getElements\(\)](#), [Util::GetLastIdOfType\(\)](#), [ElementManager::getNumberOfElements\(\)](#) and [List< T >::insert\(\)](#).

Here is the call graph for this function:



8.27.2.2 ~Entity() virtual Entity::~Entity () [virtual], [default]

8.27.3 Member Function Documentation

8.27.3.1 _check() bool Entity::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 155 of file [Entity.cpp](#).

```
00155
00156     return true;
00157 }
```

8.27.3.2 _loadInstance() bool Entity::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 146 of file [Entity.cpp](#).

```
00146
00147     // never loads an entity
00148     return true;
00149 }
```

8.27.3.3 _saveInstance() std::map< std::string, std::string > * Entity::_saveInstance () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 151 of file [Entity.cpp](#).

```
00151
00152     return new std::map<std::string, std::string>();
00153 }
```

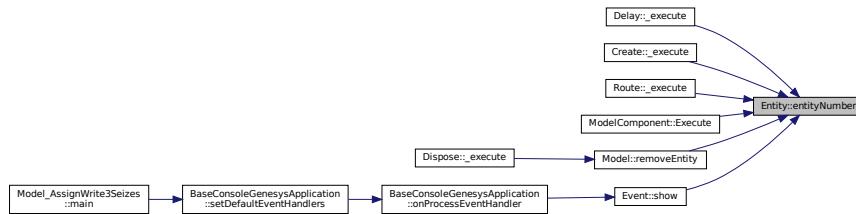
8.27.3.4 entityNumber() `Util::identification Entity::entityNumber () const`

Definition at line 142 of file `Entity.cpp`.

```
00142
00143     return _entityNumber;
00144 }
```

Referenced by `Delay::_execute()`, `Create::_execute()`, `Route::_execute()`, `ModelComponent::Execute()`, `Model::removeEntity()`, and `Event::show()`.

Here is the caller graph for this function:



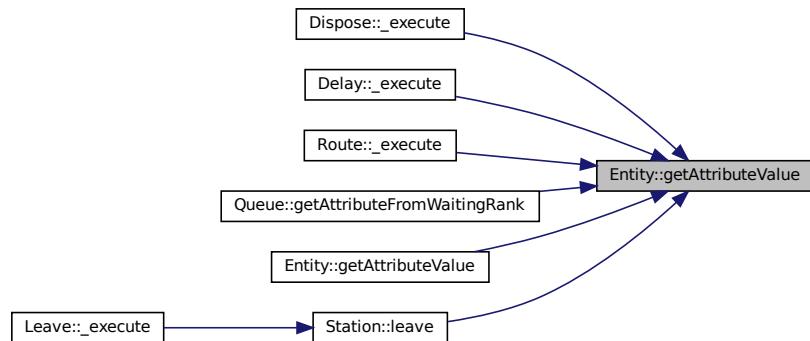
8.27.3.5 getAttributeValue() [1/4] `double Entity::getAttributeValue (std::string attributeName)`

Definition at line 83 of file `Entity.cpp`.

```
00083
00084     return getAttributeValue("", attributeName);
00085 }
```

Referenced by `Dispose::_execute()`, `Delay::_execute()`, `Route::_execute()`, `Queue::getAttributeFromWaitingRank()`, `getAttributeValue()`, and `Station::leave()`.

Here is the caller graph for this function:



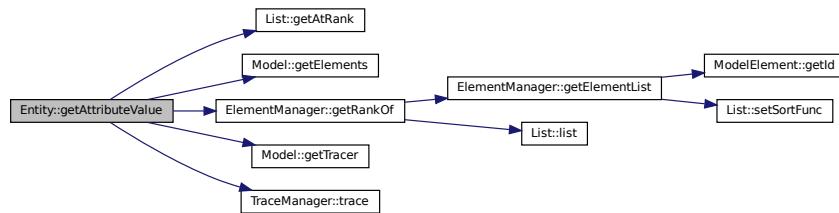
```
8.27.3.6 getAttributeValue() [2/4] double Entity::getAttributeValue (
    std::string index,
    std::string attributeName )
```

Definition at line 87 of file Entity.cpp.

```
00087     {
00088     int rank = _parentModel->getElements()->getRankOf(Util::TypeOf<Attribute>(), attributeName);
00089     if (rank >= 0) {
00090         std::map<std::string, double>* map = this->_attributeValues->getAtRank(rank);
00091         std::map<std::string, double>::iterator mapIt = map->find(index);
00092         if (mapIt != map->end()) { //found
00093             return (*mapIt).second;
00094         } else { // not found
00095             return 0.0;
00096         }
00097     }
00098     _parentModel->getTracer()->trace(Util::TraceLevel::errorRecover, "Attribute \""
+ attributeName +
"\\" not found");
00099     return 0.0; /* \todo: !! Never should happen. check how to report */
00100 }
```

References [ModelElement::_parentModel](#), [Util::errorRecover](#), [List< T >::getAtRank\(\)](#), [Model::getElements\(\)](#), [ElementManager::getRankOf\(\)](#), [Model::getTracer\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



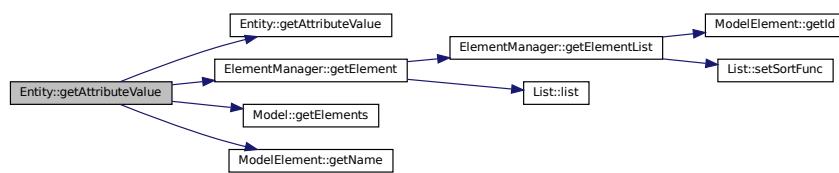
```
8.27.3.7 getAttributeValue() [3/4] double Entity::getAttributeValue (
    std::string index,
    Util::identification attributeID )
```

Definition at line 106 of file Entity.cpp.

```
00106     {
00107     ModelElement* element = _parentModel->getElements()->getElement(Util::TypeOf<Attribute>(),
00108     attributeID);
00109     if (element != nullptr) {
00110         return getAttributeValue(index, element->getName());
00111     }
00112     return 0.0; // attribute not found
00113 }
```

References [ModelElement::_parentModel](#), [getAttributeValue\(\)](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), and [ModelElement::getName\(\)](#).

Here is the call graph for this function:



8.27.3.8 getAttributeValue() [4/4] `double Entity::getAttributeValue (`
 `Util::identification attributeID)`

Definition at line 102 of file [Entity.cpp](#).

```
00102
00103     return getAttributeValue("", attributeID);
00104 }
```

References [getAttributeValue\(\)](#).

Here is the call graph for this function:



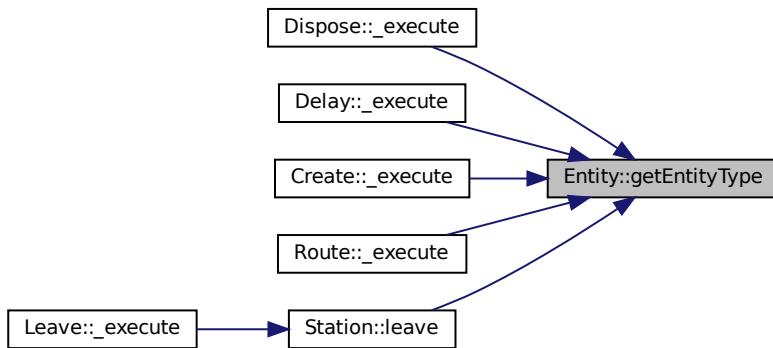
8.27.3.9 getEntityType() `EntityType * Entity::getEntityType () const`

Definition at line 48 of file [Entity.cpp](#).

```
00048
00049     return _entityType;
00050 }
```

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), and [Station::leave\(\)](#).

Here is the caller graph for this function:



8.27.3.10 getEntityTypeName() std::string Entity::getEntityTypeName () const

Definition at line 40 of file [Entity.cpp](#).

```
00040     {
00041         return this->_entityType->getName();
00042     }
```

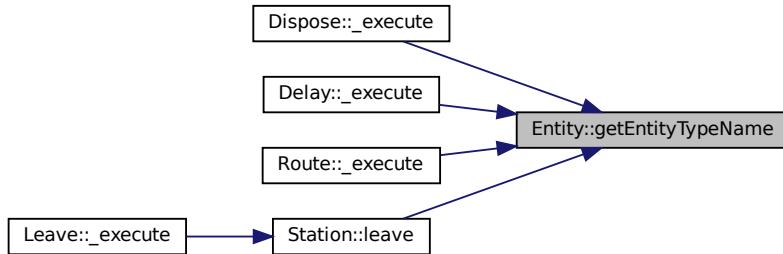
References [ModelElement::getName\(\)](#).

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Route::_execute\(\)](#), and [Station::leave\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.27.3.11 setAttributeValue() [1/4] void Entity::setAttributeValue (

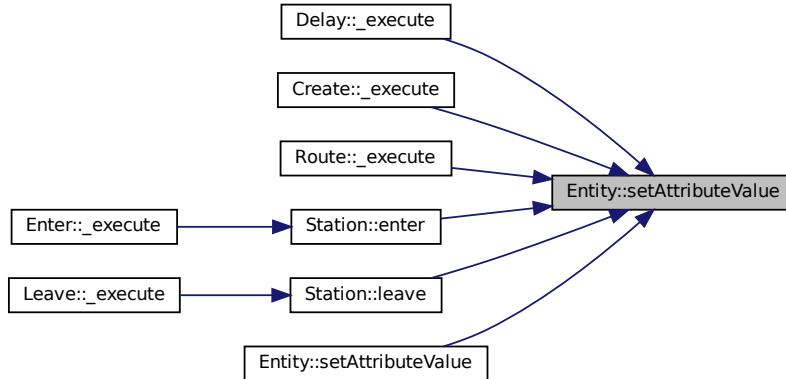
```
    std::string attributeName,
    double value )
```

Definition at line 114 of file [Entity.cpp](#).

```
00114     {
00115         setAttributeValue("", attributeName, value);
00116     }
```

Referenced by [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Station::enter\(\)](#), [Station::leave\(\)](#), and [setAttributeValue\(\)](#).

Here is the caller graph for this function:



8.27.3.12 setAttributeValue() [2/4] void Entity::setAttributeValue (std::string index, std::string attributeName, double value)

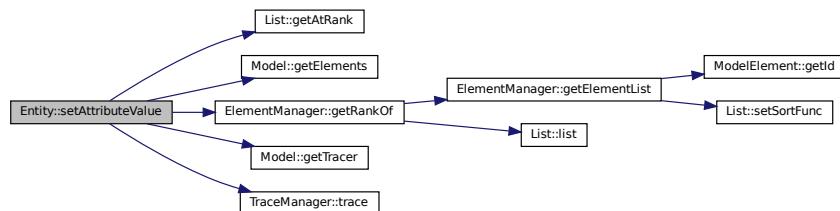
Definition at line 118 of file [Entity.cpp](#).

```

00118
00119     int rank = _parentModel->getElements()->getRankOf(Util::TypeOf<Attribute>(), attributeName);
00120     if (rank >= 0) {
00121         std::map<std::string, double>* map = _attributeValues->getAtRank(rank);
00122         std::map<std::string, double>::iterator mapIt = map->find(index);
00123         if (mapIt != map->end()) { //found
00124             (*mapIt).second = value;
00125         } else { // not found
00126             map->insert({index, value}); // (map->end(), std::pair<std::string, double>(index,
00127             value));
00128         }
00129     } else
00129         _parentModel->getTracer()->trace(Util::TraceLevel::errorRecover, "Attribute \""
00130         attributeName + "\" not found");
00131 }
```

References [ModelElement::_parentModel](#), [Util::errorRecover](#), [List< T >::getAtRank\(\)](#), [Model::getElements\(\)](#), [ElementManager::getRankOf\(\)](#), [Model::getTracer\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



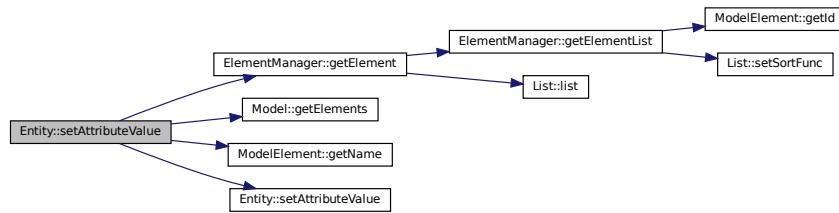
8.27.3.13 setAttributeValue() [3/4] void Entity::setAttributeValue (std::string index, Util::identification attributeID, double value)

Definition at line 137 of file Entity.cpp.

```
00137
00138     std::string attrname = _parentModel->getElements ()->getElement (Util::TypeOf<Attribute>(),
00139         attributeID)->getName ();
00140     setAttributeValue (index, attrname, value);
00140 }
```

References ModelElement::_parentModel, ElementManager::getElement(), Model::getElements(), ModelElement::getName(), and setAttributeValue().

Here is the call graph for this function:



8.27.3.14 setAttributeValue() [4/4] void Entity::setAttributeValue (Util::identification attributeID, double value)

Definition at line 133 of file Entity.cpp.

```
00133
00134     setAttributeValue ("", attributeID, value);
00135 }
```

References setAttributeValue().

Here is the call graph for this function:



8.27.3.15 setEntityType() void Entity::setEntityType (EntityType * entityType)

Definition at line 44 of file [Entity.cpp](#).

```
00044     _entityType = entityType;
00045 }
00046 }
```

Referenced by [Create::_execute\(\)](#).

Here is the caller graph for this function:



8.27.3.16 setEntityTypeName() void Entity::setEntityTypeName (std::string entityTypeName) throw ()

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

```
00031
00032     EntityType* entitytype = dynamic_cast<EntityType*>
00033         (_parentModel->getElements()->getElement(Util::TypeOf<EntityType>(), entityTypeName));
00034     if (entitytype != nullptr) {
00035         this->_entityType = entitytype;
00036     } else {
00037         throw std::invalid_argument("EntityType does not exist");
00038     }
```

8.27.3.17 show() std::string Entity::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 52 of file [Entity.cpp](#).

```
00052     {
00053         std::string message = ModelElement::show();
00054         if (this->_entityType != nullptr) {
00055             message += ",entityType=\"";
00056             message += this->_entityType->getName() + "\"";
00057         }
00058         message += ",attributes=\"";
00059         _attributeValues->front();
00060         for (unsigned int i = 0; i < _attributeValues->size(); i++) {
00061             std::map<std::string, double>* map = _attributeValues->current();
00062             std::string attributeName =
00063                 _parentModel->getElements()->getElementList(Util::TypeOf<Attribute>())->getAtRank(i)->getName();
00064             message += attributeName + "=";
00065             if (map->size() == 0) { // scalar
00066                 message += "NaN"; // std::to_string(map->begin() ->second) + ";";
00067             } else if (map->size() == 1) { // scalar
00068                 message += std::to_string(map->begin() ->second) + ",";
00069             } else {
00070                 // array or matrix
00071                 message += "[";
00072                 for (std::map<std::string, double>::iterator valIt = map->begin(); valIt != map->end();
00073                     valIt++)
00074                     message += valIt->second + ",";
```

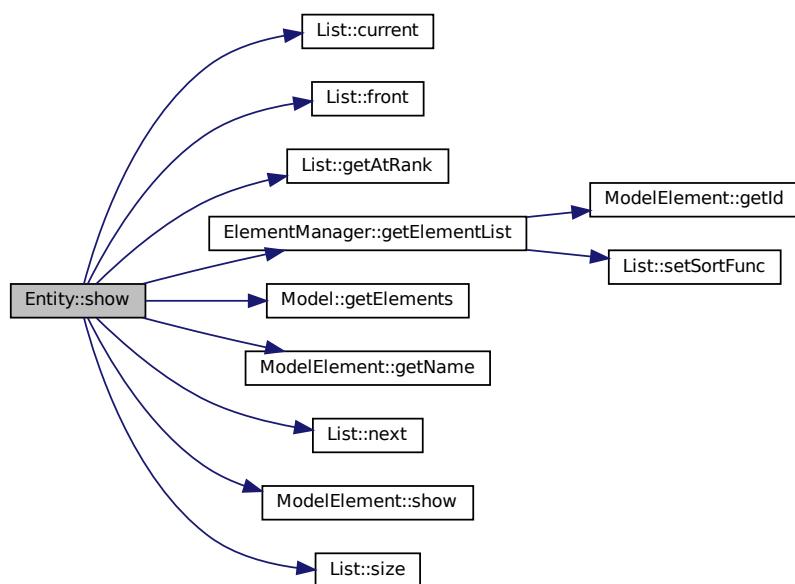
```

00071             message += (*valIt).first + "=>" + std::to_string((*valIt).second) + ",";
00072         }
00073         message = message.substr(0, message.length() - 1);
00074         message += "]";
00075     }
00076     _attributeValues->next();
00077 }
00078 message = message.substr(0, message.length() - 1);
00079 message += "]";
00080 return message;
00081 }

```

References [ModelElement::_parentModel](#), [List< T >::current\(\)](#), [List< T >::front\(\)](#), [List< T >::getAtRank\(\)](#), [ElementManager::getElementList\(\)](#), [Model::getElements\(\)](#), [ModelElement::getName\(\)](#), [List< T >::next\(\)](#), [ModelElement::show\(\)](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



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

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

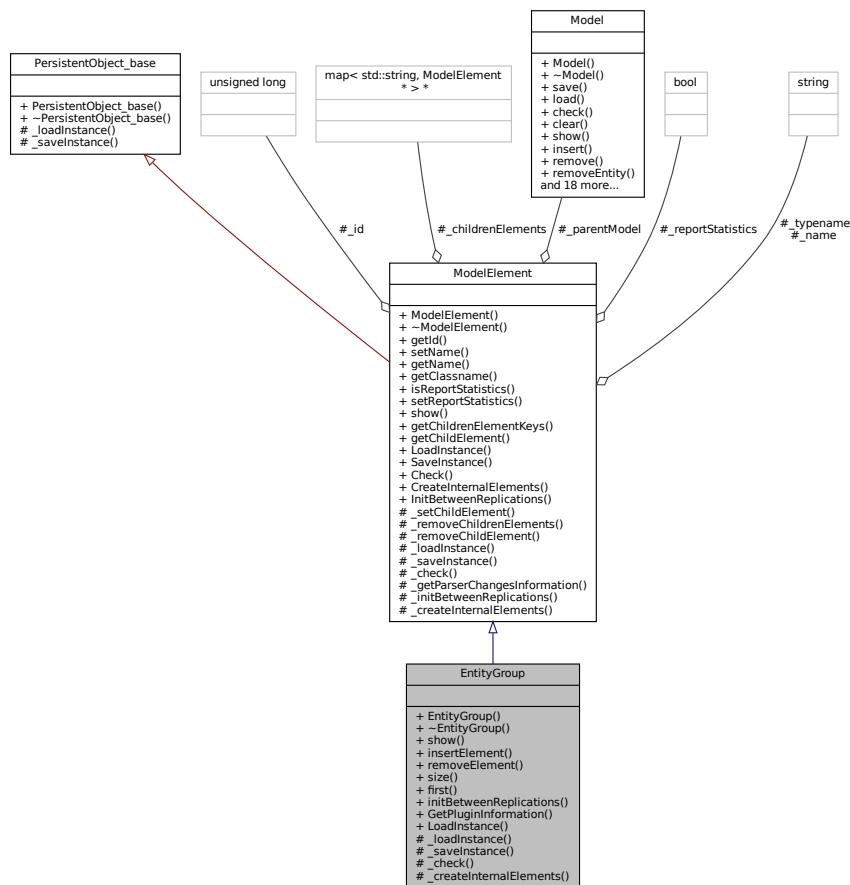
8.28 EntityGroup Class Reference

```
#include <EntityGroup.h>
```

Inheritance diagram for EntityGroup:



Collaboration diagram for EntityGroup:



Public Member Functions

- `EntityGroup (Model *model, std::string name="")`
- `virtual ~EntityGroup ()`
- `virtual std::string show ()`
- `void insertElement (Entity *element)`
- `void removeElement (Entity *element)`
- `unsigned int size ()`
- `Entity * first ()`
- `void initBetweenReplications ()`

Static Public Member Functions

- `static PluginInformation * GetPluginInformation ()`
- `static ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)
- virtual void [_createInternalElements](#) ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Additional Inherited Members

8.28.1 Detailed Description

Definition at line 25 of file [EntityGroup.h](#).

8.28.2 Constructor & Destructor Documentation

8.28.2.1 EntityGroup() EntityGroup::EntityGroup (

```
Model * model,
std::string name = "")
```

Definition at line 18 of file [EntityGroup.cpp](#).

```
00018     Util::TypeOf<EntityGroup>(), name) {
00019 }
```

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

Here is the caller graph for this function:



8.28.2.2 ~EntityGroup() EntityGroup::~EntityGroup () [virtual]

Definition at line 21 of file [EntityGroup.cpp](#).

```
00021 {
00022     //__parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), __cstatNumberInGroup);
00023 }
```

8.28.3 Member Function Documentation

8.28.3.1 `_check()` `bool EntityGroup::_check (std::string * errorMessage) [protected], [virtual]`

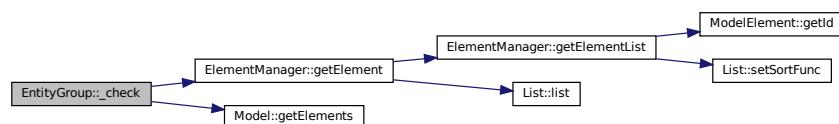
Reimplemented from [ModelElement](#).

Definition at line 88 of file [EntityGroup.cpp](#).

```
00088     {
00089         std::string newNeededAttributeName = "Entity.Group";
00090         if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), newNeededAttributeName) ==
00091             nullptr) {
00092             new Attribute(_parentModel, newNeededAttributeName);
00093         }
00094     }
00094 }
```

References [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



8.28.3.2 `_createInternalElements()` `void EntityGroup::_createInternalElements () [protected], [virtual]`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

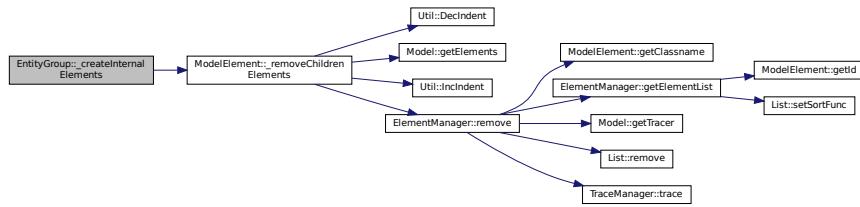
Reimplemented from [ModelElement](#).

Definition at line 96 of file [EntityGroup.cpp](#).

```
00096     {
00097         if (_reportStatistics) {
00098             if (_cstatNumberInGroup == nullptr) {
00099                 _cstatNumberInGroup = new StatisticsCollector(_parentModel, "NumberInGroup", this);
00100                 _childrenElements->insert({"NumberInGroup", _cstatNumberInGroup});
00101             }
00102         } else
00103             if (_cstatNumberInGroup != nullptr) {
00104                 _removeChildrenElements();
00105             }
00106     }
```

References [ModelElement::_childrenElements](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements\(\)](#), and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



8.28.3.3 `_loadInstance()` `bool EntityGroup::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 73 of file [EntityGroup.cpp](#).

```

00073
00074     bool res = ModelElement::_loadInstance(fields);
00075     if (res) {
00076       try {
00077         } catch (...) {
00078       }
00079     }
00080     return res;
00081 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.28.3.4 `_saveInstance()` `std::map< std::string, std::string > * EntityGroup::_saveInstance ()`
[protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 83 of file [EntityGroup.cpp](#).

```
00083
00084     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00085     //Util::TypeOf<Group>());
00086     return fields;
00086 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.28.3.5 `first()` `Entity * EntityGroup::first ()`

Definition at line 50 of file [EntityGroup.cpp](#).

```
00050
00051     return _list->front();
00052 }
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



8.28.3.6 GetPluginInformation() `PluginInformation * EntityGroup::GetPluginInformation () [static]`

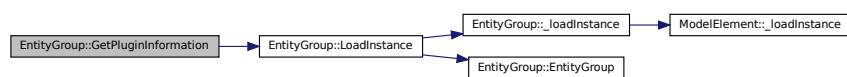
Definition at line 58 of file `EntityGroup.cpp`.

```
00058     {
00059         PluginInformation* info = new PluginInformation(Util::TypeOf<EntityGroup>(),
00060             &EntityGroup::LoadInstance);
00060         return info;
00061     }
```

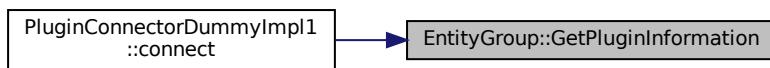
References `LoadInstance()`.

Referenced by `PluginConnectorDummyImpl1::connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



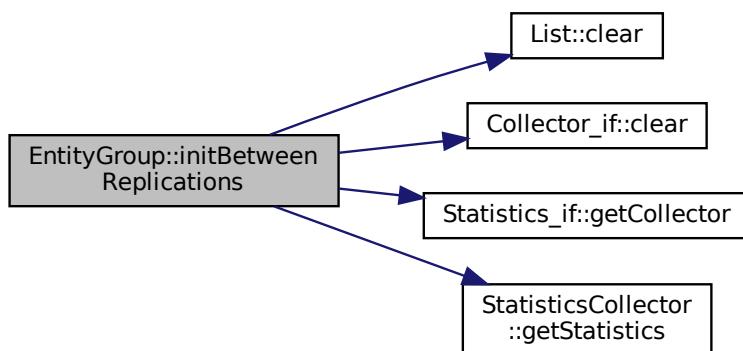
8.28.3.7 initBetweenReplications() `void EntityGroup::initBetweenReplications ()`

Definition at line 41 of file `EntityGroup.cpp`.

```
00041     {
00042         this->_list->clear();
00043         this->_cstatNumberInGroup->getStatistics()->getCollector()->clear();
00044     }
```

References `List< T >::clear()`, `Collector_if::clear()`, `Statistics_if::getCollector()`, and `StatisticsCollector::getStatistics()`.

Here is the call graph for this function:



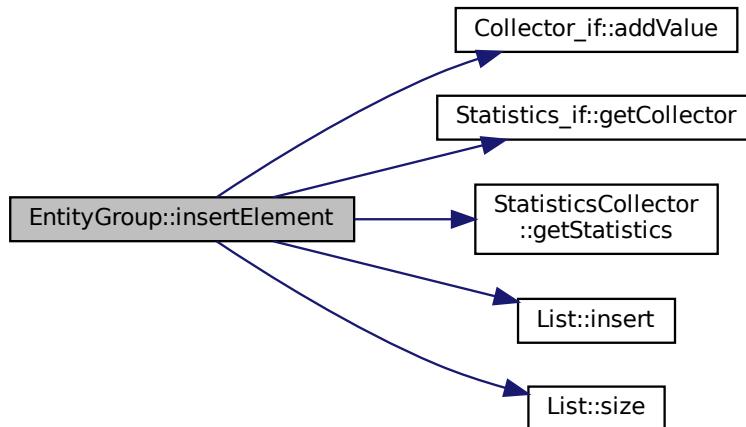
8.28.3.8 insertElement() `void EntityGroup::insertElement (Entity * element)`

Definition at line 30 of file [EntityGroup.cpp](#).

```
00030
00031     _list->insert(element);
00032     this->cstatNumberInGroup->getStatistics()->getCollector()->addValue(_list->size());
00033 }
```

References [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [List< T >::insert\(\)](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.28.3.9 LoadInstance() `ModelElement * EntityGroup::LoadInstance (Model * model, std::map< std::string, std::string * > * fields) [static]`

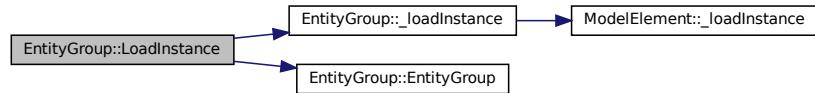
Definition at line 63 of file [EntityGroup.cpp](#).

```
00063
00064     EntityGroup* newElement = new EntityGroup(model);
00065     try {
00066         newElement->_loadInstance(fields);
00067     } catch (const std::exception& e) {
00068     }
00069 }
00070     return newElement;
00071 }
```

References [_loadInstance\(\)](#), and [EntityGroup\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.28.3.10 removeElement() void EntityGroup::removeElement (Entity * element)

Definition at line 35 of file [EntityGroup.cpp](#).

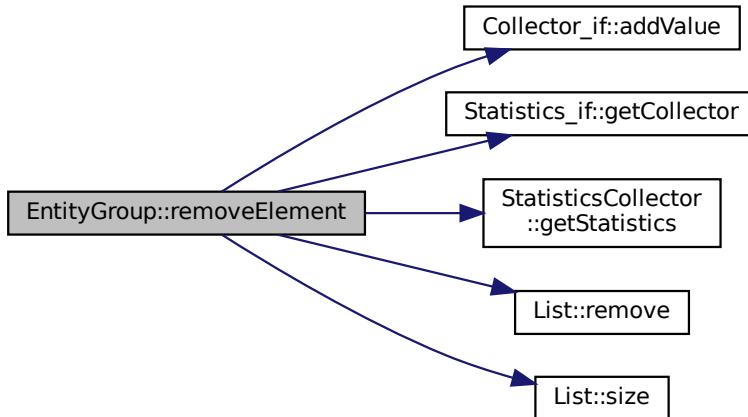
```

00035     {
00036     //double tnow = this->_elements->getParentModel()->simulation()->getSimulatedTime();
00037     _list->remove(element);
00038     this->_cstatNumberInGroup->getStatistics()->getCollector()->addValue(_list->size());
00039 }

```

References [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [List< T >::remove\(\)](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.28.3.11 show() std::string EntityGroup::show () [virtual]

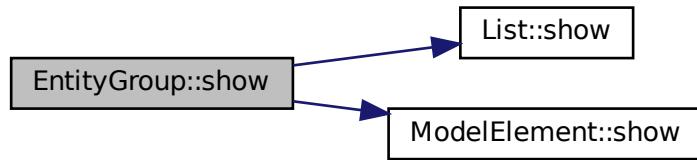
Reimplemented from [ModelElement](#).

Definition at line 25 of file [EntityGroup.cpp](#).

```
00025     {
00026     return ModelElement::show() +
00027         ",entities=" + this->_list->show();
00028 }
```

References [List< T >::show\(\)](#), and [ModelElement::show\(\)](#).

Here is the call graph for this function:

**8.28.3.12 size()** unsigned int EntityGroup::size ()

Definition at line 46 of file [EntityGroup.cpp](#).

```
00046 {
00047     return _list->size();
00048 }
```

References [List< T >::size\(\)](#).

Here is the call graph for this function:



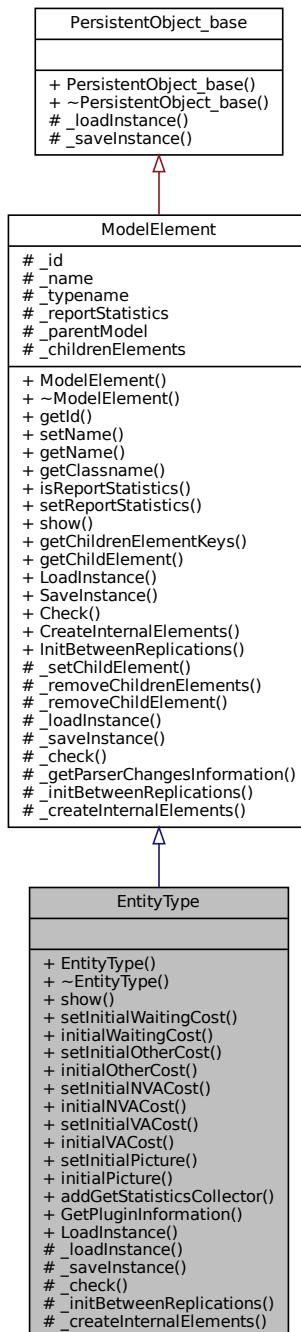
The documentation for this class was generated from the following files:

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

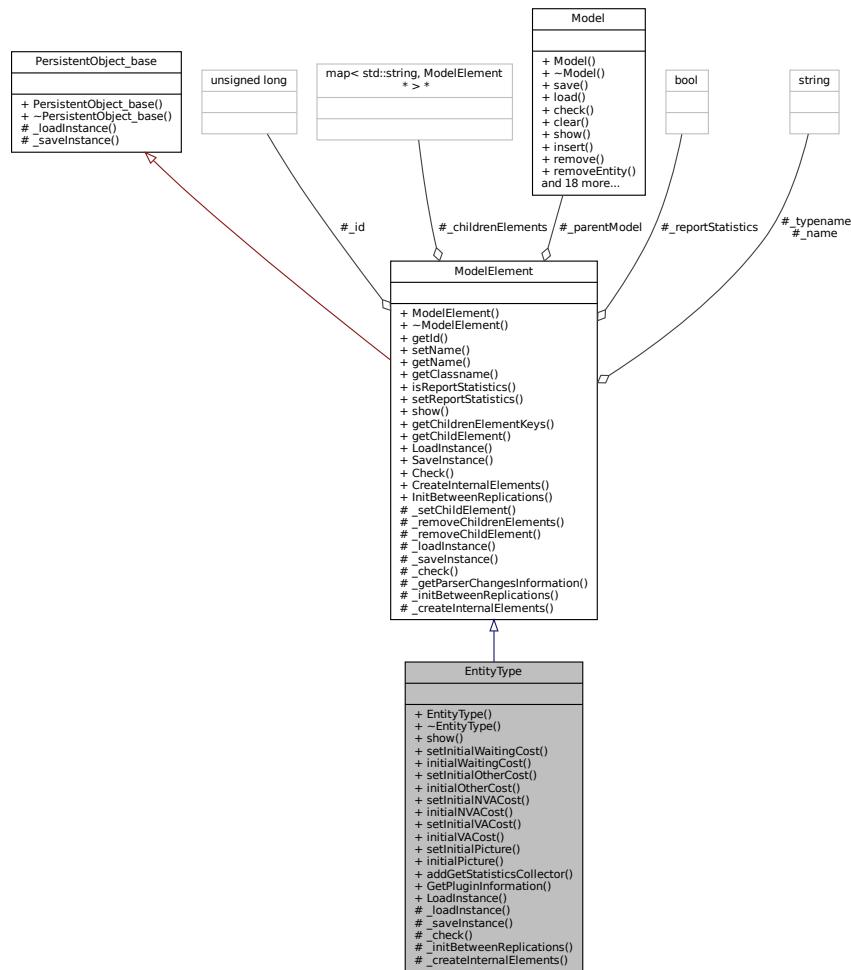
8.29 EntityType Class Reference

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

Inheritance diagram for EntityType:



Collaboration diagram for EntityType:



Public Member Functions

- `EntityType (Model *model, std::string name="")`
- virtual `~EntityType ()`
- virtual `std::string show ()`
- void `setInitialWaitingCost (double _initialWaitingCost)`
- double `initialWaitingCost () const`
- void `setInitialOtherCost (double _initialOtherCost)`
- double `initialOtherCost () const`
- void `setInitialNVACost (double _initialNVACost)`
- double `initialNVACost () const`
- void `setInitialVACost (double _initialVACost)`
- double `initialVACost () const`
- void `setInitialPicture (std::string _initialPicture)`
- std::string `initialPicture () const`
- `StatisticsCollector * addGetStatisticsCollector (std::string name)`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_initBetweenReplications ()`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Additional Inherited Members

8.29.1 Detailed Description

Definition at line 26 of file [EntityType.h](#).

8.29.2 Constructor & Destructor Documentation

```
8.29.2.1 EntityType() EntityType::EntityType (
    Model * model,
    std::string name = "")
```

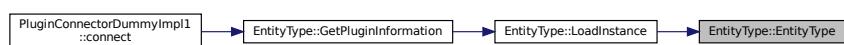
Definition at line 20 of file [EntityType.cpp](#).

```
00020     name) {
00021 }
```

`: ModelElement (model, Util::TypeOf<EntityType> (),`

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

Here is the caller graph for this function:



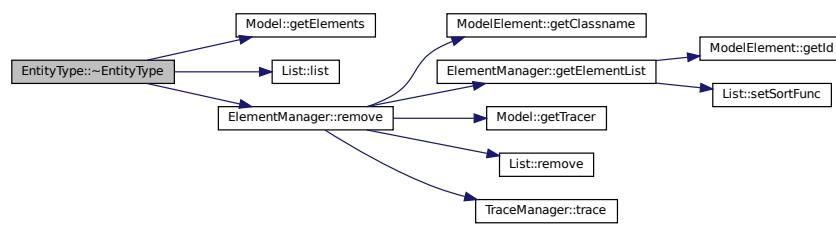
8.29.2.2 ~EntityType() EntityType::~EntityType () [virtual]

Definition at line 33 of file [EntityType.cpp](#).

```
00033     {
00034         // remove all CStats
00035         for (std::list<StatisticsCollector*>::iterator it = this->_statisticsCollectors->list()->begin();
00036             it != this->_statisticsCollectors->list()->end(); it++) {
00037             _parentModel->getElements()->remove(Util::TypeOf<StatisticsCollector>(), (*it));
00038     }
```

References [ModelElement::_parentModel](#), [Model::getElements\(\)](#), [List< T >::list\(\)](#), and [ElementManager::remove\(\)](#).

Here is the call graph for this function:



8.29.3 Member Function Documentation

8.29.3.1 _check() bool EntityType::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 138 of file [EntityType.cpp](#).

```
00138
00139     return true;
00140 }
```

8.29.3.2 _createInternalElements() void EntityType::_createInternalElements () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Reimplemented from [ModelElement](#).

Definition at line 142 of file [EntityType.cpp](#).

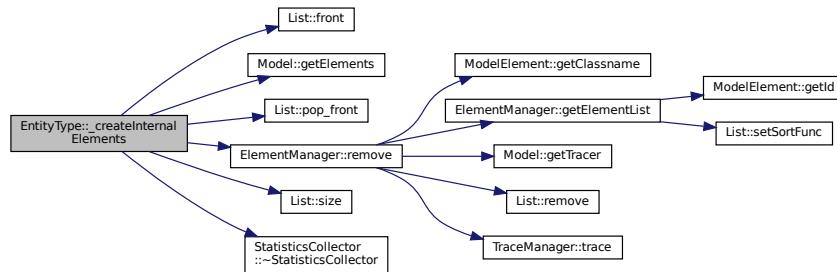
```
00142
00143     if (_reportStatistics) {
00144     } else {
00145         while (_statisticsCollectors->size() > 0) {
00146             _parentModel->getElements()->remove(_statisticsCollectors->front());
```

```

00147     _statisticsCollectors->front () ->~StatisticsCollector () ;
00148     _statisticsCollectors->pop_front () ;
00149 }
00150 }
00151 }
```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [List< T >::front\(\)](#), [Model::getElements\(\)](#), [List< T >::pop_front\(\)](#), [ElementManager::remove\(\)](#), [List< T >::size\(\)](#), and [StatisticsCollector::~StatisticsCollector\(\)](#).

Here is the call graph for this function:



8.29.3.3 `_initBetweenReplications()` void EntityType::_initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

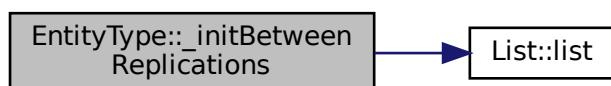
Definition at line 23 of file [EntityType.cpp](#).

```

00023     _initialWaitingCost = 0.0;
00024     _initialVACost = 0.0;
00025     _initialNVACost = 0.0;
00026     _initialOtherCost = 0.0;
00027
00028     for (std::list<StatisticsCollector*>::iterator it = this->_statisticsCollectors->list ()->begin ();
00029         it != this->_statisticsCollectors->list ()->end (); it++)
00030     {
00031         (*it)->getStatistics ()->getCollector ()->clear ();
00032     }
```

References [List< T >::list\(\)](#).

Here is the call graph for this function:



```
8.29.3.4 _loadInstance() bool EntityType::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

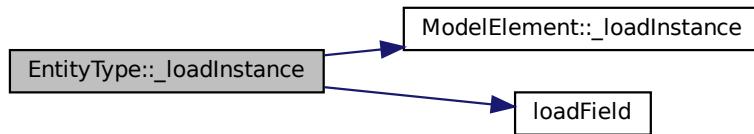
Definition at line 116 of file [EntityType.cpp](#).

```
00116
00117     bool res = ModelElement::_loadInstance(fields);
00118     if (res) {
00119         this->_initialNVACost = std::stod(loadField(fields, "initialNVACost", "0.0"));
00120         this->_initialOtherCost = std::stod(loadField(fields, "initialOtherCost", "0.0"));
00121         this->_initialPicture = (loadField(fields, "initialPicture", "0.0"));
00122         this->_initialVACost = std::stod(loadField(fields, "initialVACost", "0.0"));
00123         this->_initialWaitingCost = std::stod(loadField(fields, "initialWaitingCost", "0.0"));
00124     }
00125     return res;
00126 }
```

References [ModelElement::_loadInstance\(\)](#), and [loadField\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reimplemented from [ModelElement](#).

Definition at line 128 of file [EntityType.cpp](#).

```
00128
00129     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00130     //Util::TypeOf<EntityType>();
00131     if (_initialNVACost != 0.0) fields->emplace("initialNVACost",
00132         std::to_string(this->_initialNVACost));
00133     if (_initialOtherCost != 0.0) fields->emplace("initialOtherCost",
00134         std::to_string(this->_initialOtherCost));
00135     if (_initialPicture != "report") fields->emplace("initialPicture", this->_initialPicture);
00136     if (_initialVACost != 0.0) fields->emplace("initialVACost", std::to_string(this->_initialVACost));
00137     if (_initialWaitingCost != 0.0) fields->emplace("initialWaitingCost",
00138         std::to_string(this->_initialWaitingCost));
```

```
00135     return fields;
00136 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.29.3.6 addGetStatisticsCollector() [StatisticsCollector * EntityType::addGetStatisticsCollector](#)

```
(  
    std::string name )
```

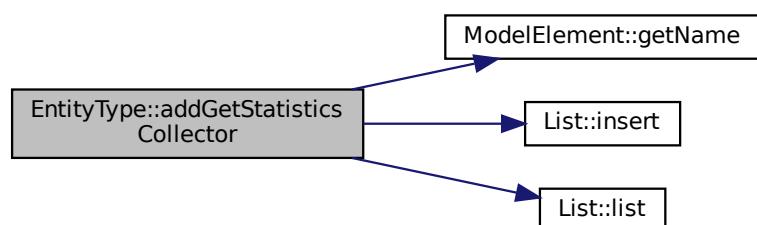
Definition at line 85 of file [EntityType.cpp](#).

```
00085
00086     StatisticsCollector* cstat;
00087     for (std::list<StatisticsCollector*>::iterator it = _statisticsCollectors->list()->begin(); it != _statisticsCollectors->list()->end(); it++) {
00088         cstat = (*it);
00089         if (cstat->getName() == name) {
00090             return cstat;
00091         }
00092     }
00093     // not found. Create it, insert it into the list of cstats, into the model element manager, and then return it
00094     cstat = new StatisticsCollector(_parentModel, name, this);
00095     _statisticsCollectors->insert(cstat); // \todo _statisticsCollectors list is probably redundant
00096     _childrenElements->insert({name, cstat});
00097     //_parentModel->insert(cstat); // unnecessary
00098     return cstat;
00099 }
```

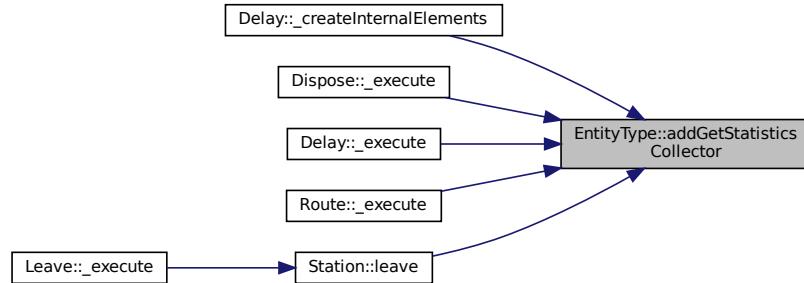
References [ModelElement::_childrenElements](#), [ModelElement::_parentModel](#), [ModelElement::getName\(\)](#), [List< T >::insert\(\)](#), and [List< T >::list\(\)](#).

Referenced by [Delay::_createInternalElements\(\)](#), [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Route::_execute\(\)](#), and [Station::leave\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.29.3.7 GetPluginInformation() `PluginInformation * EntityType::GetPluginInformation () [static]`

Definition at line 101 of file [EntityType.cpp](#).

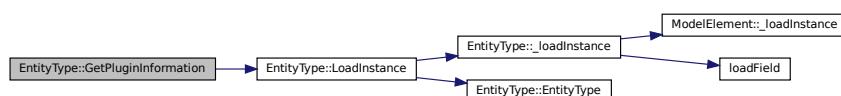
```

00101
00102     PluginInformation* info = new PluginInformation(Util::TypeOf<EntityType>(),
00103         &EntityType::LoadInstance);
00104     return info;
00105 }
```

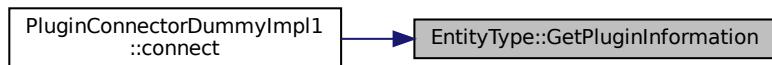
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.29.3.8 initialNVACost() double EntityType::initialNVACost () const

Definition at line 65 of file [EntityType.cpp](#).

```
00065     {  
00066         return _initialNVACost;  
00067     }
```

8.29.3.9 initialOtherCost() double EntityType::initialOtherCost () const

Definition at line 57 of file [EntityType.cpp](#).

```
00057     {  
00058         return _initialOtherCost;  
00059     }
```

8.29.3.10 initialPicture() std::string EntityType::initialPicture () const

Definition at line 81 of file [EntityType.cpp](#).

```
00081     {  
00082         return _initialPicture;  
00083     }
```

8.29.3.11 initialVACost() double EntityType::initialVACost () const

Definition at line 73 of file [EntityType.cpp](#).

```
00073     {  
00074         return _initialVACost;  
00075     }
```

8.29.3.12 initialWaitingCost() double EntityType::initialWaitingCost () const

Definition at line 49 of file [EntityType.cpp](#).

```
00049     {  
00050         return _initialWaitingCost;  
00051     }
```

```
8.29.3.13 LoadInstance() ModelElement * EntityType::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

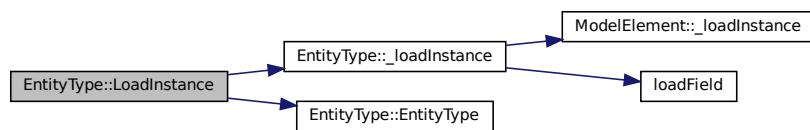
Definition at line 106 of file [EntityType.cpp](#).

```
00106
00107     EntityType* newElement = new EntityType(model);
00108     try {
00109         newElement->_loadInstance(fields);
00110     } catch (const std::exception& e) {
00111     }
00112     return newElement;
00113 }
```

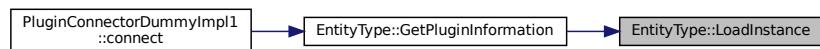
References [_loadInstance\(\)](#), and [EntityType\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 61 of file [EntityType.cpp](#).

```
00061
00062     this->_initialNVACost = _initialNVACost;
00063 }
```

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

Definition at line 53 of file [EntityType.cpp](#).

```
00053
00054     this->_initialOtherCost = _initialOtherCost;
00055 }
```

8.29.3.16 setInitialPicture() void EntityType::setInitialPicture (std::string _initialPicture)

Definition at line 77 of file [EntityType.cpp](#).

```
00077
00078     this->_initialPicture = _initialPicture;
00079 }
```

8.29.3.17 setInitialVACost() void EntityType::setInitialVACost (double _initialVACost)

Definition at line 69 of file [EntityType.cpp](#).

```
00069
00070     this->_initialVACost = _initialVACost;
00071 }
```

8.29.3.18 setInitialWaitingCost() void EntityType::setInitialWaitingCost (double _initialWaitingCost)

Definition at line 45 of file [EntityType.cpp](#).

```
00045
00046     this->_initialWaitingCost = _initialWaitingCost;
00047 }
```

8.29.3.19 show() std::string EntityType::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 40 of file [EntityType.cpp](#).

```
00040
00041     {
00042         return ModelElement::show() +
00043             ",initialPicture=" + this->_initialPicture; // add more...
00043 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



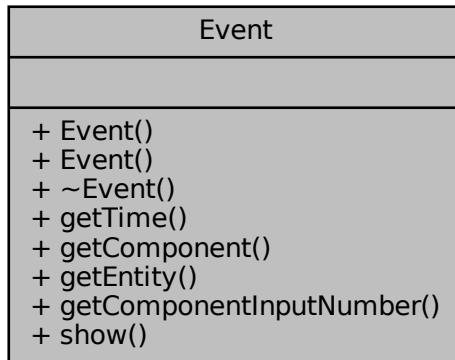
The documentation for this class was generated from the following files:

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

8.30 Event Class Reference

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

Collaboration diagram for Event:



Public Member Functions

- `Event (double time, Entity *entity, ModelComponent *component, unsigned int componentInputNumber=0)`
- `Event (double time, Entity *entity, Connection *connection)`
- `virtual ~Event ()=default`
- `double getTime () const`
- `ModelComponent * getComponent () const`
- `Entity * getEntity () const`
- `unsigned int getComponentInputNumber () const`
- `std::string show ()`

8.30.1 Detailed Description

An instantaneous event, triggered at a certain moment by an entity upon reaching a component. The simulated time advances in discrete points in time and that are the instants that an event is triggered.

Definition at line 28 of file [Event.h](#).

8.30.2 Constructor & Destructor Documentation

8.30.2.1 Event() [1/2] `Event::Event (`

```
    double time,
    Entity * entity,
    ModelComponent * component,
    unsigned int componentInputNumber = 0 )
```

Definition at line 18 of file [Event.cpp](#).

```
00018 {
00019     _time = time;
00020     _entity = entity;
00021     _component = component;
00022     _componentInputNumber = componentInputNumber;
00023 }
```

8.30.2.2 Event() [2/2] `Event::Event (`

```
    double time,
    Entity * entity,
    Connection * connection )
```

Definition at line 25 of file [Event.cpp](#).

```
00025 {
00026     _time = time;
00027     _entity = entity;
00028     _component = connection->first;
00029     _componentInputNumber = connection->second;
00030 }
```

8.30.2.3 ~Event() `virtual Event::~Event () [virtual], [default]`

8.30.3 Member Function Documentation

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

Definition at line 46 of file [Event.cpp](#).

```
00046 {
00047     return _component;
00048 }
```

8.30.3.2 getComponentInputNumber() `unsigned int Event::getComponentInputNumber () const`

Definition at line 38 of file [Event.cpp](#).

```
00038 {
00039     return _componentInputNumber;
00040 }
```

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

Definition at line 50 of file [Event.cpp](#).

```
00050
00051     return _entity;
00052 }
```

Referenced by [BaseConsoleGenesysApplication::onEntityRemoveHandler\(\)](#).

Here is the caller graph for this function:

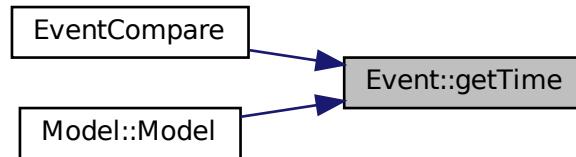
**8.30.3.4 getTime()** `double Event::getTime () const`

Definition at line 42 of file [Event.cpp](#).

```
00042
00043     return _time;
00044 }
```

Referenced by [EventCompare\(\)](#), and [Model::Model\(\)](#).

Here is the caller graph for this function:



8.30.3.5 `show()` `std::string Event::show()`

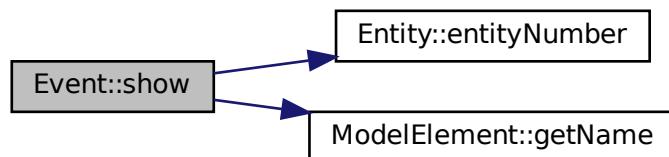
Definition at line 32 of file [Event.cpp](#).

```
00032     {
00033         return "time=" + std::to_string(_time) +
00034             ",entity=" + std::to_string(_entity->entityNumber()) +
00035             ",comp=\" " + _component->getName() + "\"; //+std::to_string(_component->getId())+"";
00036 }
```

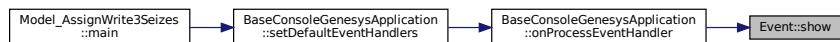
References [Entity::entityNumber\(\)](#), and [ModelElement::getName\(\)](#).

Referenced by [BaseConsoleGenesysApplication::onProcessEventHandler\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



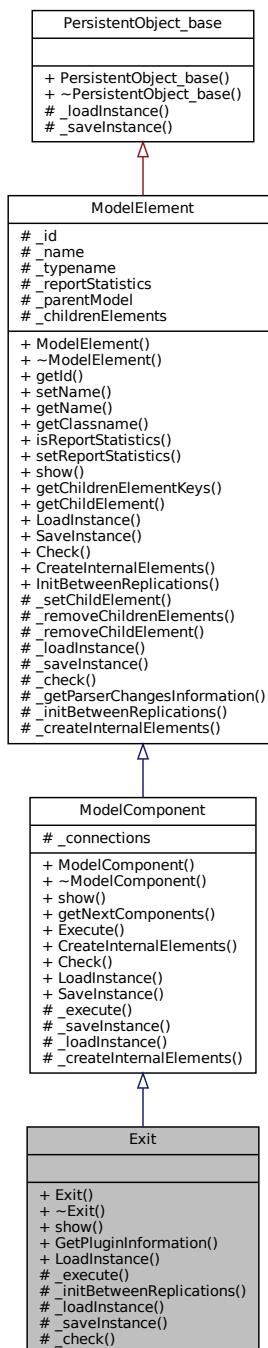
The documentation for this class was generated from the following files:

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

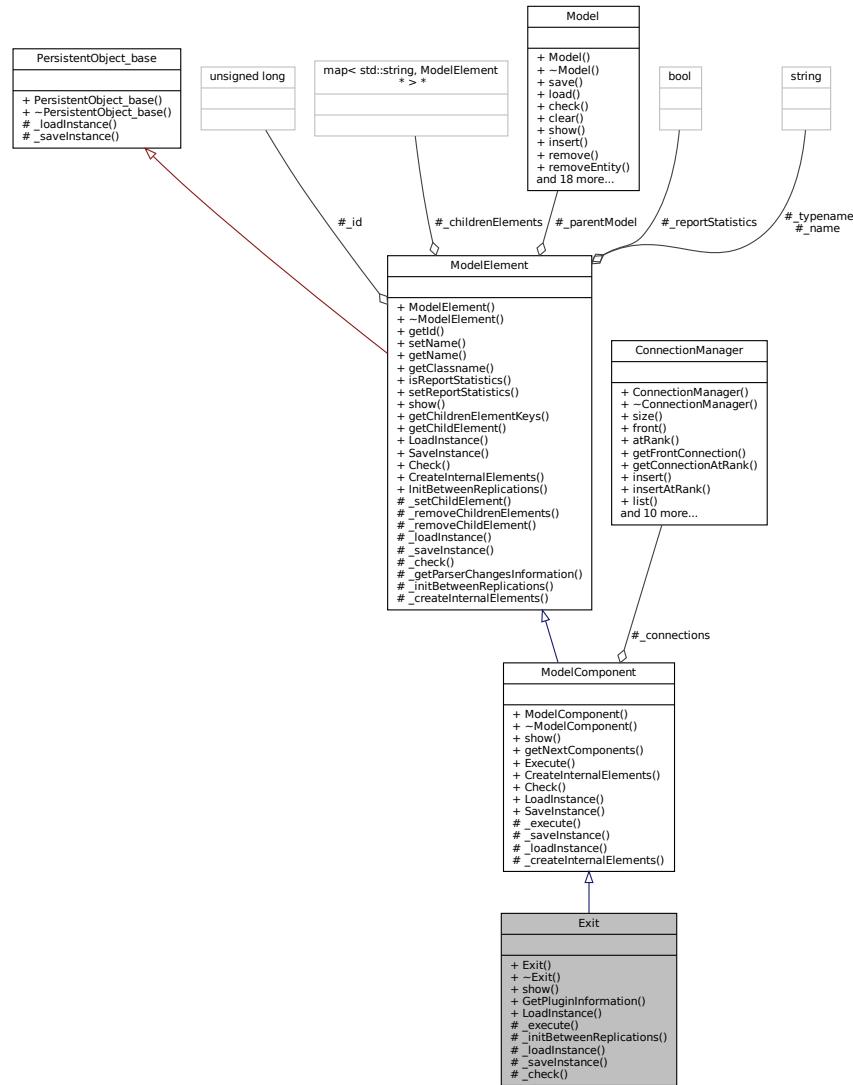
8.31 Exit Class Reference

```
#include <Exit.h>
```

Inheritance diagram for Exit:



Collaboration diagram for Exit:



Public Member Functions

- **Exit (Model *model, std::string name="")**
- virtual **~Exit ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.31.1 Detailed Description

Exit module DESCRIPTION The [Exit](#) module releases the entity's cells on the specified conveyor. If another entity is waiting in queue for the conveyor at the same station when the cells are released, it will then access the conveyor.
TYPICAL USES Cases exit a conveyor for packing Bad parts are removed from the conveyor and disposed Passengers remove luggage from the baggage claim conveyor
PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor on which the entity will exit. If left blank, the previously accessed conveyor is assumed.

of Cells Number of contiguous conveyor cells the entity will relinquish.

Definition at line 37 of file [Exit.h](#).

8.31.2 Constructor & Destructor Documentation

```
8.31.2.1 Exit() Exit::Exit (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Exit.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Exit>(), name) {
00019 }
```

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

Here is the caller graph for this function:



```
8.31.2.2 ~Exit() virtual Exit::~Exit ( ) [virtual], [default]
```

8.31.3 Member Function Documentation

8.31.3.1 `_check()` `bool Exit::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Exit.cpp](#).

```
00057 {  
00058     bool resultAll = true;  
00059     //...  
00060     return resultAll;  
00061 }
```

8.31.3.2 `_execute()` `void Exit::_execute (Entity * entity) [protected], [virtual]`

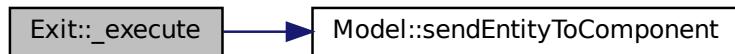
Implements [ModelComponent](#).

Definition at line 35 of file [Exit.cpp](#).

```
00035 {  
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);  
00038 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.31.3.3 `_initBetweenReplications()` `void Exit::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Exit.cpp](#).

```
00048 {  
00049 }
```

```
8.31.3.4 _loadInstance() bool Exit::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

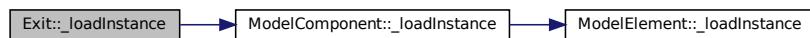
Definition at line 40 of file [Exit.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

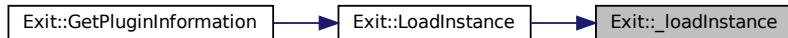
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.31.3.5 _saveInstance() std::map< std::string, std::string > * Exit::_saveInstance ( ) [protected], [virtual]
```

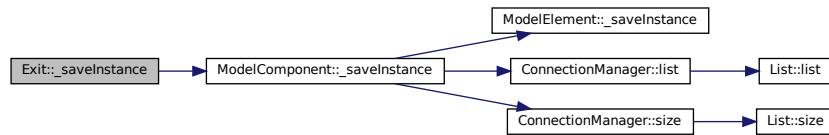
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Exit.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.31.3.6 GetPluginInformation() `PluginInformation * Exit::GetPluginInformation () [static]`

Definition at line 63 of file `Exit.cpp`.

```
00063
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Exit>(), &Exit::LoadInstance);
00065     // ...
00066     return info;
00067 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.31.3.7 LoadInstance() `ModelComponent * Exit::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

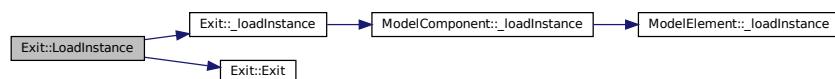
Definition at line 25 of file `Exit.cpp`.

```
00025
00026     Exit* newComponent = new Exit(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031     return newComponent;
00032 }
00033 }
```

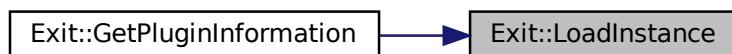
References `_loadInstance()`, and `Exit()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.31.3.8 show() std::string Exit::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Exit.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



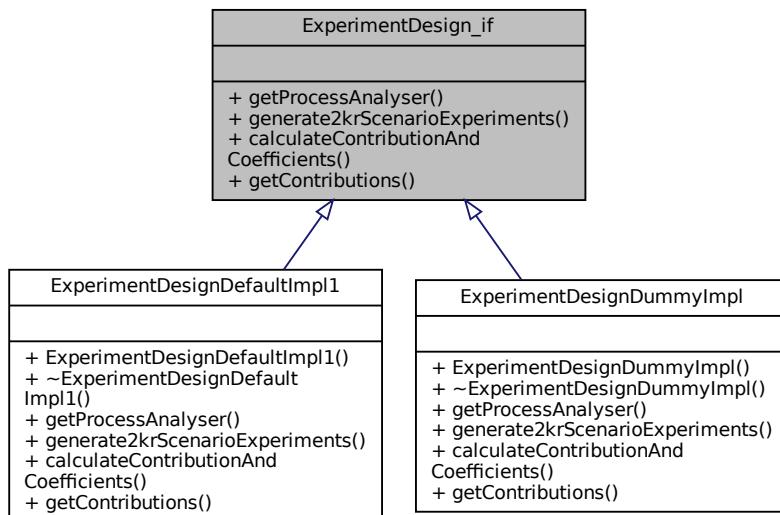
The documentation for this class was generated from the following files:

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

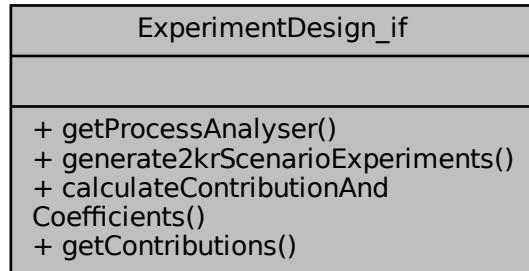
8.32 ExperimentDesign_if Class Reference

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

Inheritance diagram for ExperimentDesign_if:



Collaboration diagram for ExperimentDesign_if:



Public Member Functions

- virtual `ExperimentManager_if * getProcessAnalyser () const =0`
- virtual bool `generate2krScenarioExperiments ()=0`
- virtual bool `calculateContributionAndCoefficients ()=0`
- virtual std::list< `FactorOrInteractionContribution * > * getContributions () const =0`

8.32.1 Detailed Description

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

Definition at line 23 of file [ExperimentDesign_if.h](#).

8.32.2 Member Function Documentation

8.32.2.1 calculateContributionAndCoefficients() `virtual bool ExperimentDesign_if::calculate←ContributionAndCoefficients () [pure virtual]`

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

8.32.2.2 generate2krScenarioExperiments() `virtual bool ExperimentDesign_if::generate2krScenario←Experiments () [pure virtual]`

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

```
8.32.2.3 getContributions() virtual std::list<FactorOrInteractionContribution*>* Experiment->  
Design_if::getContributions ( ) const [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

```
8.32.2.4 getProcessAnalyser() virtual ExperimentManager_if* ExperimentDesign_if::getProcess->  
Analyser ( ) const [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

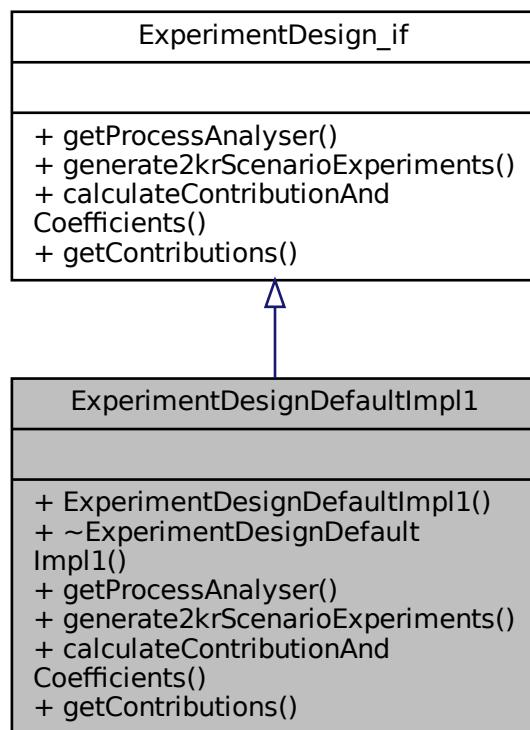
The documentation for this class was generated from the following file:

- [ExperimentDesign_if.h](#)

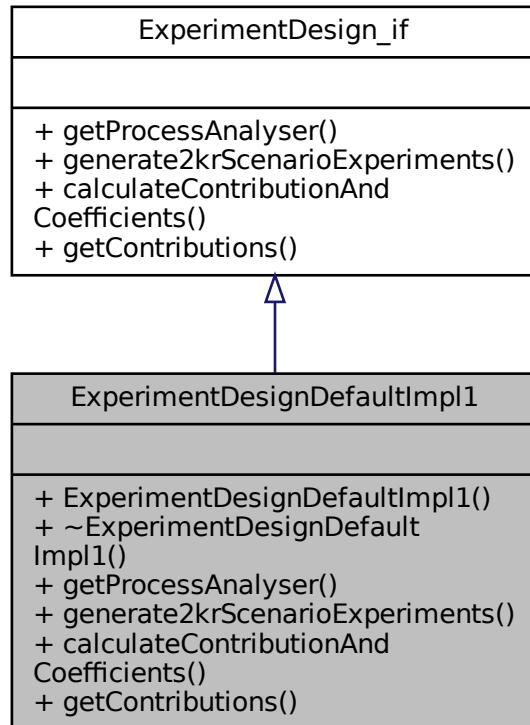
8.33 ExperimentDesignDefaultImpl1 Class Reference

```
#include <ExperimentDesignDefaultImpl1.h>
```

Inheritance diagram for ExperimentDesignDefaultImpl1:



Collaboration diagram for ExperimentDesignDefaultImpl1:



Public Member Functions

- `ExperimentDesignDefaultImpl1 ()`
- virtual `~ExperimentDesignDefaultImpl1 ()=default`
- virtual `ExperimentManager_if * getProcessAnalyser () const`
- virtual `bool generate2krScenarioExperiments ()`
- virtual `bool calculateContributionAndCoefficients ()`
- virtual `std::list< FactorOrInteractionContribution * > * getContributions () const`

8.33.1 Detailed Description

Definition at line 19 of file [ExperimentDesignDefaultImpl1.h](#).

8.33.2 Constructor & Destructor Documentation

8.33.2.1 ExperimentDesignDefaultImpl1() ExperimentDesignDefaultImpl1::ExperimentDesignDefault<→
Impl1 ()

Definition at line 16 of file [ExperimentDesignDefaultImpl1.cpp](#).

```
00016 {  
00017 }
```

8.33.2.2 ~ExperimentDesignDefaultImpl1() virtual ExperimentDesignDefaultImpl1::~ExperimentDesignDefaultImpl1 () [virtual], [default]

8.33.3 Member Function Documentation

8.33.3.1 calculateContributionAndCoefficients() bool ExperimentDesignDefaultImpl1::calculateContributionAndCoefficients () [virtual]

Implements [ExperimentDesign_if](#).

Definition at line 27 of file [ExperimentDesignDefaultImpl1.cpp](#).

```
00027 {  
00028     return true;  
00029 }
```

8.33.3.2 generate2krScenarioExperiments() bool ExperimentDesignDefaultImpl1::generate2krScenarioExperiments () [virtual]

Implements [ExperimentDesign_if](#).

Definition at line 23 of file [ExperimentDesignDefaultImpl1.cpp](#).

```
00023 {  
00024     return true;  
00025 }
```

8.33.3.3 getContributions() std::list< FactorOrInteractionContribution * > * ExperimentDesignDefaultImpl1::getContributions () const [virtual]

Implements [ExperimentDesign_if](#).

Definition at line 19 of file [ExperimentDesignDefaultImpl1.cpp](#).

```
00019 {  
00020     return _contributions;  
00021 }
```

8.33.3.4 getProcessAnalyser() `ExperimentManager_if * ExperimentDesignDefaultImpl1::getProcessAnalyser() const [virtual]`

Implements `ExperimentDesign_if`.

Definition at line 31 of file `ExperimentDesignDefaultImpl1.cpp`.

```
00031
00032     return _processAnalyser;
00033 }
```

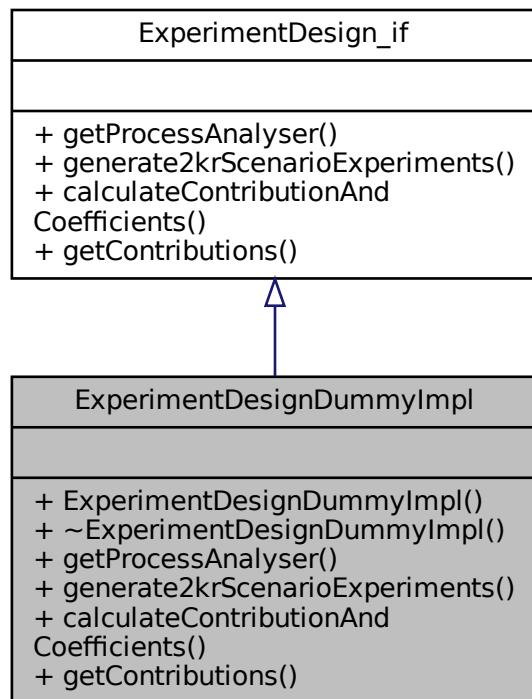
The documentation for this class was generated from the following files:

- `ExperimentDesignDefaultImpl1.h`
- `ExperimentDesignDefaultImpl1.cpp`

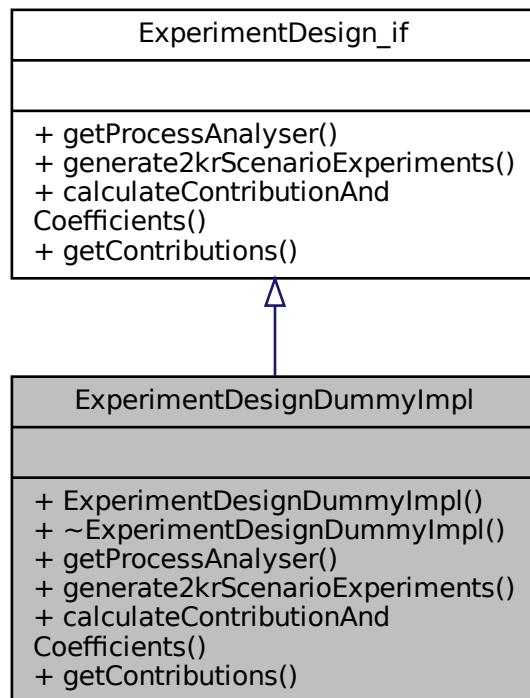
8.34 ExperimentDesignDummyImpl Class Reference

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

Inheritance diagram for `ExperimentDesignDummyImpl`:



Collaboration diagram for ExperimentDesignDummyImpl:



Public Member Functions

- [ExperimentDesignDummyImpl \(\)](#)
- virtual [~ExperimentDesignDummyImpl \(\)=default](#)
- virtual [ExperimentManager_if * getProcessAnalyser \(\) const](#)
- virtual bool [generate2krScenarioExperiments \(\)](#)
- virtual bool [calculateContributionAndCoefficients \(\)](#)
- virtual std::list< [FactorOrInteractionContribution](#) * > * [getContributions \(\) const](#)

8.34.1 Detailed Description

Definition at line 22 of file [ExperimentDesignDummyImpl.h](#).

8.34.2 Constructor & Destructor Documentation

8.34.2.1 ExperimentDesignDummyImpl() [ExperimentDesignDummyImpl::ExperimentDesignDummyImpl \(\)](#)

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

```

00017
00018 } 
```

8.34.2.2 ~ExperimentDesignDummyImpl() virtual ExperimentDesignDummyImpl::~ExperimentDesignDummyImpl () [virtual], [default]

8.34.3 Member Function Documentation

8.34.3.1 calculateContributionAndCoefficients() bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients () [virtual]

Implements [ExperimentDesign_if](#).

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

```
00028
00029     return true;
00030 }
```

8.34.3.2 generate2krScenarioExperiments() bool ExperimentDesignDummyImpl::generate2krScenarioExperiments () [virtual]

Implements [ExperimentDesign_if](#).

Definition at line 24 of file [ExperimentDesignDummyImpl.cpp](#).

```
00024
00025     return true;
00026 }
```

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

Implements [ExperimentDesign_if](#).

Definition at line 20 of file [ExperimentDesignDummyImpl.cpp](#).

```
00020
00021     return _contributions;
00022 }
```

8.34.3.4 getProcessAnalyser() [ExperimentManager_if](#) * ExperimentDesignDummyImpl::getProcessAnalyser () const [virtual]

Implements [ExperimentDesign_if](#).

Definition at line 32 of file [ExperimentDesignDummyImpl.cpp](#).

```
00032
00033     return _processAnalyser;
00034 }
```

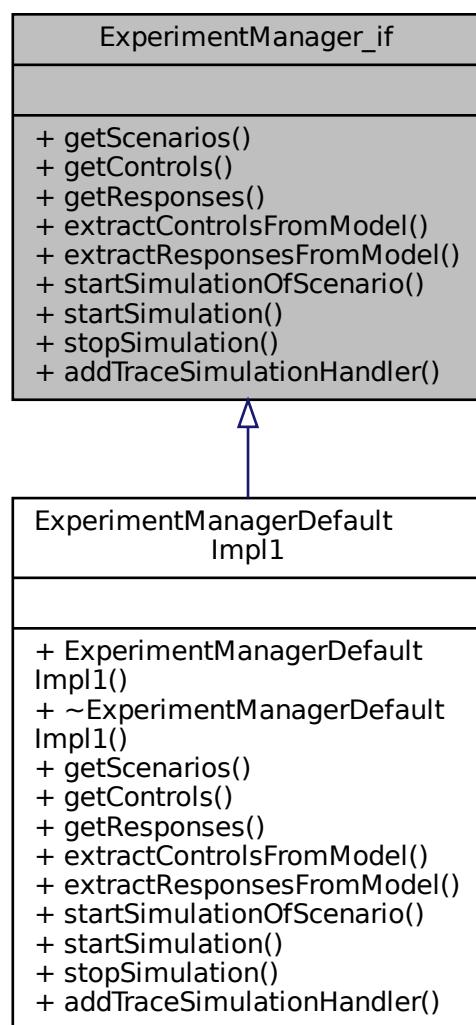
The documentation for this class was generated from the following files:

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

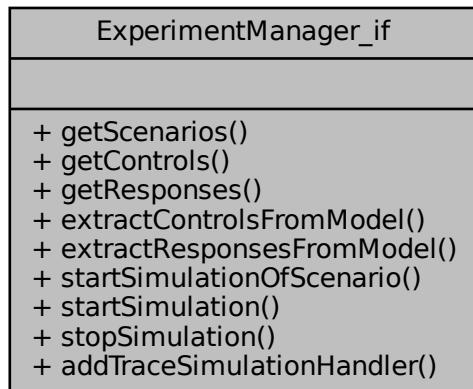
8.35 ExperimentManager_if Class Reference

```
#include <ExperimentManager_if.h>
```

Inheritance diagram for ExperimentManager_if:



Collaboration diagram for ExperimentManager_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`

8.35.1 Detailed Description

The experiment manager allows to extract controls and responses from a model, include some of them as controls and responses for a set of scenarios to be simulated

Definition at line 26 of file [ExperimentManager_if.h](#).

8.35.2 Member Function Documentation

8.35.2.1 addTraceSimulationHandler() `virtual void ExperimentManager_if::addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener) [pure virtual]`

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.2 extractControlsFromModel() virtual List<SimulationControl*>* ExperimentManager_if::extractControlsFromModel ( std::string modelFilename ) const [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

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

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.4 getControls() virtual List<SimulationControl*>* ExperimentManager_if::getControls ( ) const [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.5 getResponses() virtual List<SimulationResponse*>* ExperimentManager_if::getResponses ( ) const [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.6 getScenarios() virtual List<SimulationScenario*>* ExperimentManager_if::getScenarios ( ) const [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.7 startSimulation() virtual void ExperimentManager_if::startSimulation ( ) [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

```
8.35.2.8 startSimulationOfScenario() virtual void ExperimentManager_if::startSimulationOfScenario ( SimulationScenario * scenario ) [pure virtual]
```

Implemented in [ExperimentManagerDefaultImpl1](#).

8.35.2.9 stopSimulation() virtual void ExperimentManager_if::stopSimulation () [pure virtual]

Implemented in [ExperimentManagerDefaultImpl1](#).

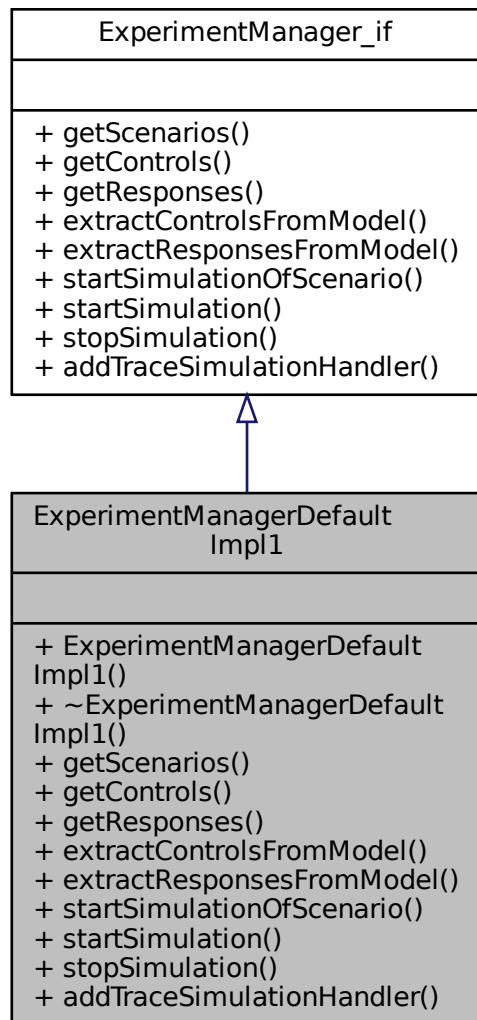
The documentation for this class was generated from the following file:

- [ExperimentManager_if.h](#)

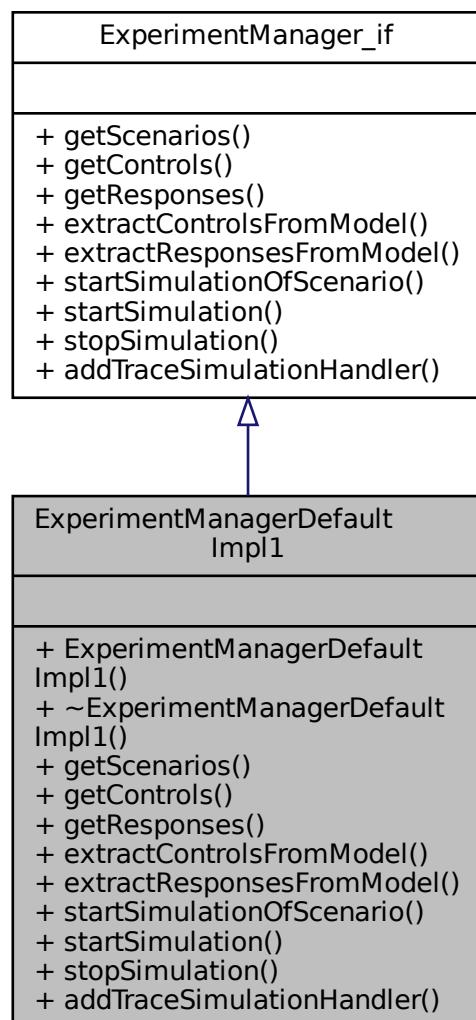
8.36 ExperimentManagerDefaultImpl1 Class Reference

```
#include <ExperimentManagerDefaultImpl1.h>
```

Inheritance diagram for ExperimentManagerDefaultImpl1:



Collaboration diagram for ExperimentManagerDefaultImpl1:



Public Member Functions

- `ExperimentManagerDefaultImpl1 ()`
- virtual `~ExperimentManagerDefaultImpl1 ()=default`
- virtual `List< SimulationScenario * > * getScenarios () const`
- virtual `List< SimulationControl * > * getControls () const`
- virtual `List< SimulationResponse * > * getResponses () const`
- virtual `List< SimulationControl * > * extractControlsFromModel (std::string modelFilename) const`
- virtual `List< SimulationResponse * > * extractResponsesFromModel (std::string modelFilename) const`
- virtual void `startSimulationOfScenario (SimulationScenario *scenario)`
- virtual void `startSimulation ()`
- virtual void `stopSimulation ()`
- virtual void `addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)`

8.36.1 Detailed Description

Definition at line 22 of file [ExperimentManagerDefaultImpl1.h](#).

8.36.2 Constructor & Destructor Documentation

8.36.2.1 ExperimentManagerDefaultImpl1() `ExperimentManagerDefaultImpl1::ExperimentManagerDefaultImpl1()`

Definition at line 16 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00016
00017 }
```

8.36.2.2 ~ExperimentManagerDefaultImpl1() `virtual ExperimentManagerDefaultImpl1::~ExperimentManagerDefaultImpl1() [virtual], [default]`

8.36.3 Member Function Documentation

8.36.3.1 addTraceSimulationHandler() `void ExperimentManagerDefaultImpl1::addTraceSimulationHandler(traceSimulationProcessListener traceSimulationProcessListener) [virtual]`

Implements [ExperimentManager_if](#).

Definition at line 51 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00051
00052 }
```

8.36.3.2 extractControlsFromModel() `List< SimulationControl * > * ExperimentManagerDefaultImpl1::extractControlsFromModel(std::string modelFilename) const [virtual]`

Implements [ExperimentManager_if](#).

Definition at line 31 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00031
00032     {
00033 // \todo: implement
00034 }
```

```
8.36.3.3 extractResponsesFromModel() List< SimulationResponse * > * ExperimentManagerDefaultImpl1::extractResponsesFromModel( std::string modelFilename ) const [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 35 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00035 {  
00036     // \todo: implement  
00037 }
```

```
8.36.3.4 getControls() List< SimulationControl * > * ExperimentManagerDefaultImpl1::getControls( ) const [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 23 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00023 {  
00024     return _controls;  
00025 }
```

```
8.36.3.5 getResponses() List< SimulationResponse * > * ExperimentManagerDefaultImpl1::getResponses( ) const [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 27 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00027 {  
00028     // \todo: implement  
00029 }
```

```
8.36.3.6 getScenarios() List< SimulationScenario * > * ExperimentManagerDefaultImpl1::getScenarios( ) const [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 19 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00019 {  
00020     // \todo: implement  
00021 }
```

```
8.36.3.7 startSimulation() void ExperimentManagerDefaultImpl1::startSimulation( ) [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 43 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00043 {  
00044     // \todo: implement  
00045 }
```

```
8.36.3.8 startSimulationOfScenario() void ExperimentManagerDefaultImpl1::startSimulationOf<-
Scenario (
    SimulationScenario * scenario ) [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 39 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00039
00040     // \todo: implement
00041 }
```

```
8.36.3.9 stopSimulation() void ExperimentManagerDefaultImpl1::stopSimulation () [virtual]
```

Implements [ExperimentManager_if](#).

Definition at line 47 of file [ExperimentManagerDefaultImpl1.cpp](#).

```
00047
00048     // \todo: implement
00049 }
```

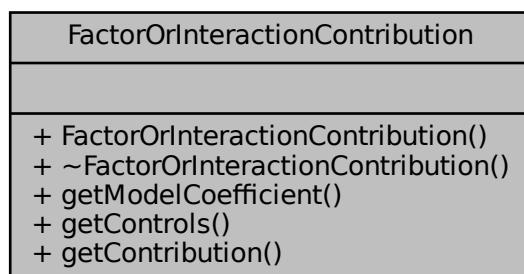
The documentation for this class was generated from the following files:

- [ExperimentManagerDefaultImpl1.h](#)
- [ExperimentManagerDefaultImpl1.cpp](#)

8.37 FactorOrInteractionContribution Class Reference

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

Collaboration diagram for FactorOrInteractionContribution:



Public Member Functions

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

8.37.1 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](#).

Definition at line 23 of file [FactorOrInteractionContribution.h](#).

8.37.2 Constructor & Destructor Documentation

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

Definition at line 16 of file [FactorOrInteractionContribution.cpp](#).

```
00016
00017     _contribution = contribution;
00018     _modelCoefficient = modelCoefficient;
00019     _controls = controls;
00020 }
```

```
8.37.2.2 ~FactorOrInteractionContribution() FactorOrInteractionContribution::~FactorOrInteractionContribution (
    )
```

8.37.3 Member Function Documentation

```
8.37.3.1 getContribution() double FactorOrInteractionContribution::getContribution ( ) const
```

Definition at line 30 of file [FactorOrInteractionContribution.cpp](#).

```
00030
00031     return _contribution;
00032 }
```

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

Definition at line 26 of file [FactorOrInteractionContribution.cpp](#).

```
00026
00027     return _controls;
00028 }
```

8.37.3.3 getModelCoefficient() double FactorOrInteractionContribution::getModelCoefficient ()
const

Definition at line 22 of file [FactorOrInteractionContribution.cpp](#).

```
00022
00023     return _modelCoefficient;
00024 }
```

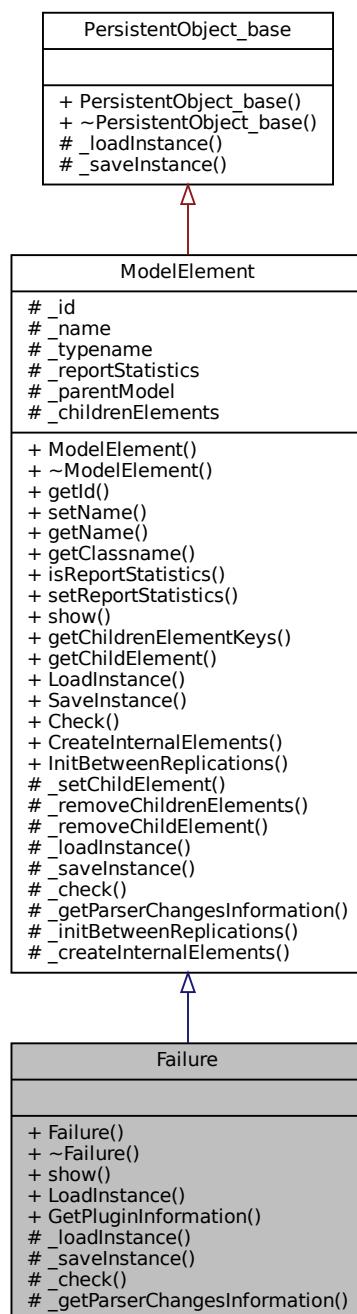
The documentation for this class was generated from the following files:

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

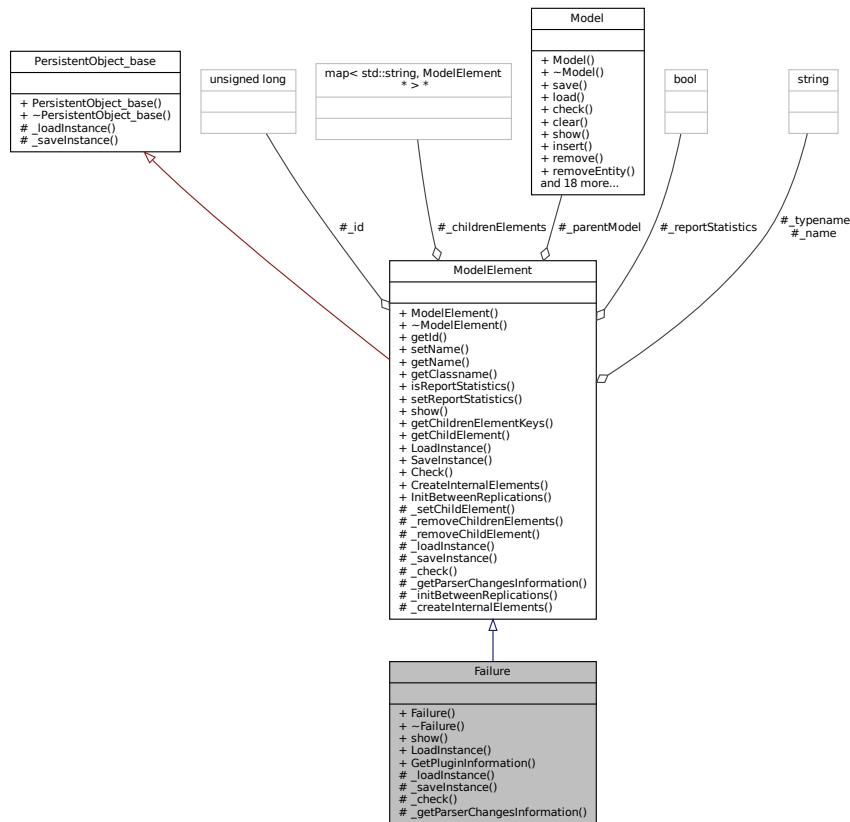
8.38 Failure Class Reference

```
#include <Failure.h>
```

Inheritance diagram for Failure:



Collaboration diagram for Failure:



Public Member Functions

- **Failure (Model *model, std::string name="")**
- virtual **~Failure ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**
- static **PluginInformation * GetPluginInformation ()**

Protected Member Functions

- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**
- virtual **ParserChangesInformation * _getParserChangesInformation ()**

Additional Inherited Members

8.38.1 Detailed Description

Failure module DESCRIPTION The **Failure** module is designed for use with resources. When a failure occurs, the entire resource (regardless of its capacity) is failed. Failures are designed to be used with single-capacity resources or with multiple-capacity resources whose individual resource units all fail at the same time. TYPICAL USES Breakdown information for a machine Cash register tape refill every “x” customers Random computer shutdowns or restarts PROMPTS Recordset Name of the recordset in the specified file from which to read values. This field is available only if you specify a **File** Name with a file access type, path, and recordset. Arena uses the Rows and Columns properties to determine the amount of data to read from the recordset. A recordset is required for all file types except .xml. The recordset size must be equal to or greater than the number of rows and columns specified for the expression. Expression Values Lists the value or values of the expression. This property is not available if you specify a **File** Name from which to read expression values. Expression Value Expression value associated with the expression name. Prompt Description Name The name of the failure associated with one or more resources. Type Determines if the failure is time-based or count-based. Count Defines the number of resource releases for count-based failures. Valid when the Type is Count. Up Time Defines the time between failures for time-based failures. Valid when the Type is Time. Up Time Units Time units for the time between failures (Up Time) for timebased failures. Down Time Defines the duration of the failure. Down Time Units Time units for the duration of the failure (Down Time). Uptime in this State only Defines the state that should be considered for the time between failures (only for time-based failures). If state is not specified, then all states are considered (that is, the time between failures does not depend on the time spent in a specific state, but rather on the total simulation time). For example, you might want to define a failure to be based only on the state Busy, and therefore, the time between downtimes would be based on the amount of time that a resource is busy, not simulated clock time.

Definition at line 67 of file [Failure.h](#).

8.38.2 Constructor & Destructor Documentation

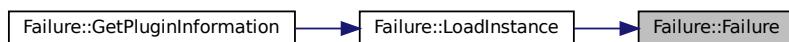
```
8.38.2.1 Failure() Failure::Failure (
    Model * model,
    std::string name = "" )
```

Definition at line 16 of file [Failure.cpp](#).

```
00016 : ModelElement(model, Util::TypeOf<Failure>(), name) {
00017 }
```

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

Here is the caller graph for this function:



```
8.38.2.2 ~Failure() virtual Failure::~Failure ( ) [virtual], [default]
```

8.38.3 Member Function Documentation

8.38.3.1 `_check()` `bool Failure::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Failure.cpp](#).

```
00058
00059     bool resultAll = true;
00060     // resultAll |= ...
00061     return resultAll;
00062 }
```

8.38.3.2 `_getParserChangesInformation()` `ParserChangesInformation * Failure::_getParserChangesInformation () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 64 of file [Failure.cpp](#).

```
00064
00065     ParserChangesInformation* changes = new ParserChangesInformation();
00066     //changes->getProductionToAdd()->insert(...);
00067     //changes->getTokensToAdd()->insert(...);
00068     return changes;
00069 }
```

8.38.3.3 `_loadInstance()` `bool Failure::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 39 of file [Failure.cpp](#).

```
00039
00040     bool res = ModelElement::_loadInstance(fields);
00041     if (res) {
00042         try {
00043             //this->_attributeName = (*fields->find("attributeName")).second;
00044             //this->_orderRule = static_cast<OrderRule>
00045             (std::stoi((*fields->find("orderRule")).second));
00046         } catch (...) {
00047         }
00048     }
00049 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.38.3.4 `_saveInstance()` `std::map< std::string, std::string > * Failure::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

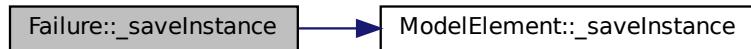
Definition at line 51 of file [Failure.cpp](#).

```

00051
00052     std::map<std::string, std::string>* fields = ModelElement::_saveInstance ();
00053     //Util::TypeOf<Failure>());
00054     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055     //fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00056     return fields;
00056 }
  
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.38.3.5 `GetPluginInformation()` `PluginInformation * Failure::GetPluginInformation () [static]`

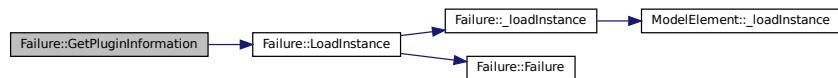
Definition at line 24 of file [Failure.cpp](#).

```

00024
00025     PluginInformation* info = new PluginInformation(Util::TypeOf<Failure>(), &Failure::LoadInstance);
00026     return info;
00027 }
  
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.38.3.6 LoadInstance() `ModelElement * Failure::LoadInstance (`

```
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

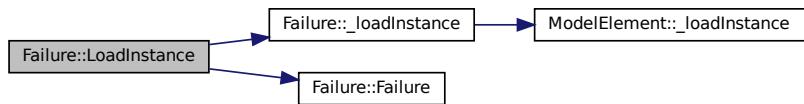
Definition at line 29 of file [Failure.cpp](#).

```
00029
00030     Failure* newElement = new Failure(model);
00031     try {
00032         newElement->_loadInstance(fields);
00033     } catch (const std::exception& e) {
00034
00035     }
00036     return newElement;
00037 }
```

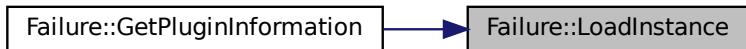
References [_loadInstance\(\)](#), and [Failure\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.38.3.7 show() `std::string Failure::show () [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 19 of file [Failure.cpp](#).

```
00019
00020     {
00021         return ModelElement::show() +
00022         "";
00023 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



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

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

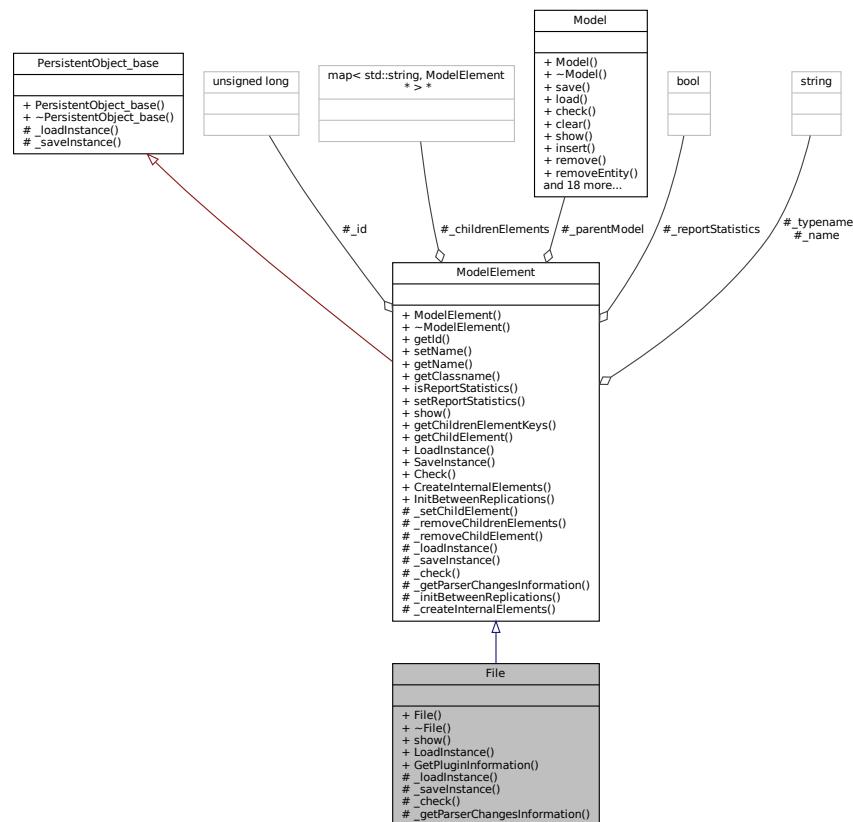
8.39 File Class Reference

```
#include <File.h>
```

Inheritance diagram for File:



Collaboration diagram for File:



Public Member Functions

- **File** (*Model* *model, std::string name="")
 - virtual ~**File** ()=default
 - virtual std::string **show** ()

Static Public Member Functions

- static ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)
 - static PluginInformation * GetPluginInformation ()

Protected Member Functions

- virtual bool _loadInstance (std::map< std::string, std::string > *fields)
 - virtual std::map< std::string, std::string > * _saveInstance ()
 - virtual bool _check (std::string *errorMessage)
 - virtual ParserChangesInformation * getParserChangesInformation ()

Additional Inherited Members

8.39.1 Detailed Description

File module DESCRIPTION Use the [File](#) module to access external files for the [ReadWrite](#) module, [Variable](#) module, and [Expression](#) module. The [File](#) module identifies the system file name and defines the access type and operational characteristics of the file. TYPICAL USES [File](#) containing predefined airline flight data [File](#) specifying customer order times and relevant information [File](#) to write user model configuration data from menu input PR← OMPTS Prompt Description Name The name of the file whose characteristics are being defined. This name must be unique. [Access](#) Type The file type. Operating System [File](#) Name Name of the actual file that is being read from or to which it is being written. Connecting String Connection string used to open ADO connection to the data source. Structure [File](#) structure, which can be unformatted, free format, or a specific C or FORTRAN format. End of [File](#) Action Type of action to occur if an end of file condition is reached. Initialize Option Action to be taken on file at beginning of each simulation replication. Comment Character Character indicating comment record. Recordset Name Name used to identify the recordset in the Expression, [ReadWrite](#), and [Variable](#) modules. This name must be unique within the file. This field is available for Microsoft Excel, Microsoft Excel 2007, Microsoft [Access](#), Microsoft [Access](#) 2007, and ActiveX Data Objects files. CommandText Text of the command that will be used to open the recordset (for example, SQL statement, procedure name, table name.) This field is available for ActiveX Data Object files only. CommandType Type of command entered in the CommandText. Named Range The named range in the Excel workbook to which the recordset refers. Table Name The name of the table in the [Access](#) database to which the recordset refers.

Definition at line 64 of file [File.h](#).

8.39.2 Constructor & Destructor Documentation

8.39.2.1 File() [File](#)::File (

```
Model * model,
std::string name = "")
```

Definition at line 16 of file [File.cpp](#).

```
00016 : ModelElement(model, Util::TypeOf<File>(), name) {
00017 //elems = elems;
00018 }
```

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

Here is the caller graph for this function:



8.39.2.2 ~File() virtual [File](#)::~File () [virtual], [default]

8.39.3 Member Function Documentation

8.39.3.1 `_check()` `bool File::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 59 of file [File.cpp](#).

```
00059
00060     bool resultAll = true;
00061     // resultAll |= ...
00062     return resultAll;
00063 }
```

8.39.3.2 `_getParserChangesInformation()` `ParserChangesInformation * File::_getParserChangesInformation () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 65 of file [File.cpp](#).

```
00065
00066     ParserChangesInformation* changes = new ParserChangesInformation();
00067     //changes->getProductionToAdd()->insert(...);
00068     //changes->getTokensToAdd()->insert(...);
00069     return changes;
00070 }
```

8.39.3.3 `_loadInstance()` `bool File::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 40 of file [File.cpp](#).

```
00040
00041     bool res = ModelElement::_loadInstance(fields);
00042     if (res) {
00043         try {
00044             //this->_attributeName = (*fields->find("attributeName")).second;
00045             //this->_orderRule = static_cast<OrderRule>
00046             (std::stoi((*fields->find("orderRule")).second));
00047         } catch (...) {
00048     }
00049     return res;
00050 }
```

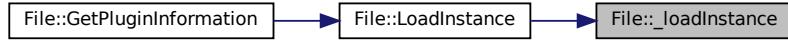
References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.39.3.4 _saveInstance() `std::map< std::string, std::string > * File::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 52 of file [File.cpp](#).

```

00052
00053     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00054     //Util::TypeOf<File>());
00055     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00056     //fields->emplace("attributeName", "\""+this->_attributeName+"\"");
00057     return fields;
00058 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.39.3.5 GetPluginInformation() `PluginInformation * File::GetPluginInformation () [static]`

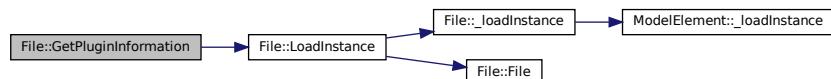
Definition at line 25 of file [File.cpp](#).

```

00025
00026     PluginInformation* info = new PluginInformation(Util::TypeOf<File>(), &File::LoadInstance);
00027     return info;
00028 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



```
8.39.3.6 LoadInstance() ModelElement * File::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

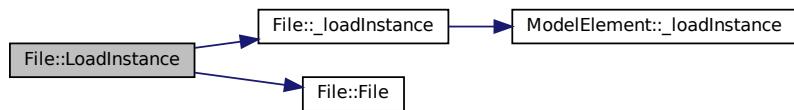
Definition at line 30 of file [File.cpp](#).

```
00030
00031     File* newElement = new File(model);
00032     try {
00033         newElement->_loadInstance(fields);
00034     } catch (const std::exception& e) {
00035
00036     }
00037     return newElement;
00038 }
```

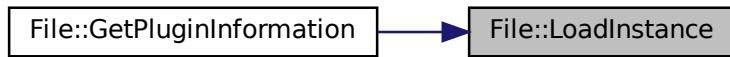
References [_loadInstance\(\)](#), and [File\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.39.3.7 show() std::string File::show () [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 20 of file [File.cpp](#).

```
00020
00021     return ModelElement::show() +
00022         "";
00023 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



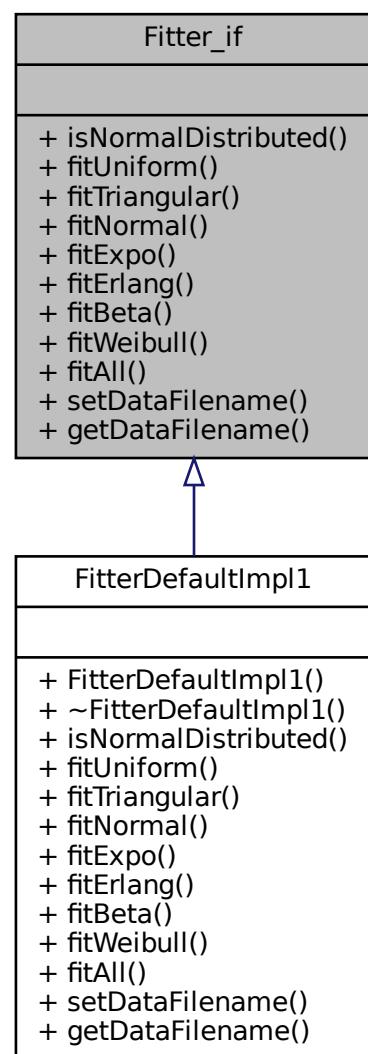
The documentation for this class was generated from the following files:

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

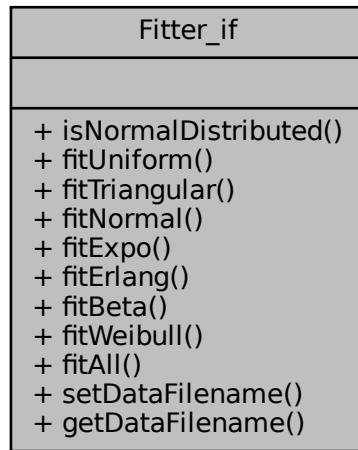
8.40 Fitter_if Class Reference

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

Inheritance diagram for Fitter_if:



Collaboration diagram for Fitter_if:



Public Member Functions

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

8.40.1 Detailed Description

Definition at line 19 of file [Fitter_if.h](#).

8.40.2 Member Function Documentation

8.40.2.1 `fitAll()` virtual void Fitter_if::fitAll (

```

        double * sqrerror,
        std::string * name ) [pure virtual]
  
```

Implemented in [FitterDefaultImpl1](#).

```
8.40.2.2 fitBeta() virtual void Fitter_if::fitBeta (
    double * sqrerror,
    double * alpha,
    double * beta,
    double * infLimit,
    double * supLimit ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

```
8.40.2.3 fitErlang() virtual void Fitter_if::fitErlang (
    double * sqrerror,
    double * avg,
    double * m ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

```
8.40.2.4 fitExpo() virtual void Fitter_if::fitExpo (
    double * sqrerror,
    double * avg1 ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

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

Implemented in [FitterDefaultImpl1](#).

```
8.40.2.11 setDataFilename() virtual void Fitter_if::setDataFilename (  
    std::string datafilename ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

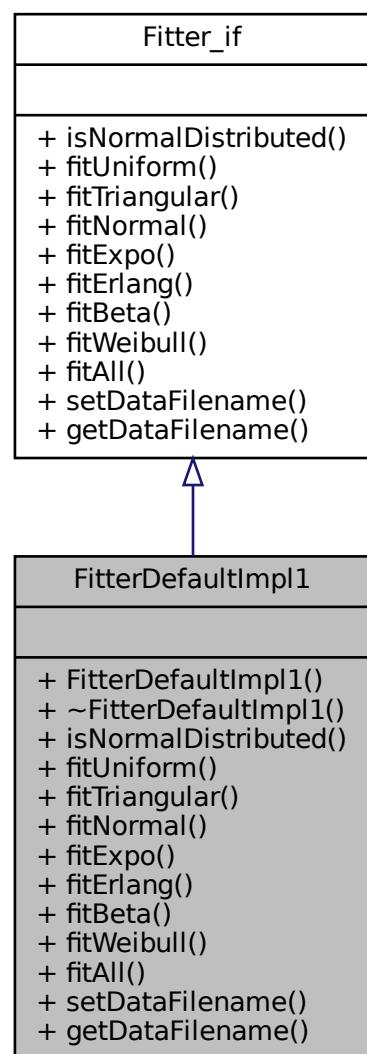
The documentation for this class was generated from the following file:

- [Fitter_if.h](#)

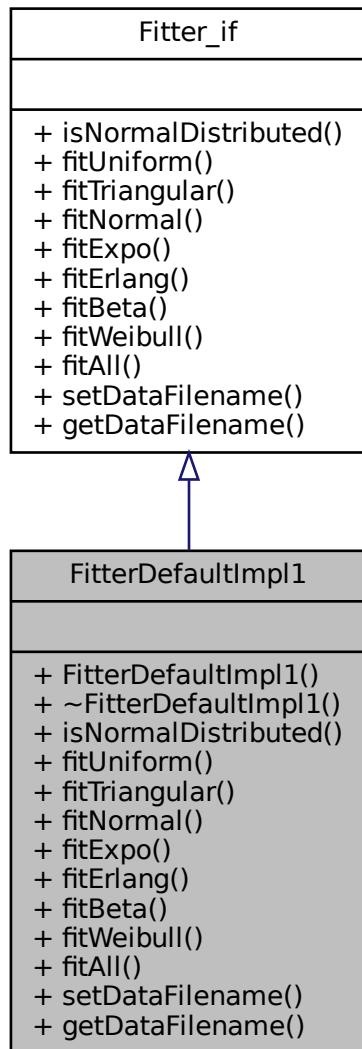
8.41 FitterDefaultImpl1 Class Reference

```
#include <FitterDefaultImpl1.h>
```

Inheritance diagram for FitterDefaultImpl1:



Collaboration diagram for FitterDefaultImpl1:



Public Member Functions

- [FitterDefaultImpl1 \(\)](#)
- virtual [~FitterDefaultImpl1 \(\)](#)=default
- bool [isNormalDistributed](#) (double confidencelevel)
- void [fitUniform](#) (double *sqrerror, double *min, double *max)
- void [fitTriangular](#) (double *sqrerror, double *min, double *mo, double *max)
- void [fitNormal](#) (double *sqrerror, double *avg, double *stddev)
- void [fitExpo](#) (double *sqrerror, double *avg1)
- void [fitErlang](#) (double *sqrerror, double *avg, double *m)
- void [fitBeta](#) (double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit)
- void [fitWeibull](#) (double *sqrerror, double *alpha, double *scale)
- void [fitAll](#) (double *sqrerror, std::string *name)
- void [setDataFilename](#) (std::string dataFilename)
- std::string [getDataFilename \(\)](#)

8.41.1 Detailed Description

Definition at line 19 of file [FitterDefaultImpl1.h](#).

8.41.2 Constructor & Destructor Documentation

8.41.2.1 **FitterDefaultImpl1()** `FitterDefaultImpl1::FitterDefaultImpl1 ()`

Definition at line 16 of file [FitterDefaultImpl1.cpp](#).

```
00016 {  
00017 }
```

8.41.2.2 **~FitterDefaultImpl1()** `virtual FitterDefaultImpl1::~FitterDefaultImpl1 () [virtual], [default]`

8.41.3 Member Function Documentation

8.41.3.1 **fitAll()** `void FitterDefaultImpl1::fitAll (double * sgrerror, std::string * name) [virtual]`

Implements [Fitter_if](#).

Definition at line 50 of file [FitterDefaultImpl1.cpp](#).

```
00050 {  
00051  
00052 }
```

8.41.3.2 **fitBeta()** `void FitterDefaultImpl1::fitBeta (double * sgrerror, double * alpha, double * beta, double * infLimit, double * supLimit) [virtual]`

Implements [Fitter_if](#).

Definition at line 42 of file [FitterDefaultImpl1.cpp](#).

```
00042 {  
00043  
00044 }
```

8.41.3.3 fitErlang() void FitterDefaultImpl1::fitErlang (double * *sqrerror*, double * *avg*, double * *m*) [virtual]

Implements [Fitter_if](#).

Definition at line 38 of file [FitterDefaultImpl1.cpp](#).

```
00038
00039
00040 }
```

8.41.3.4 fitExpo() void FitterDefaultImpl1::fitExpo (double * *sqrerror*, double * *avg1*) [virtual]

Implements [Fitter_if](#).

Definition at line 35 of file [FitterDefaultImpl1.cpp](#).

```
00035
00036 }
```

8.41.3.5 fitNormal() void FitterDefaultImpl1::fitNormal (double * *sqrerror*, double * *avg*, double * *stddev*) [virtual]

Implements [Fitter_if](#).

Definition at line 31 of file [FitterDefaultImpl1.cpp](#).

```
00031
00032
00033 }
```

8.41.3.6 fitTriangular() void FitterDefaultImpl1::fitTriangular (double * *sqrerror*, double * *min*, double * *mo*, double * *max*) [virtual]

Implements [Fitter_if](#).

Definition at line 27 of file [FitterDefaultImpl1.cpp](#).

```
00027
00028
00029 }
```

8.41.3.7 fitUniform() void FitterDefaultImpl1::fitUniform (double * *sqrerror*, double * *min*, double * *max*) [virtual]

Implements [Fitter_if](#).

Definition at line 23 of file [FitterDefaultImpl1.cpp](#).

```
00023
00024
00025 }
```

8.41.3.8 fitWeibull() void FitterDefaultImpl1::fitWeibull (double * *sqrerror*, double * *alpha*, double * *scale*) [virtual]

Implements [Fitter_if](#).

Definition at line 46 of file [FitterDefaultImpl1.cpp](#).

```
00046
00047
00048 }
```

8.41.3.9 getDataFilename() std::string FitterDefaultImpl1::getDataFilename () [virtual]

Implements [Fitter_if](#).

Definition at line 58 of file [FitterDefaultImpl1.cpp](#).

```
00058
00059     return ""; // \todo
00060 }
```

8.41.3.10 isNormalDistributed() bool FitterDefaultImpl1::isNormalDistributed (double *confidencelevel*) [virtual]

Implements [Fitter_if](#).

Definition at line 19 of file [FitterDefaultImpl1.cpp](#).

```
00019
00020     return true;
00021 }
```

8.41.3.11 setDataFilename() void FitterDefaultImpl1::setDataFilename (std::string *datafilename*) [virtual]

Implements [Fitter_if](#).

Definition at line 54 of file [FitterDefaultImpl1.cpp](#).

```
00054
00055
00056 }
```

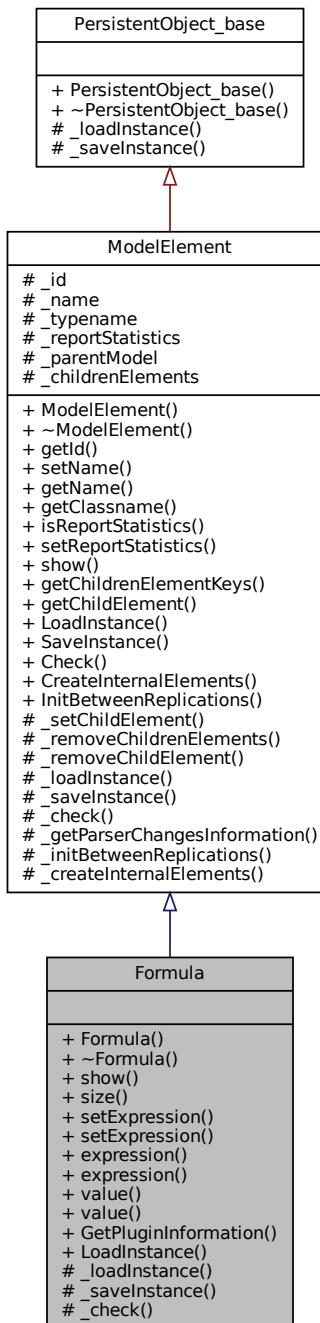
The documentation for this class was generated from the following files:

- [FitterDefaultImpl1.h](#)
- [FitterDefaultImpl1.cpp](#)

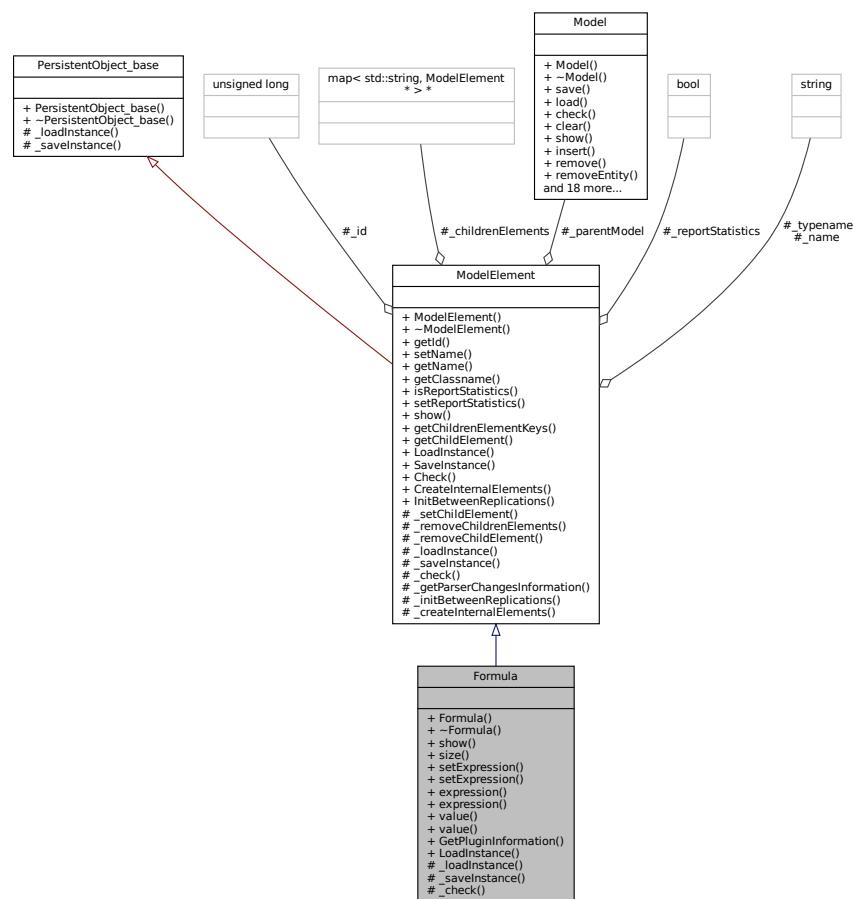
8.42 Formula Class Reference

```
#include <Formula.h>
```

Inheritance diagram for Formula:



Collaboration diagram for Formula:



Public Member Functions

- `Formula (Model *model, std::string name="")`
 - `virtual ~Formula ()=default`
 - `virtual std::string show ()`
 - `unsigned int size ()`
 - `void setExpression (std::string index, std::string formulaExpression)`
 - `void setExpression (std::string formulaExpression)`
 - `std::string expression (std::string index)`
 - `std::string expression ()`
 - `double value ()`
 - `double value (std::string index)`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
 - static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)

Additional Inherited Members

8.42.1 Detailed Description

Definition at line [22](#) of file [Formula.h](#).

8.42.2 Constructor & Destructor Documentation

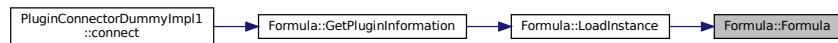
8.42.2.1 Formula() `Formula::Formula (`
 `Model * model,`
 `std::string name = "")`

Definition at line [20](#) of file [Formula.cpp](#).

```
00020 : ModelElement(model, Util::TypeOf<Formula>(), name) {  
00021     //_myPrivateParser = new Traits<Parser_if>::Implementation(_parentModel);  
00022 }
```

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

Here is the caller graph for this function:



8.42.2.2 ~Formula() `virtual Formula::~Formula () [virtual], [default]`

8.42.3 Member Function Documentation

```
8.42.3.1 _check() bool Formula::_check (
    std::string * errorMessage ) [protected], [virtual]
```

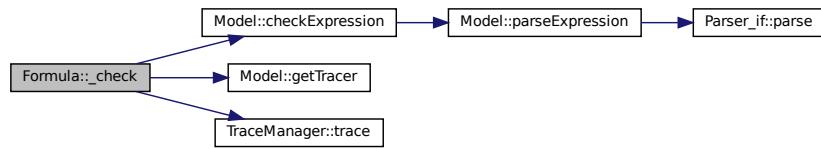
Reimplemented from [ModelElement](#).

Definition at line 103 of file [Formula.cpp](#).

```
00103
00104     std::string errorMsg = "";
00105     bool res, resAll = true;
00106     //unsigned int i = 0;
00107     for (std::map<std::string, std::string>::iterator it = _formulaExpressions->begin(); it != _formulaExpressions->end(); it++) {
00108         res = \_parentModel->checkExpression((*it).second, "formula expression[" + (*it).first + "]", &errorMsg);
00109         if (!res) {
00110             \_parentModel->getTracer\(\)->trace(Util::TraceLevel::errorFatal, "Error parsing expression
\\"" + (*it).second + "\\"");
00111         }
00112         resAll &= res;
00113     }
00114     return resAll;
00115 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), [Util::errorFatal](#), [Model::getTracer\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



```
8.42.3.2 _loadInstance() bool Formula::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

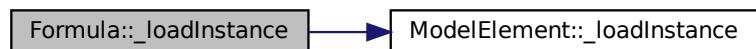
Definition at line 93 of file [Formula.cpp](#).

```
00093
00094     return ModelElement::_loadInstance(fields);
00095 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.3.3 `_saveInstance()` `std::map< std::string, std::string > * Formula::_saveInstance ()`
[protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 97 of file [Formula.cpp](#).

```
00097
00098     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00099     //fields->emplace("...", std::to_string(this->...));
00100     return fields;
00101 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.42.3.4 `expression() [1/2]` `std::string Formula::expression ()`

Definition at line 59 of file [Formula.cpp](#).

```
00059
00060     return expression("");
00061 }
```

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

Here is the caller graph for this function:



8.42.3.5 expression() [2/2] `std::string Formula::expression (std::string index)`

Definition at line 46 of file [Formula.cpp](#).

```
00046     {
00047         std::map<std::string, std::string>::iterator it = _formulaExpressions->find(index);
00048         if (it == _formulaExpressions->end()) {
00049             return ""; // index does not exist. No formula expressions returned. \todo: Should it be
00050             traced?.
00051         } else {
00052             return it->second;
00053     }
```

8.42.3.6 GetPluginInformation() `PluginInformation * Formula::GetPluginInformation () [static]`

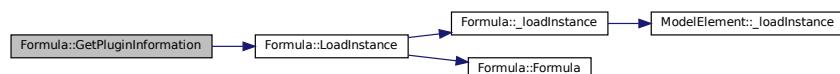
Definition at line 78 of file [Formula.cpp](#).

```
00078     {
00079         PluginInformation* info = new PluginInformation(Util::TypeOf<Formula>(), &Formula::LoadInstance);
00080         return info;
00081     }
```

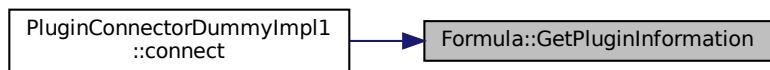
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.3.7 **LoadInstance()**

```
ModelElement * Formula::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

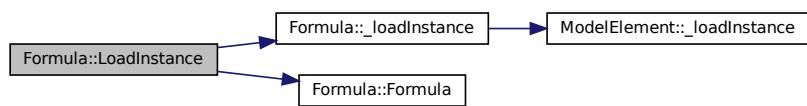
Definition at line 83 of file [Formula.cpp](#).

```
00083
00084     Formula* newElement = new Formula(model);
00085     try {
00086         newElement->_loadInstance(fields);
00087     } catch (const std::exception& e) {
00088     }
00089 }
00090 return newElement;
00091 }
```

References [_loadInstance\(\)](#), and [Formula\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.3.8 **setExpression() [1/2]**

```
void Formula::setExpression (
    std::string formulaExpression )
```

Definition at line 55 of file [Formula.cpp](#).

```
00055
00056     setExpression("", formulaExpression);
00057 }
```

References [setExpression\(\)](#).

Here is the call graph for this function:



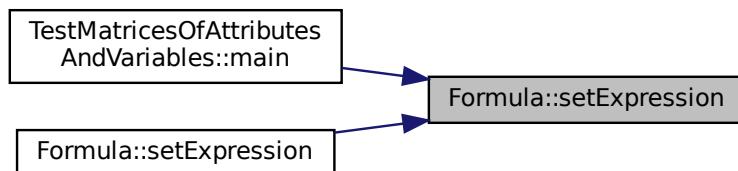
8.42.3.9 setExpression() [2/2] void Formula::setExpression (std::string index, std::string formulaExpression)

Definition at line 37 of file [Formula.cpp](#).

```
00037
00038     std::map<std::string, std::string>::iterator mapIt = _formulaExpressions->find(index);
00039     if (mapIt != _formulaExpressions->end()) { //found
00040         (*mapIt).second = formulaExpression;
00041     } else { //not found
00042         _formulaExpressions->insert({index, formulaExpression});
00043     }
00044 }
```

Referenced by [TestMatricesOfAttributesAndVariables::main\(\)](#), and [setExpression\(\)](#).

Here is the caller graph for this function:



8.42.3.10 show() std::string Formula::show () [virtual]

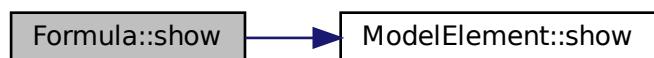
Reimplemented from [ModelElement](#).

Definition at line 24 of file [Formula.cpp](#).

```
00024
00025     std::string expressions = "";
00026     unsigned int i = 0;
00027     // for (std::list<std::string>::iterator it = _formulaExpressions->list()->begin(); it != _formulaExpressions->list()->end(); it++) {
00028     //expressions += "expression[" + std::to_string(i++) + "]=" + (*it) + "\n";
00029     //}
00030     return ModelElement::show() + expressions;
00031 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



8.42.3.11 size() `unsigned int Formula::size ()`

Definition at line 33 of file [Formula.cpp](#).

```
00033     {
00034     return _formulaExpressions->size();
00035 }
```

8.42.3.12 value() [1/2] `double Formula::value ()`

Definition at line 74 of file [Formula.cpp](#).

```
00074     {
00075     return value("");
00076 }
```

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

Here is the caller graph for this function:

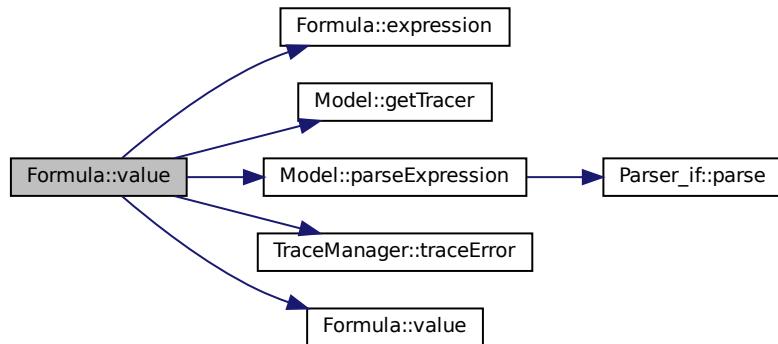
**8.42.3.13 value() [2/2]** `double Formula::value (std::string index)`

Definition at line 63 of file [Formula.cpp](#).

```
00063     {
00064     std::string strexpression = this->expression(index);
00065     double value = 0.0;
00066     try {
00067         value = _parentModel->parseExpression(strexpression);
00068     } catch (const std::exception& e) {
00069         _parentModel->getTracer()->traceError(e, "Error parsing formula \" + strexpression + '\"");
00070     }
00071     return value;
00072 }
```

References [ModelElement::_parentModel](#), [expression\(\)](#), [Model::getTracer\(\)](#), [Model::parseExpression\(\)](#), [TraceManager::traceError\(\)](#), and [value\(\)](#).

Here is the call graph for this function:



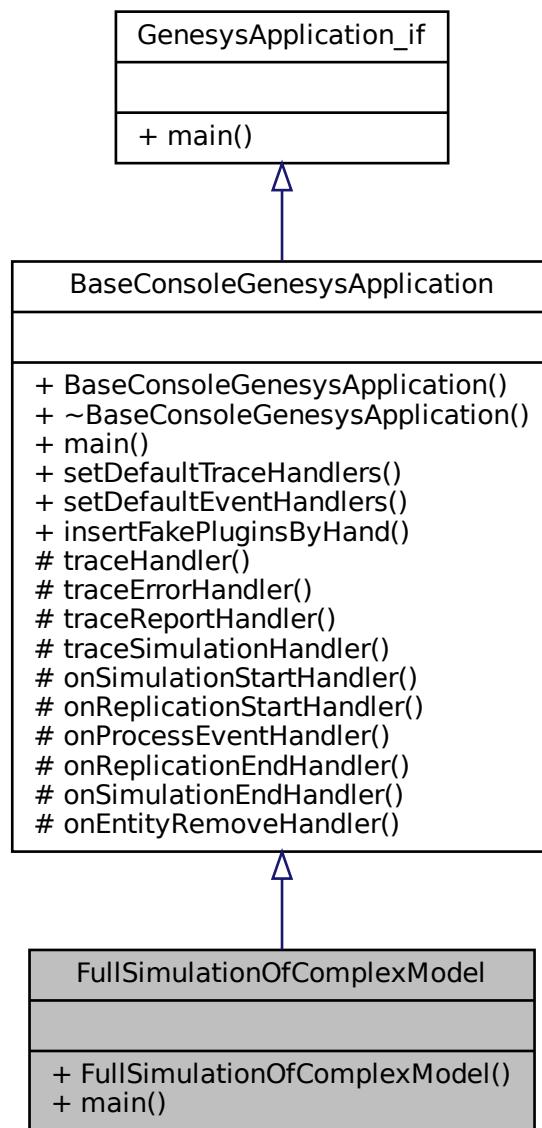
The documentation for this class was generated from the following files:

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

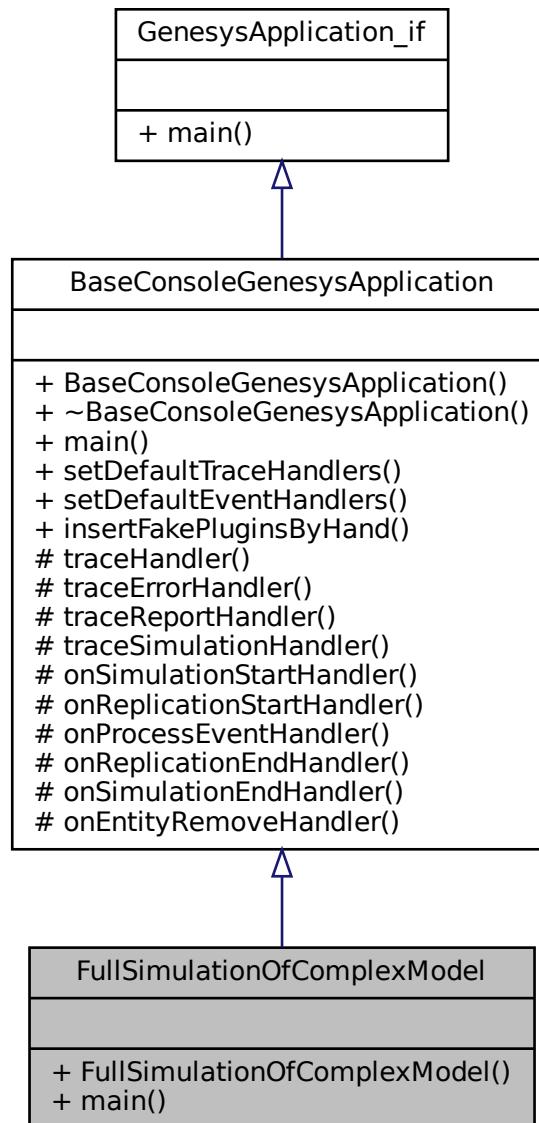
8.43 FullSimulationOfComplexModel Class Reference

```
#include <FullSimulationOfComplexModel.h>
```

Inheritance diagram for FullSimulationOfComplexModel:



Collaboration diagram for FullSimulationOfComplexModel:



Public Member Functions

- `FullSimulationOfComplexModel ()`
- virtual int `main` (int argc, char **argv)

Additional Inherited Members

8.43.1 Detailed Description

Definition at line 20 of file [FullSimulationOfComplexModel.h](#).

8.43.2 Constructor & Destructor Documentation

8.43.2.1 FullSimulationOfComplexModel() FullSimulationOfComplexModel::FullSimulationOfComplex<
Model ()

Definition at line 32 of file [FullSimulationOfComplexModel.cpp](#).

```
00032
00033
00034 }
```

8.43.3 Member Function Documentation

8.43.3.1 main() int FullSimulationOfComplexModel::main (

```
    int argc,
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

Definition at line 40 of file [FullSimulationOfComplexModel.cpp](#).

```
00040
00041     Simulator* genesys = new Simulator();
00042     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00043     this->insertFakePluginsByHand(genesys);
00044     // creates an empty model
00045     Model* model = genesys->getModels()->newModel();
00046     // Handle traces and simulation events to output them
00047     TraceManager* tm = model->getTracer();
00048     this->setDefaultTraceHandlers(tm);
00049     // get easy access to classes used to insert components and elements into a model
00050     ComponentManager* components = model->getComponents();
00051     ElementManager* elements = model->getElements();
00052     //
00053     // build the simulation model
00054     //
00055     ModelInfo* infos = model->getInfos();
00056     infos->setAnalystName("Your name");
00057     infos->setProjectTitle("The title of the project");
00058     infos->setDescription("This simulation model tests the components and elements that have been
implemented so far.");
00059
00060     ModelSimulation* sim = model->getSimulation();
00061     sim->setReplicationLength(1e4);
00062     sim->setReplicationLengthTimeUnit(Util::TimeUnit::minute);
00063     sim->setNumberOfReplications(3000);
00064     tm->setTraceLevel(Util::TraceLevel::modelResult);
00065
00066     EntityType* entityType1 = new EntityType(model, "Representative_EntityType");
00067
00068     Create* create1 = new Create(model);
00069     create1->setEntityType(entityType1);
00070     create1->setTimeBetweenCreationsExpression("EXPO(5)");
00071     create1->setTimeUnit(Util::TimeUnit::minute);
00072     create1->setEntitiesPerCreation(1);
00073     components->insert(create1);
00074
00075     Attribute* attribute1 = new Attribute(model, "Attribute_1");
00076     Variable* variable1 = new Variable(model, "Variable_1");
00077
00078     Assign* assign1 = new Assign(model);
00079     Assign::Assignment* attrib2Assignment = new Assign::Assignment("Variable_1", "Variable_1 + 1");
00080     assign1->getAssignments()->insert(attrib2Assignment);
00081     Assign::Assignment* attrib1Assignment = new Assign::Assignment("Attribute_1", "Variable_1");
```

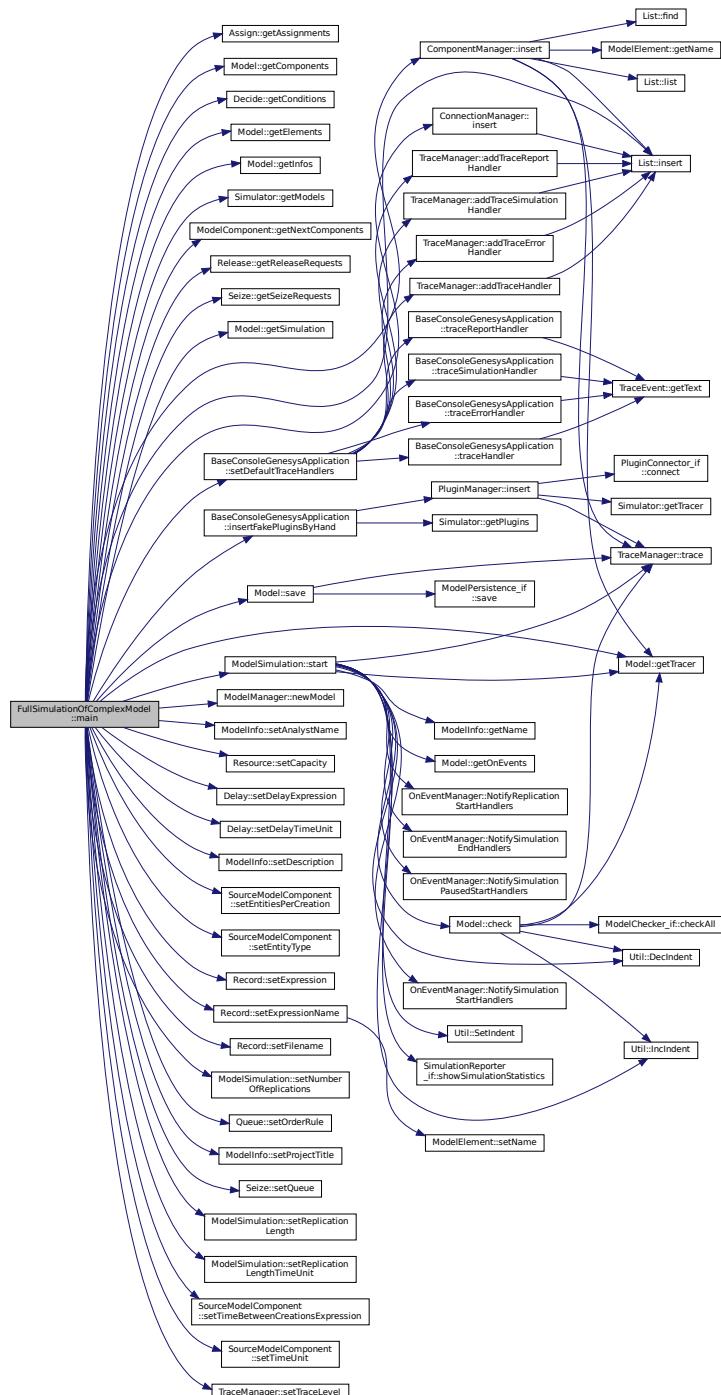
```

00082     assign1->getAssignments()->insert(attrib1Assignment);
00083
00084     Decide* decide1 = new Decide(model);
00085     decide1->getConditions()->insert("UNIF(0,1) > 0.5");
00086
00087     Resource* maquinal = new Resource(model, "Máquina 1");
00088     maquinal->setCapacity(1);
00089
00090     Queue* filaSeize1 = new Queue(model);
00091     filaSeize1->setOrderRule(Queue::OrderRule::FIFO);
00092
00093     Seize* seize1 = new Seize(model);
00094     seize1->getSeizeRequests()->insert(new SeizableItemRequest(maquinal));
00095     seize1->setQueue(filaSeize1);
00096
00097     Delay* delay1 = new Delay(model);
00098     delay1->setDelayExpression("NORM(5, 3)");
00099     delay1->setDelayTimeUnit(Util::TimeUnit::minute);
00100
00101     Release* release1 = new Release(model);
00102     release1->getReleaseRequests()->insert(new SeizableItemRequest(maquinal));
00103
00104     Record* record1 = new Record(model);
00105     record1->setExpressionName("Tempo total no sistema");
00106     record1->setExpression("TNOW - Entity.ArrivalTime");
00107     record1->setFilename("./temp/TotalTimeInSystem.txt");
00108
00109     Dispose* dispose1 = new Dispose(model);
00110
00111     // connect model components to create a "workflow" -- should always start from a
00112     // SourceModelComponent and end at a SinkModelComponent (it will be checked)
00113     create1->getNextComponents()->insert(assign1);
00114     assign1->getNextComponents()->insert(decide1);
00115     decide1->getNextComponents()->insert(seize1);
00116     seize1->getNextComponents()->insert(delay1);
00117     delay1->getNextComponents()->insert(release1);
00118     release1->getNextComponents()->insert(record1);
00119     record1->getNextComponents()->insert(dispose1);
00120     // then save the model into a text file
00121     model->save("./models/AssignWrite3Seizes.txt");
00122     // execute the simulation
00123     model->getSimulation()->start();
00124
00125     return 0;
00126 };

```

References Queue::FIFO, Assign::getAssignments(), Model::getComponents(), Decide::getConditions(), Model::getElements(), Model::getInfos(), Simulator::getModels(), ModelComponent::getNextComponents(), Release::getReleaseRequests(), Seize::getSeizeRequests(), Model::getSimulation(), Model::getTracer(), ComponentManager::insertConnectionManager::insert(), List< T >::insert(), BaseConsoleGenesysApplication::insertFakePluginsByHand(), Util::minute, Util::modelResult, ModelManager::newModel(), Model::save(), ModelInfo::setAnalystName(), Resource::setCapacity(), BaseConsoleGenesysApplication::setDefaultTraceHandlers(), Delay::setDelayExpression(), Delay::setDelayTimeUnit(), ModelInfo::setDescription(), SourceModelComponent::setEntitiesPerCreation(), SourceModelComponent::setEntityType(), Record::setExpression(), Record::setExpressionName(), Record::setFilename(), ModelSimulation::setNumberOfReplications(), Queue::setOrderRule(), ModelInfo::setProjectTitle(), Seize::setQueue(), ModelSimulation::setReplicationLength(), ModelSimulation::setReplicationLengthTimeUnit(), SourceModelComponent::setTimeBetween(), SourceModelComponent::setTimeUnit(), TraceManager::setTraceLevel(), and ModelSimulation::start().

Here is the call graph for this function:



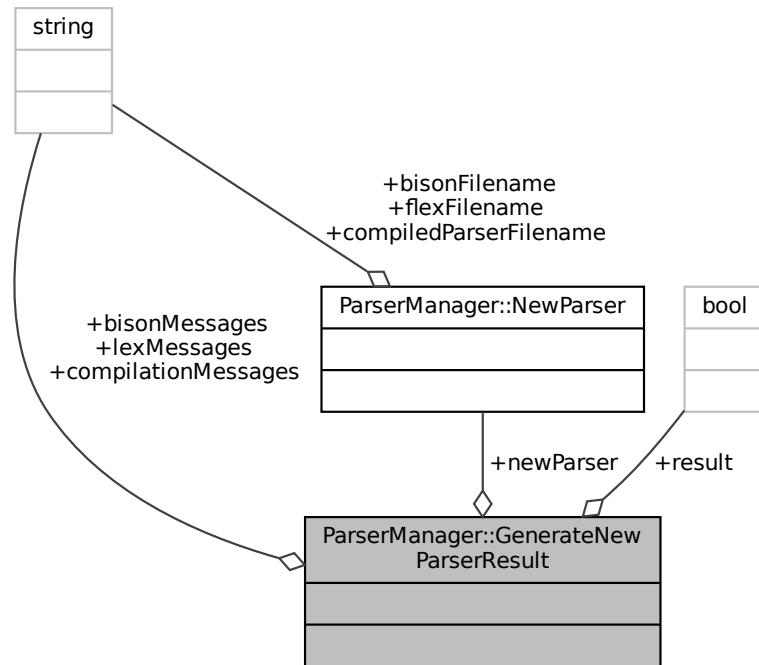
The documentation for this class was generated from the following files:

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

8.44 ParserManager::GenerateNewParserResult Struct Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager::GenerateNewParserResult:



Public Attributes

- `bool result`
- `std::string bisonMessages`
- `std::string lexMessages`
- `std::string compilationMessages`
- `NewParser newParser`

8.44.1 Detailed Description

Definition at line 30 of file [ParserManager.h](#).

8.44.2 Member Data Documentation

8.44.2.1 bisonMessages std::string ParserManager::GenerateNewParserResult::bisonMessages

Definition at line 32 of file [ParserManager.h](#).

8.44.2.2 compilationMessages std::string ParserManager::GenerateNewParserResult::compilation→

Messages

Definition at line 34 of file [ParserManager.h](#).

8.44.2.3 lexMessages std::string ParserManager::GenerateNewParserResult::lexMessages

Definition at line 33 of file [ParserManager.h](#).

8.44.2.4 newParser `NewParser ParserManager::GenerateNewParserResult::newParser`

Definition at line 35 of file [ParserManager.h](#).

8.44.2.5 **result** bool ParserManager::GenerateNewParserResult::result

Definition at line 31 of file [ParserManager.h](#).

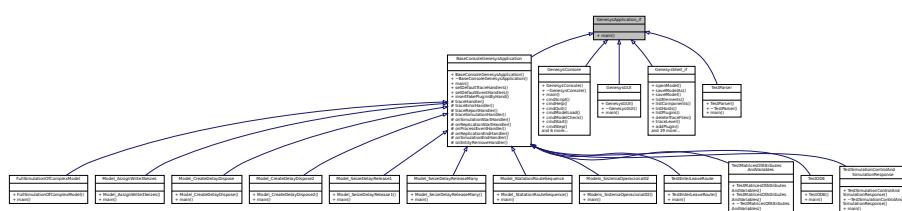
The documentation for this struct was generated from the following file:

- ParserManager.h

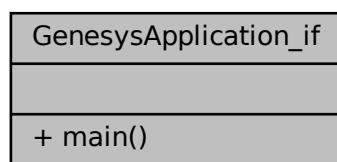
8.45 GenesysApplication if Class Reference

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

Inheritance diagram for GenesysApplication_if:



Collaboration diagram for GenesysApplication_if:



Public Member Functions

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

8.45.1 Detailed Description

Definition at line [17](#) of file [GenesysApplication_if.h](#).

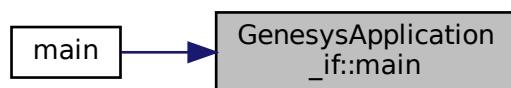
8.45.2 Member Function Documentation

8.45.2.1 main() virtual int GenesysApplication_if::main (int argc, char ** argv) [pure virtual]

Implemented in [BaseConsoleGenesysApplication](#), [GenesysConsole](#), [GenesysGUI](#), [FullSimulationOfComplexModel](#), [TestMatricesOfAttributesAndVariables](#), [TestParser](#), [TestSimulationControlAndSimulationResponse](#), [Model_AssignWrite3Seizes](#), [Model_CreateDelayDispose](#), [Model_CreateDelayDispose2](#), [Model_SeizeDelayRelease1](#), [Model_SeizeDelayReleaseMany](#), [Model_StatationRouteSequence](#), [Modelo_SistemaOperacional02](#), [TestEnterLeaveRoute](#), and [TestODE](#).

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

Here is the caller graph for this function:



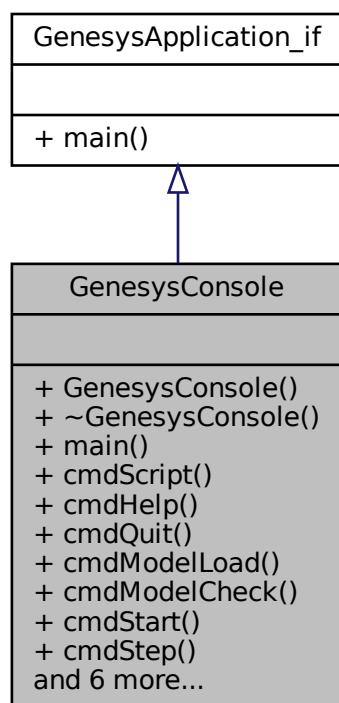
The documentation for this class was generated from the following file:

- [GenesysApplication_if.h](#)

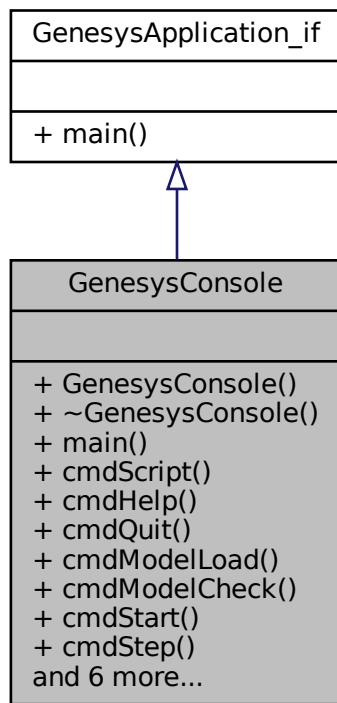
8.46 GenesysConsole Class Reference

```
#include <GenesysConsole.h>
```

Inheritance diagram for GenesysConsole:



Collaboration diagram for GenesysConsole:



Public Member Functions

- [GenesysConsole \(\)](#)
- virtual [~GenesysConsole \(\)](#)=default
- virtual int [main \(int argc, char **argv\)](#)
- void [cmdScript \(\)](#)
- void [cmdHelp \(\)](#)
- void [cmdQuit \(\)](#)
- void [cmdModelLoad \(\)](#)
- void [cmdModelCheck \(\)](#)
- void [cmdStart \(\)](#)
- void [cmdStep \(\)](#)
- void [cmdStop \(\)](#)
- void [cmdShowReport \(\)](#)
- void [cmdModelSave \(\)](#)
- void [cmdModelShow \(\)](#)
- void [cmdVersion \(\)](#)
- void [cmdTraceLevel \(\)](#)

8.46.1 Detailed Description

Definition at line 21 of file [GenesysConsole.h](#).

8.46.2 Constructor & Destructor Documentation

8.46.2.1 GenesysConsole() GenesysConsole::GenesysConsole ()

Definition at line 23 of file [GenesysConsole.cpp](#).

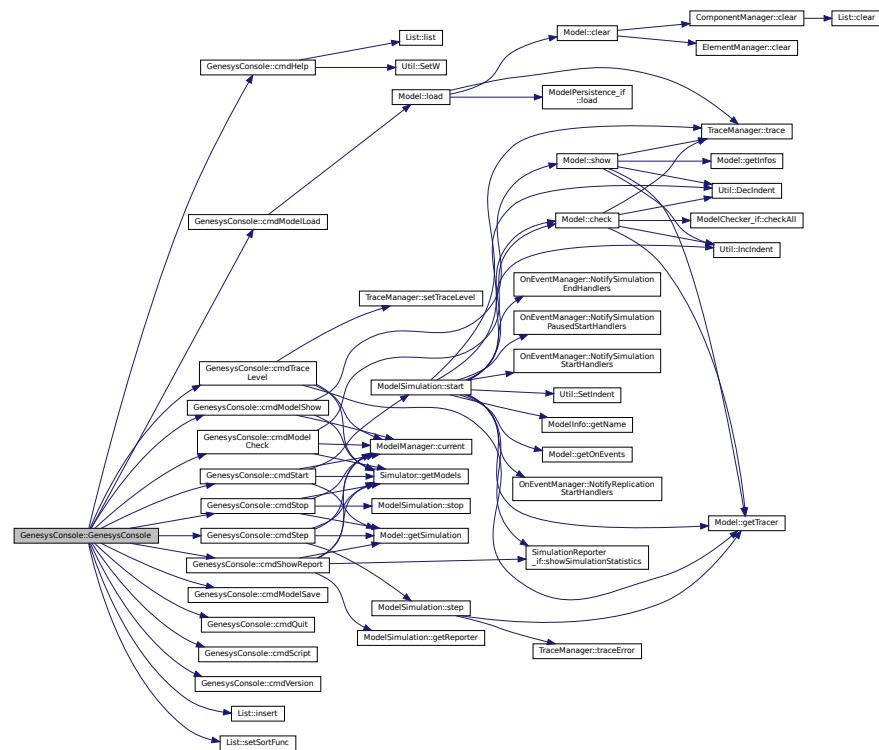
```

00023     {
00024         _commands->setSortFunc([](const ShellCommand* a, const ShellCommand * b) {
00025             return a->shortname < b->shortname;
00026         });
00027         _commands->insert(new ShellCommand("q", "quit", "", "Quit ReGenesys shell. Same as exit.",
00028             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00029         _commands->insert(new ShellCommand("x", "exit", "", "Exit ReGenesys shell. Same as quit.",
00030             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00031         _commands->insert(new ShellCommand("h", "help", "", "Show help for commands.",
00032             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdHelp)));
00033         _commands->insert(new ShellCommand("ms", "modelsave", "<filename>", "Save model.",
00034             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelSave)));
00035         _commands->insert(new ShellCommand("ml", "modelload", "<filename>", "Load Model.",
00036             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelLoad)));
00037         _commands->insert(new ShellCommand("rf", "readfile", "<filename>", "Read and execute shell command
00038             from file.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdScript)));
00039         _commands->insert(new ShellCommand("v", "version", "", "Show the version.",
00040             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdVersion)));
00041         _commands->insert(new ShellCommand("ss", "start", "", "Start simulation.",
00042             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStart)));
00043         _commands->insert(new ShellCommand("mc", "modelcheck", "", "Check model.",
00044             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelCheck)));
00045         _commands->insert(new ShellCommand("mh", "modelshow", "", "Show model.",
00046             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelShow)));
00047         _commands->insert(new ShellCommand("ps", "step", "", "step simulation.",
00048             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStep)));
00049         _commands->insert(new ShellCommand("ts", "stop", "", "Stop simulation.",
00050             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStop)));
00051         _commands->insert(new ShellCommand("sr", "showreport", "", "Show simulation report.",
00052             DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdShowReport)));
00053         _commands->insert(new ShellCommand("tl", "tracelevel", "<0|1|2|...|7>", "Set the trace level (the
00054             bigger the most verbose).", DefineExecuterMember<GenesysConsole>(this,
00055             &GenesysConsole::cmdTraceLevel)));
00056     }

```

References [cmdHelp\(\)](#), [cmdModelCheck\(\)](#), [cmdModelLoad\(\)](#), [cmdModelSave\(\)](#), [cmdModelShow\(\)](#), [cmdQuit\(\)](#), [cmdScript\(\)](#), [cmdShowReport\(\)](#), [cmdStart\(\)](#), [cmdStep\(\)](#), [cmdStop\(\)](#), [cmdTraceLevel\(\)](#), [cmdVersion\(\)](#), [List< T >::insert\(\)](#), and [List< T >::setSortFunc\(\)](#).

Here is the call graph for this function:



8.46.2.2 ~GenesysConsole() virtual GenesysConsole::~GenesysConsole () [virtual], [default]

8.46.3 Member Function Documentation

8.46.3.1 cmdHelp() void GenesysConsole::cmdHelp ()

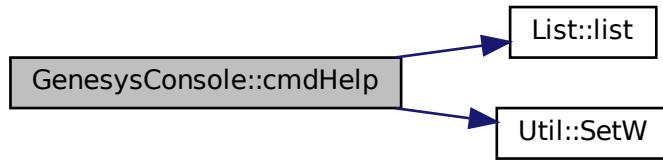
Definition at line 99 of file [GenesysConsole.cpp](#).

```
00099         {
00100     ShellCommand* command;
00101     Trace("List of commands:");
00102     Trace(Util::SetW("Short", 6) + Util::SetW("Long", 12) + Util::SetW("Parameters", 15) +
00103 "Description");
00104     for (std::list<ShellCommand*>::iterator it = _commands->list()->begin(); it != _commands->list()->end(); it++) {
00105         //Trace("Unknown command. Type \"-h\" or \"help\" for help on possible commands.");
00106         command = (*it);
00107         Trace(Util::SetW(command->shortname, 6) + Util::SetW(command->longname, 12) +
00108 Util::SetW(command->parameters, 15) + command->description);
00109     }
00110 }
```

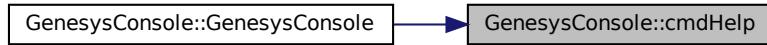
References `List< T >::list()`, and `Util::SetW()`.

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.2 cmdModelCheck() void GenesysConsole::cmdModelCheck ()

Definition at line 54 of file [GenesysConsole.cpp](#).

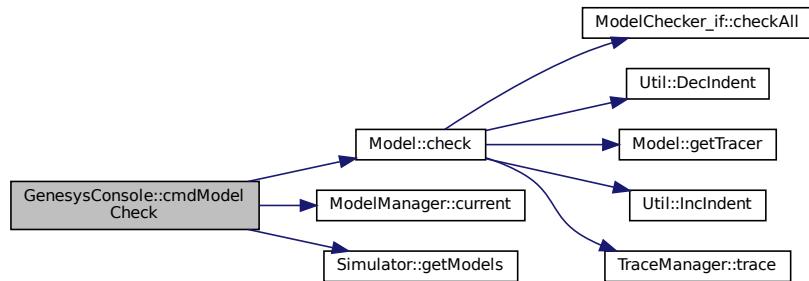
```

00054     Trace("Check model");
00055     {
00056         try {
00057             _simulator->getModels()->current()->check();
00058         } catch (...) {
00059             Trace("Error checking model");
00060         }
00061     }
  
```

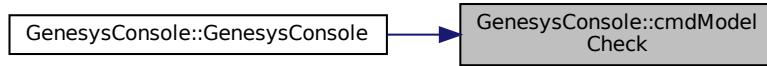
References [Model::check\(\)](#), [ModelManager::current\(\)](#), and [Simulator::getModels\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.3 cmdModelLoad() void GenesysConsole::cmdModelLoad ()

Definition at line 119 of file [GenesysConsole.cpp](#).

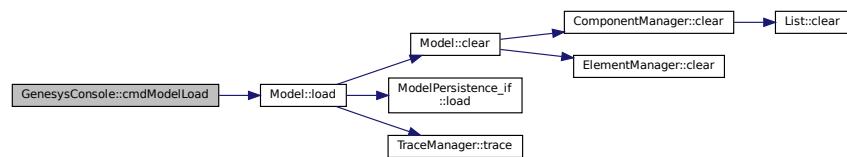
```

00119     Trace("Model load");
00120     {
00121         try {
00122             std::string filename = _parameter;
00123             Model* model = new Model(this->_simulator);
00124             model->load(filename);
00125         } catch (...) {
00126             // _commands
00127             Trace("Error loading");
00128         }
00129     }
  
```

References [Model::load\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.4 cmdModelSave() void GenesysConsole::cmdModelSave ()

Definition at line 141 of file [GenesysConsole.cpp](#).

```
00141           {
00142     //this->_parameter;
00143 }
```

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

Here is the caller graph for this function:



8.46.3.5 cmdModelShow() void GenesysConsole::cmdModelShow ()

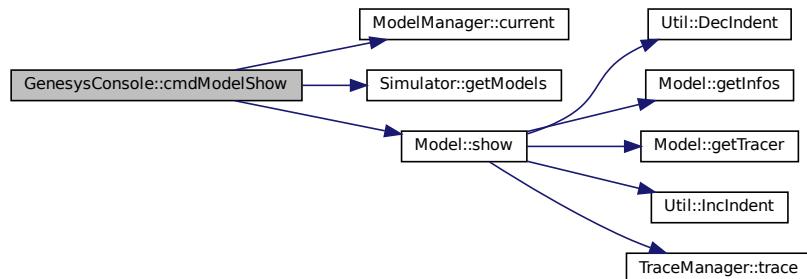
Definition at line 131 of file [GenesysConsole.cpp](#).

```
00131           {
00132     Trace("Model Show");
00133     try {
00134       _simulator->getModels()->current()->show();
00135     } catch (...) {
00136       // _commands
00137       Trace(" Error showing");
00138     }
00139 }
```

References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), and [Model::show\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



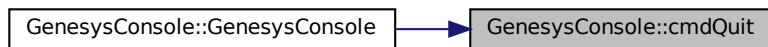
8.46.3.6 cmdQuit() void GenesysConsole::cmdQuit ()

Definition at line 110 of file [GenesysConsole.cpp](#).

```
00110     {
00111     Trace("Quiting/Exiting. Goodbye");
00112     exit(0);
00113 }
```

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

Here is the caller graph for this function:

**8.46.3.7 cmdScript()** void GenesysConsole::cmdScript ()

Definition at line 145 of file [GenesysConsole.cpp](#).

```
00145     {
00146     std::string filename = this->_parameter;
00147     //List<std::string>* arguments = new List<std::string>();
00148     std::ifstream commandfile;
00149     std::string inputText;
00150     try {
00151         commandfile.open(filename);
00152         while (getline(commandfile, inputText)) {
00153             this->tryExecuteCommand(inputText, "", "", " ");
00154         }
00155         commandfile.close();
00156     } catch (...) {
00157         Trace("    Error scripting");
00158     }
00159 }
```

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

Here is the caller graph for this function:



8.46.3.8 cmdShowReport() void GenesysConsole::cmdShowReport ()

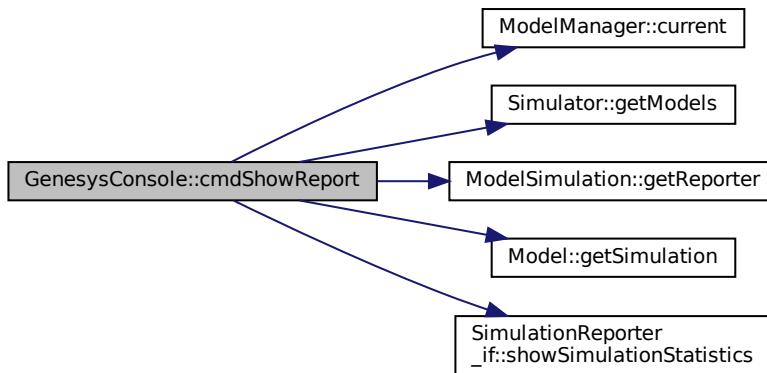
Definition at line 90 of file [GenesysConsole.cpp](#).

```
00090     {
00091         Trace("Show report");
00092         try {
00093             _simulator->getModels()->current()->getSimulation()->getReporter()->showSimulationStatistics();
00094         } catch (...) {
00095             Trace("Error showing reports");
00096         }
00097     }
```

References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), [ModelSimulation::getReporter\(\)](#), [Model::getSimulation\(\)](#), and [SimulationReporter_if::showSimulationStatistics\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.9 cmdStart() void GenesysConsole::cmdStart ()

Definition at line 63 of file [GenesysConsole.cpp](#).

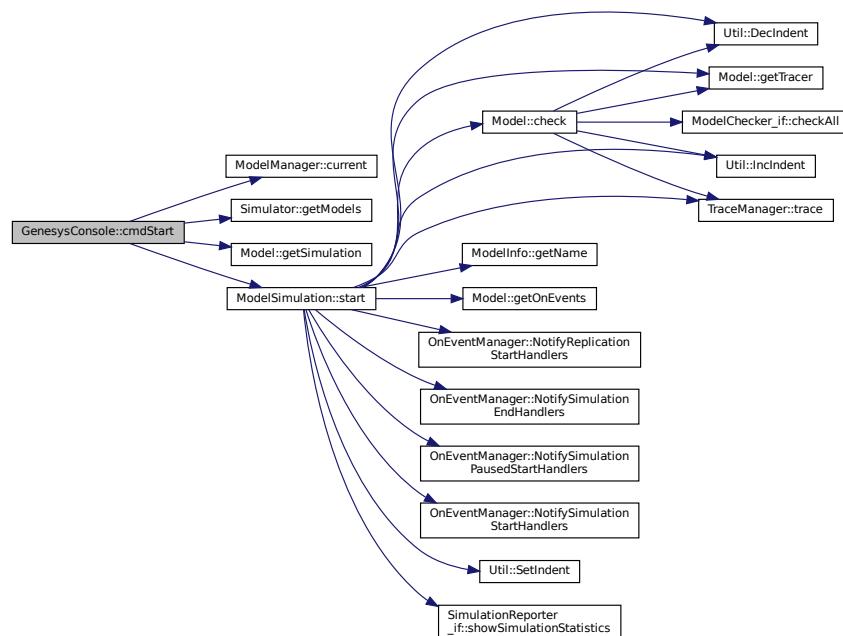
```
00063     {
00064         Trace("Start simulation");
00065         try {
00066             _simulator->getModels()->current()->getSimulation()->start();
00067         } catch (...) {
00068             Trace("Error starting simulation");
00069         }
00070     }
```

```
00070 }
```

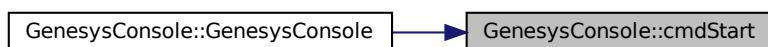
References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), [Model::getSimulation\(\)](#), and [ModelSimulation::start\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.10 cmdStep() void GenesysConsole::cmdStep()

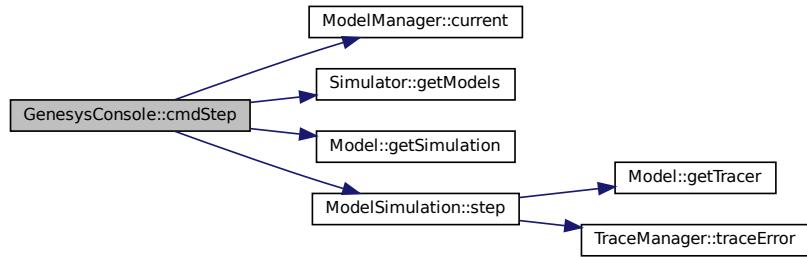
Definition at line 72 of file [GenesysConsole.cpp](#).

```
00072     {
00073         Trace("Step simulation");
00074     try {
00075         _simulator->getModels()->current()->getSimulation()->step();
00076     } catch (...) {
00077         Trace("Error stepping simulation");
00078     }
00079 }
```

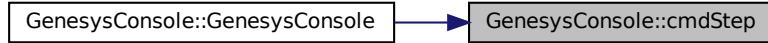
References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), [Model::getSimulation\(\)](#), and [ModelSimulation::step\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.11 cmdStop() void GenesysConsole::cmdStop ()

Definition at line 81 of file [GenesysConsole.cpp](#).

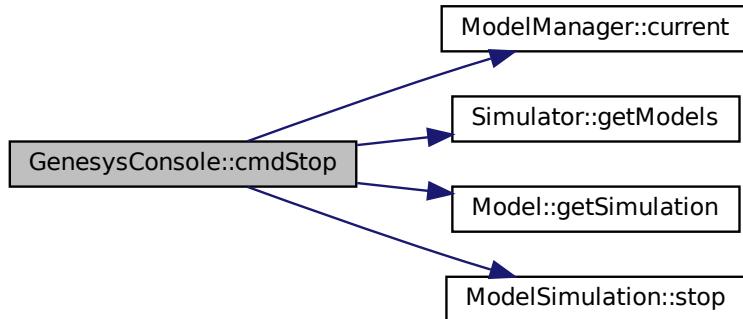
```

00081     {
00082         Trace("Stop simulation");
00083         try {
00084             _simulator->getModels()->current()->getSimulation()->stop();
00085         } catch (...) {
00086             Trace("Error stopping simulation");
00087         }
00088     }
  
```

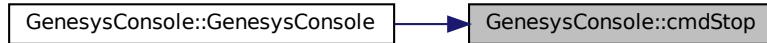
References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), [Model::getSimulation\(\)](#), and [ModelSimulation::stop\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.12 cmdTraceLevel() void GenesysConsole::cmdTraceLevel ()

Definition at line 43 of file [GenesysConsole.cpp](#).

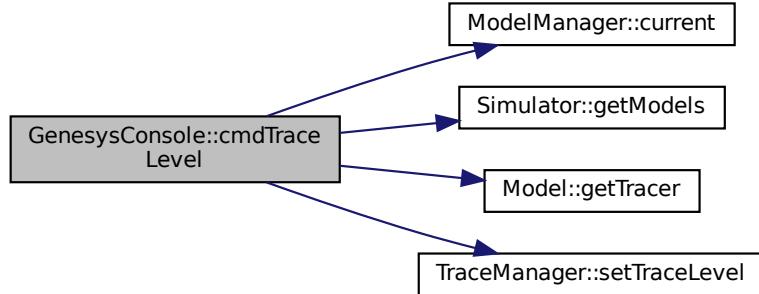
```

00043     Trace("Set trace level");
00044     {
00045         try {
00046             int tlnum = std::stoi(_parameter);
00047             Util::TraceLevel tl = static_cast<Util::TraceLevel>(tlnum);
00048             _simulator->getModels()->current()->getTracer()->setTraceLevel(tl);
00049         } catch (...) {
00050             Trace("Error setting trace level");
00051         }
00052     }
  
```

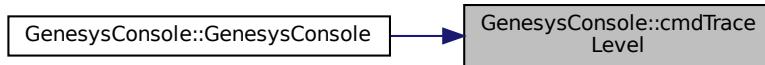
References [ModelManager::current\(\)](#), [Simulator::getModels\(\)](#), [Model::getTracer\(\)](#), and [TraceManager::setTraceLevel\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.46.3.13 cmdVersion() void GenesysConsole::cmdVersion ()

Definition at line 115 of file [GenesysConsole.cpp](#).

```

00115     {
00116     Trace("ReGenesys Shell version 2019.0528");
00117 }
```

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

Here is the caller graph for this function:



```
8.46.3.14 main() int GenesysConsole::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 214 of file [GenesysConsole.cpp](#).

```
00214                                     {
00215     //double res;
00216     //for (double x = -3.0; x <= 3.0; x += 0.05) {
00217     //    res = ProbDistrib::tStudent(x, 0, 1, 100);
00218     //    std::cout << x << ":" << res << std::endl;
00219     //}
00220     //return 0;
00221     List<std::string>* commandlineArgs = new List<std::string>();
00222     for (unsigned short i = 1; i < argc; i++) {
00223         std::string arg = argv[i];
00224         commandlineArgs->insert(arg);
00225     }
00226     //commandlineArgs->insert ("-rf=temp/script.txt");
00227     //commandlineArgs->insert ("-ml=models/genesysmodel.txt");
00228     this->run(commandlineArgs);
00229     return 0;
00230 }
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



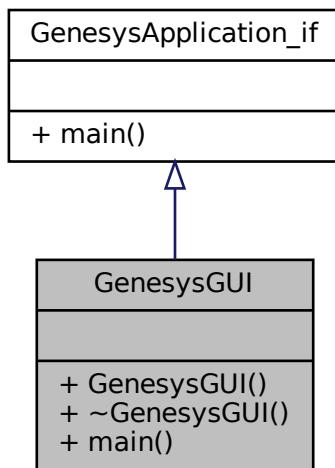
The documentation for this class was generated from the following files:

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

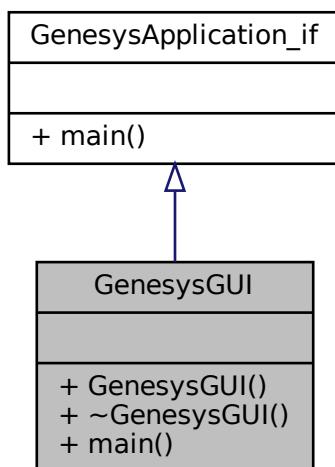
8.47 GenesysGUI Class Reference

```
#include <GenesysGUI.h>
```

Inheritance diagram for GenesysGUI:



Collaboration diagram for GenesysGUI:



Public Member Functions

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

8.47.1 Detailed Description

Definition at line 21 of file [GenesysGUI.h](#).

8.47.2 Constructor & Destructor Documentation

8.47.2.1 GenesysGUI() `GenesysGUI::GenesysGUI ()`

Definition at line 18 of file [GenesysGUI.cpp](#).

```
00018 {  
00019 }
```

8.47.2.2 ~GenesysGUI() `GenesysGUI::~GenesysGUI () [default]`

8.47.3 Member Function Documentation

8.47.3.1 main() `int GenesysGUI::main (int argc, char ** argv) [virtual]`

Implements [GenesysApplication_if](#).

Definition at line 21 of file [GenesysGUI.cpp](#).

```
00021 {  
00022     //     QApplication a(argc, argv);  
00023     //     MainWindow w;  
00024     //     w.show();  
00025     //     return a.exec();  
00026     return 0;  
00027 }
```

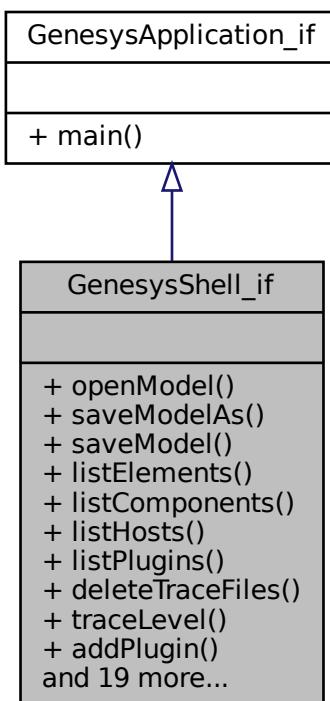
The documentation for this class was generated from the following files:

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

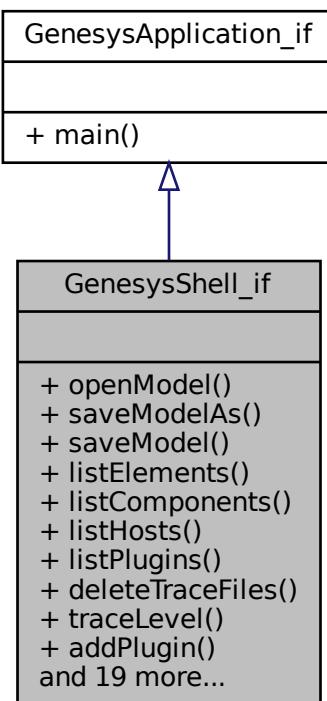
8.48 GenesysShell_if Class Reference

```
#include <GenesysShell_if.h>
```

Inheritance diagram for GenesysShell_if:



Collaboration diagram for GenesysShell_if:



Public Member Functions

- virtual void `openModel` (std::string filename)=0
- virtual void `saveModelAs` (std::string filename)=0
- virtual void `saveModel` ()=0
- virtual void `listElements` ()=0
- virtual void `listComponents` ()=0
- virtual void `listHosts` ()=0
- virtual void `listPlugins` ()=0
- virtual void `deleteTraceFiles` ()=0
- virtual void `traceLevel` (Util::TraceLevel tracelevel)=0
- virtual void `addPlugin` (std::string filename)=0
- virtual void `addFromFile` (std::string filename)=0
- virtual void `readCommandsFromFile` (std::string filename)=0
- virtual void `redirectTrace` (std::string trace, std::string dest, std::string filename)=0
- virtual void `closeModel` ()=0
- virtual void `createModel` ()=0
- virtual void `execLinuxCommand` (std::string command)=0
- virtual void `verboseMode` (bool on)=0
- virtual void `check` ()=0
- virtual void `getGenesysInfo` ()=0
- virtual void `getCommandLine` ()=0
- virtual void `sendFile` (std::string filename, std::string hostname, std::string portname)=0

- virtual void `setActivationCode` (std::string code)=0
- virtual void `receiveFile` (std::string filename)=0
- virtual void `startSimulation` ()=0
- virtual void `stepSimulation` ()=0
- virtual void `stopSimulation` ()=0
- virtual void `showInit` ()=0
- virtual void `showHelp` ()=0
- virtual void `showHostName` ()=0

8.48.1 Detailed Description

Definition at line 20 of file [GenesysShell_if.h](#).

8.48.2 Member Function Documentation

8.48.2.1 `addFromFile()` virtual void GenesysShell_if::addFromFile (std::string *filename*) [pure virtual]

8.48.2.2 `addPlugin()` virtual void GenesysShell_if::addPlugin (std::string *filename*) [pure virtual]

8.48.2.3 `check()` virtual void GenesysShell_if::check () [pure virtual]

8.48.2.4 `closeModel()` virtual void GenesysShell_if::closeModel () [pure virtual]

8.48.2.5 `createModel()` virtual void GenesysShell_if::createModel () [pure virtual]

8.48.2.6 `deleteTraceFiles()` virtual void GenesysShell_if::deleteTraceFiles () [pure virtual]

8.48.2.7 `execLinuxCommand()` virtual void GenesysShell_if::execLinuxCommand (std::string *command*) [pure virtual]

8.48.2.8 getCommandLine() virtual void GenesysShell_if::getCommandLine () [pure virtual]

8.48.2.9 getGenesysInfo() virtual void GenesysShell_if::getGenesysInfo () [pure virtual]

8.48.2.10 listComponents() virtual void GenesysShell_if::listComponents () [pure virtual]

8.48.2.11 listElements() virtual void GenesysShell_if::listElements () [pure virtual]

8.48.2.12 listHosts() virtual void GenesysShell_if::listHosts () [pure virtual]

8.48.2.13 listPlugins() virtual void GenesysShell_if::listPlugins () [pure virtual]

8.48.2.14 openModel() virtual void GenesysShell_if::openModel (std::string *filename*) [pure virtual]

8.48.2.15 readCommandsFromFile() virtual void GenesysShell_if::readCommandsFromFile (std::string *filename*) [pure virtual]

8.48.2.16 receiveFile() virtual void GenesysShell_if::receiveFile (std::string *filename*) [pure virtual]

8.48.2.17 redirectTrace() virtual void GenesysShell_if::redirectTrace (std::string *trace*, std::string *dest*, std::string *filename*) [pure virtual]

8.48.2.18 `saveModel()` virtual void GenesysShell_if::saveModel () [pure virtual]

8.48.2.19 `saveModelAs()` virtual void GenesysShell_if::saveModelAs (std::string *filename*) [pure virtual]

8.48.2.20 `sendFile()` virtual void GenesysShell_if::sendFile (std::string *filename*, std::string *hostname*, std::string *portname*) [pure virtual]

8.48.2.21 `setActivationCode()` virtual void GenesysShell_if::setActivationCode (std::string *code*) [pure virtual]

8.48.2.22 `showHelp()` virtual void GenesysShell_if::showHelp () [pure virtual]

8.48.2.23 `showHostName()` virtual void GenesysShell_if::showHostName () [pure virtual]

8.48.2.24 `showInit()` virtual void GenesysShell_if::showInit () [pure virtual]

8.48.2.25 `startSimulation()` virtual void GenesysShell_if::startSimulation () [pure virtual]

8.48.2.26 `stepSimulation()` virtual void GenesysShell_if::stepSimulation () [pure virtual]

8.48.2.27 `stopSimulation()` virtual void GenesysShell_if::stopSimulation () [pure virtual]

8.48.2.28 traceLevel() virtual void GenesysShell_if::traceLevel (Util::TraceLevel tracelevel) [pure virtual]

8.48.2.29 verboseMode() virtual void GenesysShell_if::verboseMode (bool on) [pure virtual]

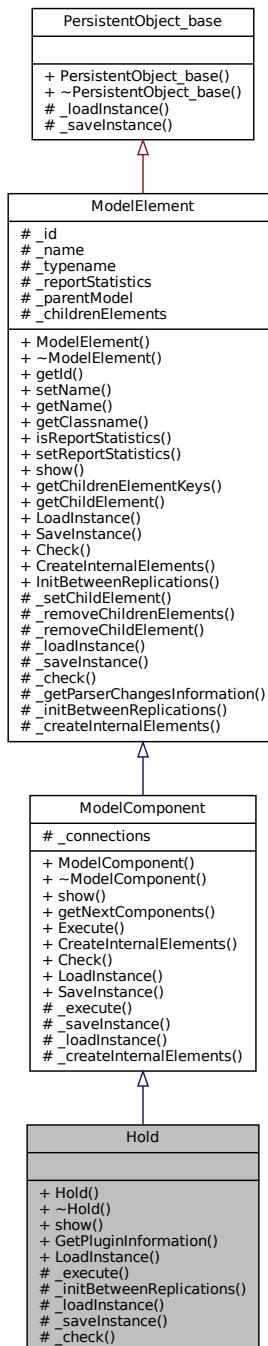
The documentation for this class was generated from the following file:

- [GenesysShell_if.h](#)

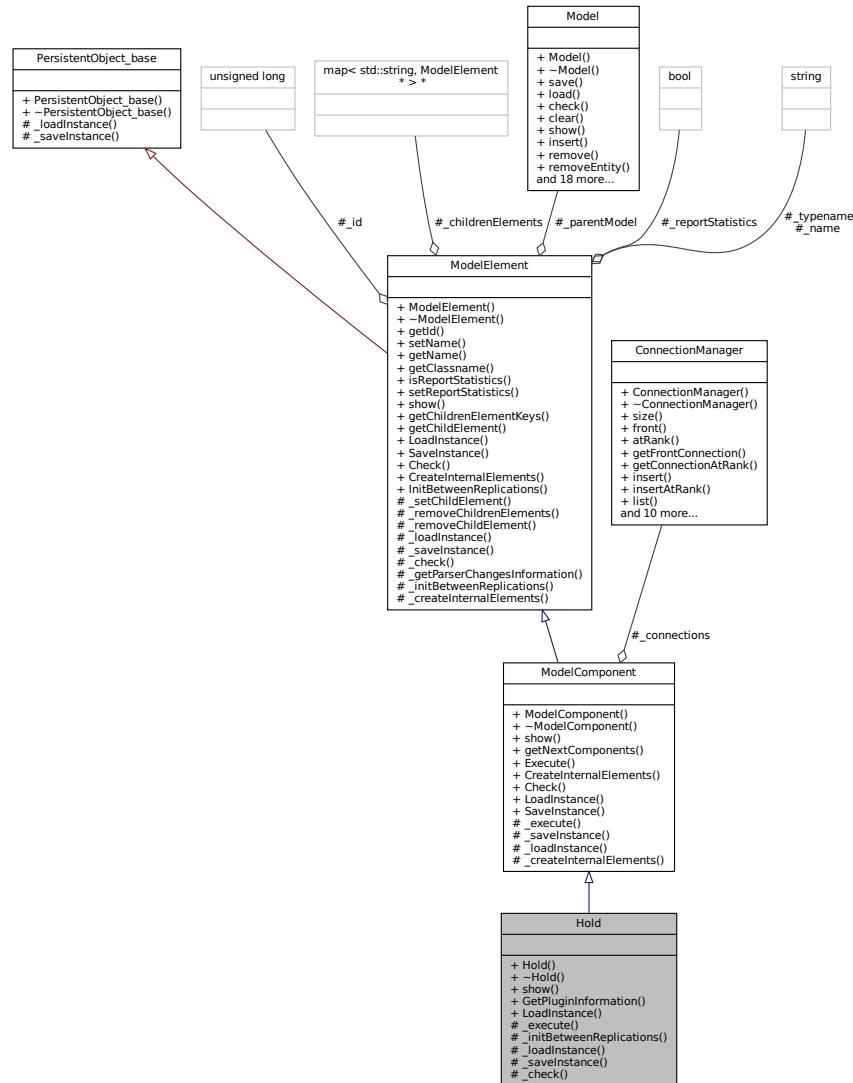
8.49 Hold Class Reference

```
#include <Hold.h>
```

Inheritance diagram for Hold:



Collaboration diagram for Hold:



Public Member Functions

- **Hold (Model *model, std::string name="")**
- virtual **~Hold ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_savemode ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.49.1 Detailed Description

Hold module DESCRIPTION This module will hold an entity in a queue to wait for a signal, wait for a specified condition to become true (scan), or be held infinitely (to be removed later with the **Remove** module). If the entity is holding for a signal, the **Signal** module is used elsewhere in the model to allow the entity to move on to the next module. If the entity is holding for a given condition to be true, the entity will remain at the module (either in a defined or internal queue) until the condition(s) becomes true. When the entity is in an infinite hold, the **Remove** module is used elsewhere in the model to allow the entity to continue processing. TYPICAL USES Waiting for a traffic light to turn green Holding a part for authorization Checking the status of a machine or operator to continue a process PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Indicates the reasoning for holding the entity within a specified or internal queue. Wait for **Signal** will hold the entity until a signal of the same value is received. Scan for Condition will hold the entity until the specified condition becomes true. Infinite **Hold** will hold the entity until it is removed from the queue by a **Remove** module. Wait for Value **Signal** code for the waiting entity. Applies only when Type is Wait for **Signal**. Limit Maximum number of waiting entities that will be released upon receipt of a signal. Applies only when Type is Wait for **Signal**. Condition Specifies the condition that will be evaluated to hold the entity at the module. If the condition is evaluated to true, the entity leaves the module immediately. If the condition is false, the entity will wait in the associated queue until the condition becomes true. Applies only when Type is Scan for Condition. Queue Type Determines the type of queue used to hold the entities. If **Queue** is selected, the queue name is specified. If **Set** is selected, the queue set and member in the set are specified. If Internal is selected, an internal queue is used to hold all waiting entities. Attribute and Expression are additional methods for defining the queue to be used. Queue Name This field is visible only if Queue Type is **Queue**, and it defines the symbol name of the queue. Set Name This field is visible only if Queue Type is **Set**, and it defines the queue set that contains the queue being referenced. Set Index This field is visible only if Queue Type is **Set**, and it defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entry for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute This field is visible only if Queue Type is **Attribute**. The attribute entered in this field will be evaluated to indicate which queue is to be used. Expression This field is visible only if Queue Type is Expression. The expression entered in this field will be evaluated to indicate which queue is to be used.

Definition at line 75 of file **Hold.h**.

8.49.2 Constructor & Destructor Documentation

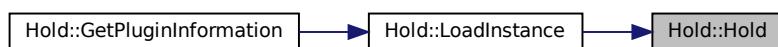
```
8.49.2.1 Hold() Hold::Hold (
    Model * model,
    std::string name = "")
```

Definition at line 17 of file **Hold.cpp**.

```
00017 : ModelComponent(model, Util::TypeOf<Hold>(), name) {
00018 }
```

Referenced by **LoadInstance()**.

Here is the caller graph for this function:



8.49.2.2 ~Hold() virtual Hold::~Hold () [virtual], [default]

8.49.3 Member Function Documentation

8.49.3.1 _check() bool Hold::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Hold.cpp](#).

```
00056     {  
00057         bool resultAll = true;  
00058         //...  
00059         return resultAll;  
00060     }
```

8.49.3.2 _execute() void Hold::_execute (Entity * entity) [protected], [virtual]

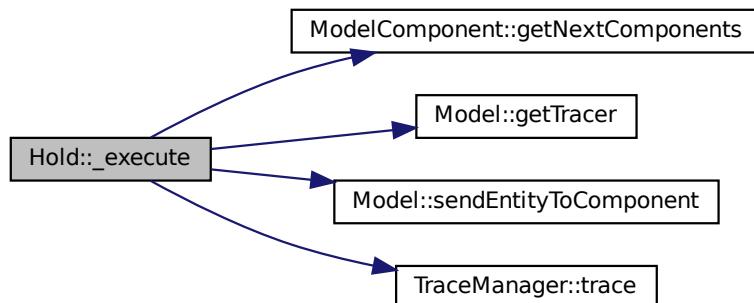
Implements [ModelComponent](#).

Definition at line 34 of file [Hold.cpp](#).

```
00034     {  
00035         _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00036         this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),  
00037         0.0);  
00037 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent\(\)](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.49.3.3 `_initBetweenReplications()` void Hold::_initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Hold.cpp](#).

```
00047
00048 }
```

8.49.3.4 `_loadInstance()` bool Hold::_loadInstance (

```
std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

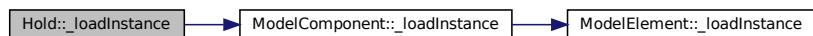
Definition at line 39 of file [Hold.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

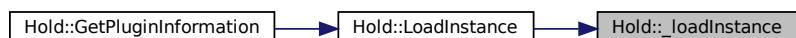
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.49.3.5 `_saveInstance()` `std::map< std::string, std::string > * Hold::_saveInstance () [protected], [virtual]`

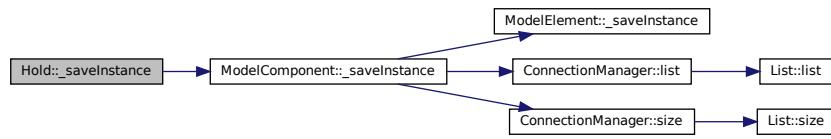
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [Hold.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.49.3.6 `GetPluginInformation()` `PluginInformation * Hold::GetPluginInformation () [static]`

Definition at line 62 of file [Hold.cpp](#).

```
00062
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Hold>(), &Hold::LoadInstance);
00064     // ...
00065     return info;
00066 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.49.3.7 LoadInstance() `ModelComponent * Hold::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

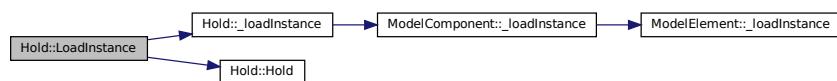
Definition at line 24 of file [Hold.cpp](#).

```
00024
00025     Hold* newComponent = new Hold(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
```

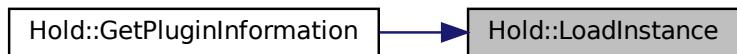
References [_loadInstance\(\)](#), and [Hold\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.49.3.8 show() `std::string Hold::show () [virtual]`

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Hold.cpp](#).

```
00020
00021     {
00022         return ModelComponent::show() + "";
00023     }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



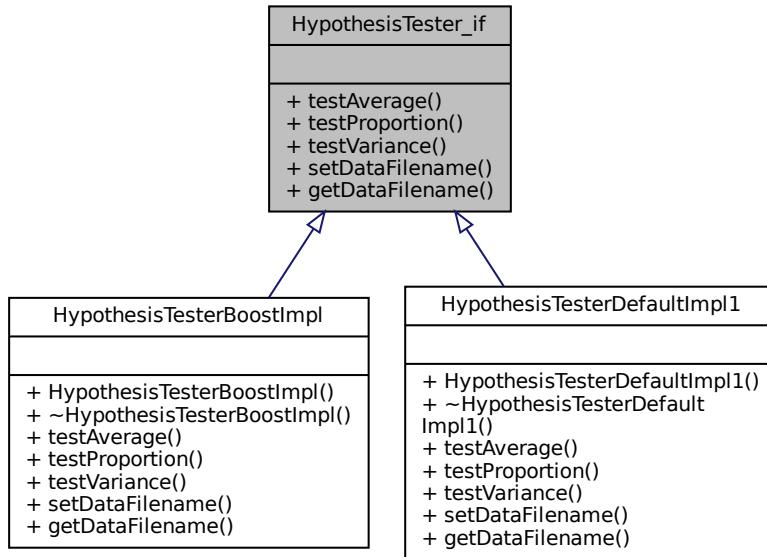
The documentation for this class was generated from the following files:

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

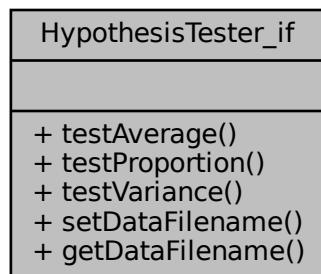
8.50 HypothesisTester_if Class Reference

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

Inheritance diagram for HypothesisTester_if:



Collaboration diagram for HypothesisTester_if:



Public Types

- enum H1Comparition { DIFFERENT = 1, LESS_THAN = 2, GREATER_THAN = 3 }

Public Member Functions

- virtual double `testAverage` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation←
DataFilename="")=0
- virtual double `testProportion` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation←
DataFilename="")=0
- virtual double `testVariance` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation←
DataFilename="")=0
- virtual void `setDataFilename` (std::string datafilename)=0
- virtual std::string `getDataFilename` ()=0

8.50.1 Detailed Description

Interface for parametric hypothesis tests based on a datafile.

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

8.50.2 Member Enumeration Documentation

8.50.2.1 `H1Comparition` enum `HypothesisTester_if::H1Comparition`

Enumerator

DIFFERENT	
LESS_THAN	
GREATER_THAN	

Definition at line 25 of file `HypothesisTester_if.h`.

```
00025      {
00026      DIFFERENT = 1,
00027      LESS_THAN = 2,
00028      GREATER_THAN = 3
00029  };
```

8.50.3 Member Function Documentation

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

Implemented in `HypothesisTesterBoostImpl`, and `HypothesisTesterDefaultImpl1`.

```
8.50.3.2 setDataFilename() virtual void HypothesisTester_if::setDataFilename (
    std::string datafilename ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

```
8.50.3.3 testAverage() virtual double HypothesisTester_if::testAverage (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDatafilename = "" ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

```
8.50.3.4 testProportion() virtual double HypothesisTester_if::testProportion (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDatafilename = "" ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

```
8.50.3.5 testVariance() virtual double HypothesisTester_if::testVariance (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDatafilename = "" ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

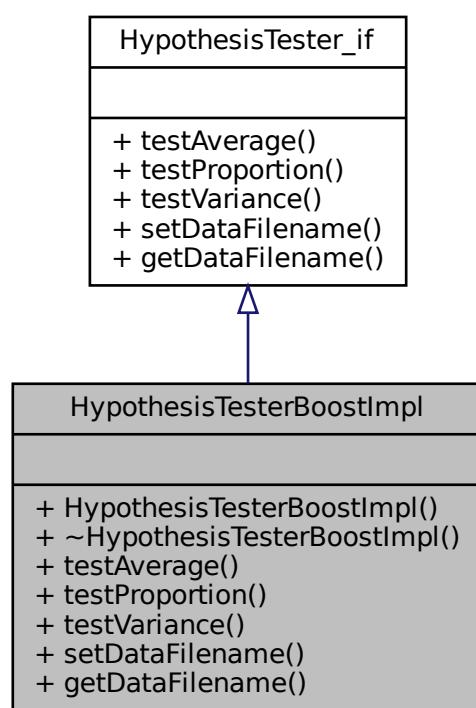
The documentation for this class was generated from the following file:

- [HypothesisTester_if.h](#)

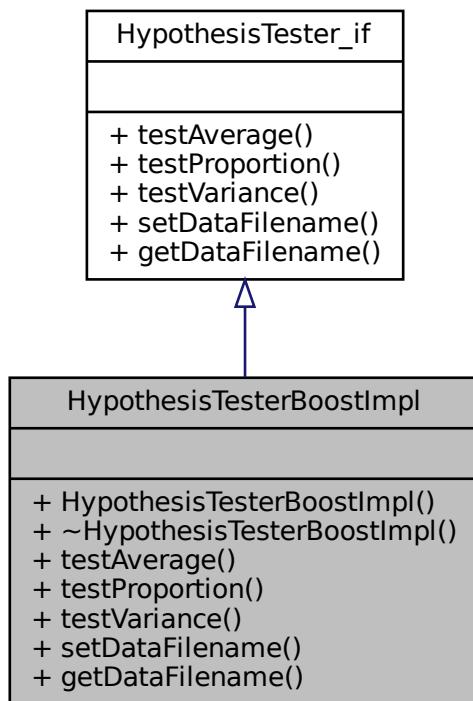
8.51 HypothesisTesterBoostImpl Class Reference

```
#include <HypothesisTesterBoostImpl.h>
```

Inheritance diagram for HypothesisTesterBoostImpl:



Collaboration diagram for HypothesisTesterBoostImpl:



Public Member Functions

- `HypothesisTesterBoostImpl ()`
- virtual `~HypothesisTesterBoostImpl ()=default`
- virtual double `testAverage` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation↔ DataFilename="")
- virtual double `testProportion` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation↔ DataFilename="")
- virtual double `testVariance` (double confidencelevel, `H1Comparition` comp, std::string secondPopulation↔ DataFilename="")
- virtual void `setDataFilename` (std::string dataFilename)
- virtual std::string `getDataFilename` ()

Additional Inherited Members

8.51.1 Detailed Description

Definition at line 20 of file [HypothesisTesterBoostImpl.h](#).

8.51.2 Constructor & Destructor Documentation

8.51.2.1 HypothesisTesterBoostImpl() HypothesisTesterBoostImpl::HypothesisTesterBoostImpl ()

Definition at line 16 of file [HypothesisTesterBoostImpl.cpp](#).

```
00016 {  
00017 }
```

8.51.2.2 ~HypothesisTesterBoostImpl() virtual HypothesisTesterBoostImpl::~HypothesisTesterBoostImpl () [virtual], [default]**8.51.3 Member Function Documentation****8.51.3.1 getDataFilename()** std::string HypothesisTesterBoostImpl::getDataFilename () [virtual]

Implements [HypothesisTester_if](#).

Definition at line 36 of file [HypothesisTesterBoostImpl.cpp](#).

```
00036 {  
00037     return ""; // \todo not implemented yet  
00038 }
```

8.51.3.2 setDataFilename() void HypothesisTesterBoostImpl::setDataFilename (std::string dataFilename) [virtual]

Implements [HypothesisTester_if](#).

Definition at line 32 of file [HypothesisTesterBoostImpl.cpp](#).

```
00032 {  
00033     // \todo not implemented yet  
00034 }
```

8.51.3.3 testAverage() double HypothesisTesterBoostImpl::testAverage (double confidencelevel, H1Comparition comp, std::string secondPopulationDataFilename = "") [virtual]

Implements [HypothesisTester_if](#).

Definition at line 19 of file [HypothesisTesterBoostImpl.cpp](#).

```
00019 {  
00020     // \todo not implemented yet  
00021 }
```

```
8.51.3.4 testProportion() double HypothesisTesterBoostImpl::testProportion (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDataFilename = "") [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 23 of file [HypothesisTesterBoostImpl.cpp](#).

```
00023
00024     // \todo not implemented yet
00025 }
```

```
8.51.3.5 testVariance() double HypothesisTesterBoostImpl::testVariance (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDataFilename = "") [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 27 of file [HypothesisTesterBoostImpl.cpp](#).

```
00027
00028     // \todo not implemented yet
00029 }
```

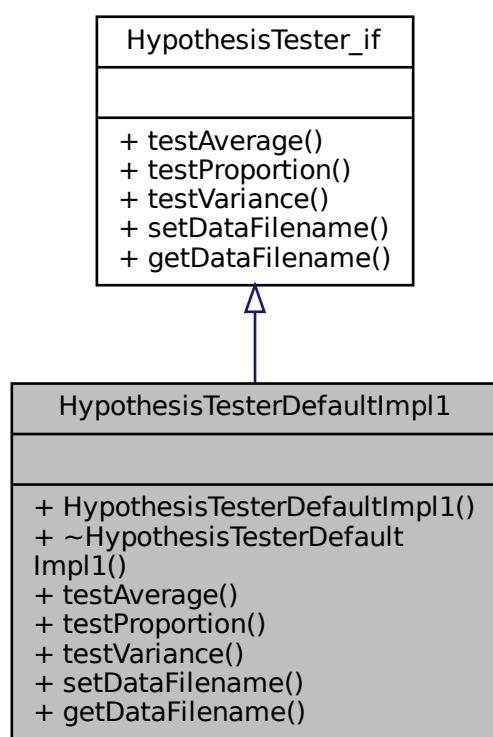
The documentation for this class was generated from the following files:

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

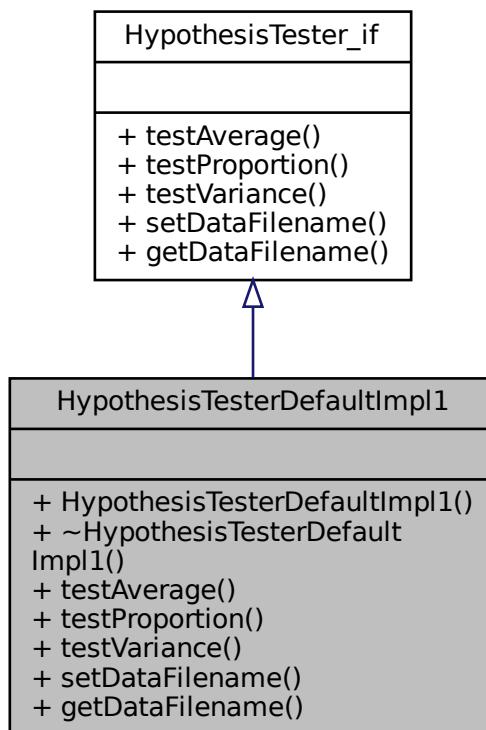
8.52 HypothesisTesterDefaultImpl1 Class Reference

```
#include <HypothesisTesterDefaultImpl1.h>
```

Inheritance diagram for HypothesisTesterDefaultImpl1:



Collaboration diagram for HypothesisTesterDefaultImpl1:



Public Member Functions

- [HypothesisTesterDefaultImpl1 \(\)](#)
- virtual [~HypothesisTesterDefaultImpl1 \(\)=default](#)
- virtual double [testAverage \(double confidencelevel, H1Comparition comp, std::string secondPopulation, std::string DataFilename=""\)](#)
- virtual double [testProportion \(double confidencelevel, H1Comparition comp, std::string secondPopulation, std::string DataFilename=""\)](#)
- virtual double [testVariance \(double confidencelevel, H1Comparition comp, std::string secondPopulation, std::string DataFilename=""\)](#)
- virtual void [setDataFilename \(std::string dataFilename\)](#)
- virtual std::string [getDataFilename \(\)](#)

Additional Inherited Members

8.52.1 Detailed Description

Definition at line 20 of file [HypothesisTesterDefaultImpl1.h](#).

8.52.2 Constructor & Destructor Documentation

8.52.2.1 HypothesisTesterDefaultImpl1() HypothesisTesterDefaultImpl1::HypothesisTesterDefaultImpl1 ()

Definition at line 17 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00017
00018     _integrator = new Traits<HypothesisTester_if>::IntegratorImplementation();
00019 }
```

8.52.2.2 ~HypothesisTesterDefaultImpl1() virtual HypothesisTesterDefaultImpl1::~HypothesisTesterDefaultImpl1 () [virtual], [default]

8.52.3 Member Function Documentation

8.52.3.1 getDataFilename() std::string HypothesisTesterDefaultImpl1::getDataFilename () [virtual]

Implements [HypothesisTester_if](#).

Definition at line 38 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00038
00039     return ""; // \todo not implemented yet
00040 }
```

8.52.3.2 setDataFilename() void HypothesisTesterDefaultImpl1::setDataFilename (std::string dataFilename) [virtual]

Implements [HypothesisTester_if](#).

Definition at line 34 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00034
00035     // \todo not implemented yet
00036 }
```

8.52.3.3 testAverage() double HypothesisTesterDefaultImpl1::testAverage (double confidencelevel, H1Comparition comp, std::string secondPopulationDataFilename = "") [virtual]

Implements [HypothesisTester_if](#).

Definition at line 21 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00021
00022     // \todo not implemented yet
00023 }
```

```
8.52.3.4 testProportion() double HypothesisTesterDefaultImpl1::testProportion (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDataFilename = "") [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 25 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00025
00026     // \todo not implemented yet
00027 }
```

```
8.52.3.5 testVariance() double HypothesisTesterDefaultImpl1::testVariance (
    double confidencelevel,
    H1Comparition comp,
    std::string secondPopulationDataFilename = "") [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 29 of file [HypothesisTesterDefaultImpl1.cpp](#).

```
00029
00030     // \todo not implemented yet
00031 }
```

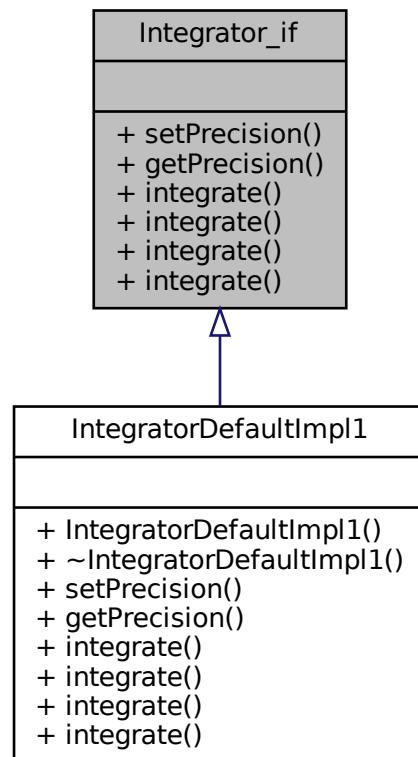
The documentation for this class was generated from the following files:

- [HypothesisTesterDefaultImpl1.h](#)
- [HypothesisTesterDefaultImpl1.cpp](#)

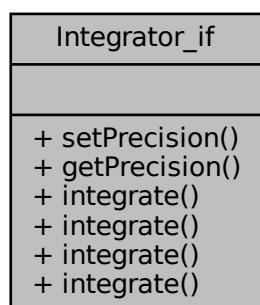
8.53 Integrator_if Class Reference

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

Inheritance diagram for Integrator_if:



Collaboration 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](#)

8.53.1 Detailed Description

Interface used by classes that perform the numerical integration of functions with one to four parameters. It is mainly used for calculating the probability of theoretical distributions, from its probability distribution functions.

Definition at line 21 of file [Integrator_if.h](#).

8.53.2 Member Function Documentation

8.53.2.1 [getPrecision\(\)](#) virtual double Integrator_if::getPrecision () [pure virtual]

Implemented in [IntegratorDefaultImpl1](#).

8.53.2.2 [integrate\(\) \[1/4\]](#) virtual double Integrator_if::integrate (double min, double max, double(*) (double, double) f, double p2) [pure virtual]

Implemented in [IntegratorDefaultImpl1](#).

8.53.2.3 [integrate\(\) \[2/4\]](#) virtual double Integrator_if::integrate (double min, double max, double(*) (double, double, double) f, double p2, double p3) [pure virtual]

Implemented in [IntegratorDefaultImpl1](#).

```
8.53.2.4 integrate() [3/4] 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](#).

```
8.53.2.5 integrate() [4/4] 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](#).

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

Implemented in [IntegratorDefaultImpl1](#).

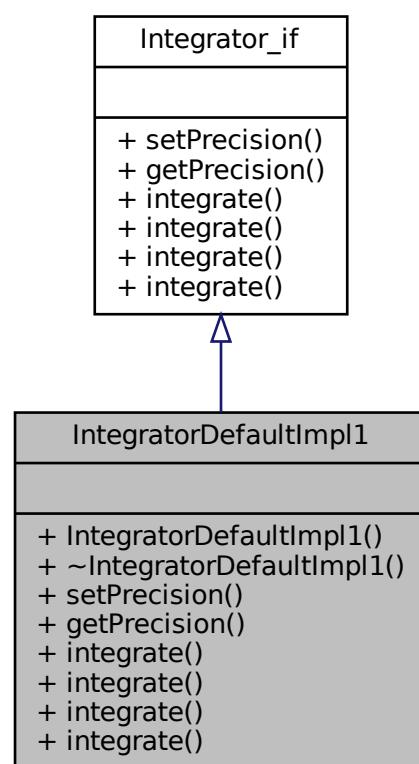
The documentation for this class was generated from the following file:

- [Integrator_if.h](#)

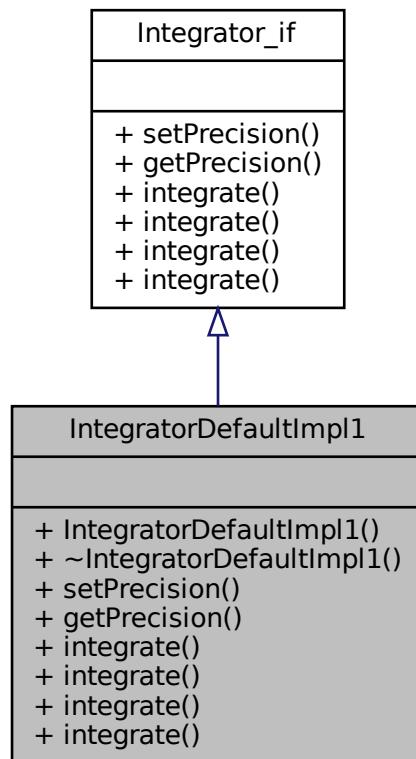
8.54 IntegratorDefaultImpl1 Class Reference

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

Inheritance diagram for IntegratorDefaultImpl1:



Collaboration diagram for IntegratorDefaultImpl1:



Public Member Functions

- `IntegratorDefaultImpl1 ()`
`https://codereview.stackexchange.com/questions/200289/implementing-numerical-integration`
- `virtual ~IntegratorDefaultImpl1 ()=default`
- `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)`

8.54.1 Detailed Description

Definition at line 19 of file [IntegratorDefaultImpl1.h](#).

8.54.2 Constructor & Destructor Documentation

8.54.2.1 IntegratorDefaultImpl1() IntegratorDefaultImpl1::IntegratorDefaultImpl1 ()

<https://codereview.stackexchange.com/questions/200289/implementing-numerical-integration-with-cpp>

Definition at line 19 of file IntegratorDefaultImpl1.cpp.

```
00019     this->_precision = Traits<Integrator_if>::Precision;
00020 }
00021 }
```

8.54.2.2 ~IntegratorDefaultImpl1() virtual IntegratorDefaultImpl1::~IntegratorDefaultImpl1 () [virtual], [default]**8.54.3 Member Function Documentation****8.54.3.1 getPrecision()** double IntegratorDefaultImpl1::getPrecision () [virtual]

Implements [Integrator_if](#).

Definition at line 27 of file IntegratorDefaultImpl1.cpp.

```
00027 {
00028     return _precision;
00029 }
```

8.54.3.2 integrate() [1/4] double IntegratorDefaultImpl1::integrate (double min, double max, double(*) (double, double) f, double p2) [virtual]

Implements [Integrator_if](#).

Definition at line 31 of file IntegratorDefaultImpl1.cpp.

```
00031 {
00032     //Rosetta:::GaussLegendreQuadrature<5>();
00033     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00034     double x, y, h = (max - min) / m;
00035     double I1 = 0.0;
00036     for (unsigned int j = 1; j <= m + 1; j++) {
00037         i = j - 1;
00038         if (i == 0 || i == m)
00039             c = 1;
00040         else
00041             c = 2;
00042         x = min + i*h;
00043         y = f(x, p2);
00044         I1 += c*y;
00045     }
00046     return (h / 2)*I1;
00047 }
```

8.54.3.3 integrate() [2/4] double IntegratorDefaultImpl1::integrate (

```
    double min,
    double max,
    double(*)(double, double, double) f,
    double p2,
    double p3 )  [virtual]
```

Implements [Integrator_if](#).

Definition at line 49 of file [IntegratorDefaultImpl1.cpp](#).

```
00049      {
00050      unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00051      double x, y, h = (max - min) / m;
00052      double I1 = 0.0;
00053      for (unsigned int j = 1; j <= m + 1; j++) {
00054          i = j - 1;
00055          if (i == 0 || i == m)
00056              c = 1;
00057          else
00058              c = 2;
00059          x = min + i*h;
00060          y = f(x, p2, p3);
00061          I1 += c*y;
00062      }
00063      return (h / 2)*I1;
00064 }
```

8.54.3.4 integrate() [3/4] double IntegratorDefaultImpl1::integrate (

```
    double min,
    double max,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 )  [virtual]
```

Implements [Integrator_if](#).

Definition at line 66 of file [IntegratorDefaultImpl1.cpp](#).

```
00066      {
00067      unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00068      double x, y, h = (max - min) / m;
00069      double I1 = 0.0;
00070      for (unsigned int j = 1; j <= m + 1; j++) {
00071          i = j - 1;
00072          if (i == 0 || i == m)
00073              c = 1;
00074          else
00075              c = 2;
00076          x = min + i*h;
00077          y = f(x, p2, p3, p4);
00078          I1 += c*y;
00079      }
00080      return (h / 2)*I1;
00081 }
```

8.54.3.5 integrate() [4/4] 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](#).

Definition at line 83 of file [IntegratorDefaultImpl1.cpp](#).

```
00083
00084     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00085     double x, y, h = (max - min) / m;
00086     double I1 = 0.0;
00087     for (unsigned int j = 1; j <= m + 1; j++) {
00088         i = j - 1;
00089         if (i == 0 || i == m)
00090             c = 1;
00091         else
00092             c = 2;
00093         x = min + i*h;
00094         y = f(x, p2, p3, p4, p5);
00095         I1 += c*y;
00096     }
00097     return (h / 2)*I1;
00098 }
```

8.54.3.6 setPrecision() void IntegratorDefaultImpl1::setPrecision (

```
    double e )  [virtual]
```

Implements [Integrator_if](#).

Definition at line 23 of file [IntegratorDefaultImpl1.cpp](#).

```
00023
00024     _precision = e;
00025 }
```

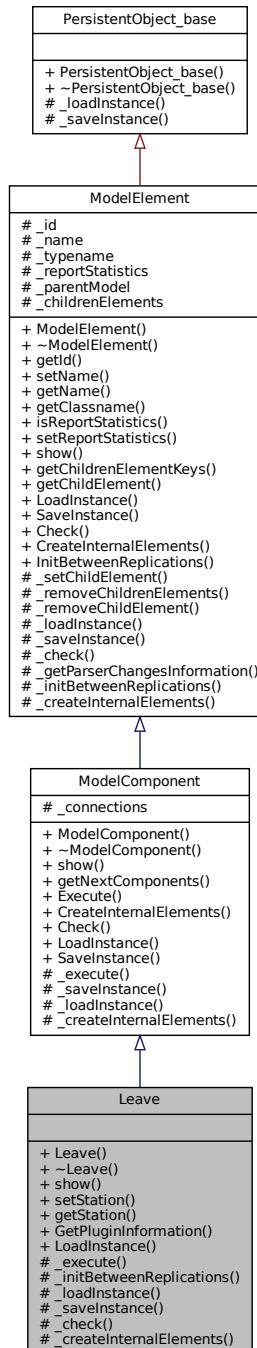
The documentation for this class was generated from the following files:

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

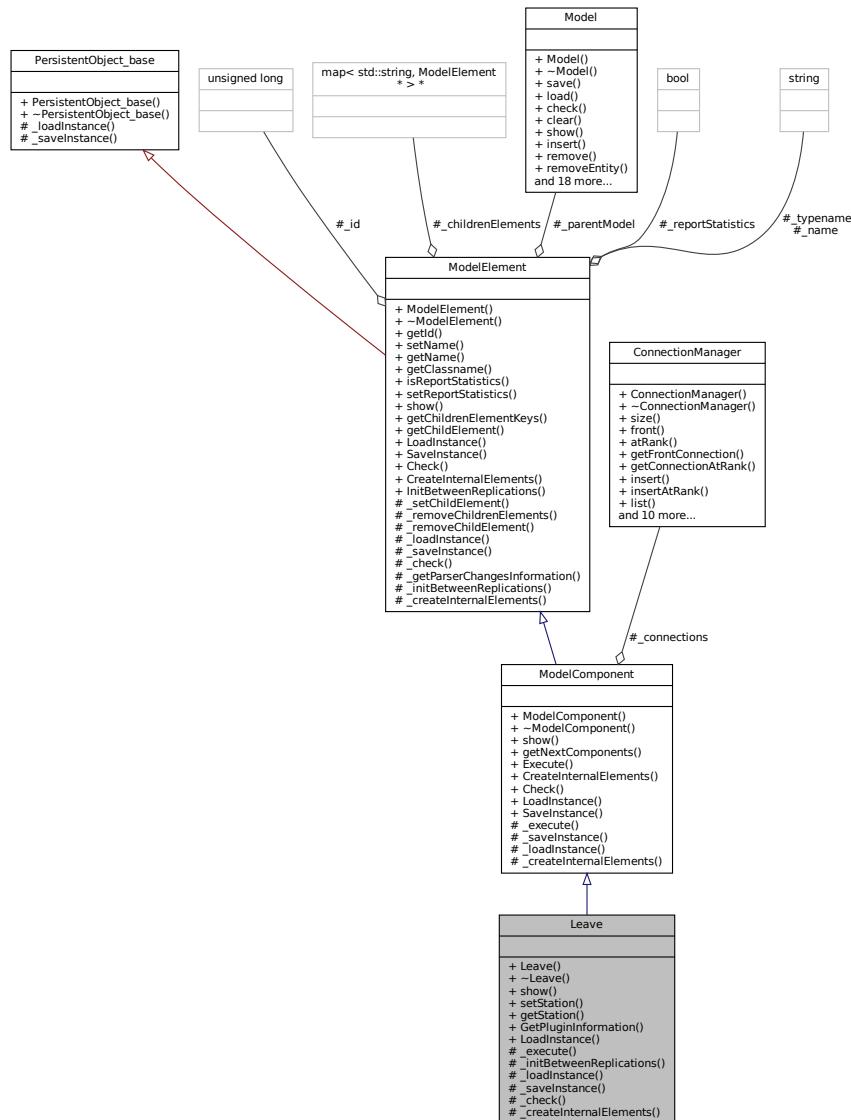
8.55 Leave Class Reference

```
#include <Leave.h>
```

Inheritance diagram for Leave:



Collaboration diagram for Leave:



Public Member Functions

- **Leave** (**Model** *model, std::string name="")
- virtual **~Leave** ()=default
- virtual std::string **show** ()
- void **setStation** (**Station** *_station)
- **Station** * **getStation** () const

Static Public Member Functions

- static **PluginInformation** * **GetPluginInformation** ()
- static **ModelComponent** * **LoadInstance** (**Model** *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.55.1 Detailed Description

Leave module DESCRIPTION The **Leave** module is used to transfer an entity to a station or module. An entity may be transferred in two ways. It can be transferred to a module that defines a station by referencing the station and routing, conveying, or transporting to that station, or a graphical connection can be used to transfer an entity to another module. When an entity arrives at a **Leave** module, it may wait to obtain a transfer device (resource, transporter, or conveyor). When the transfer device has been obtained, the entity may experience a loading delay. Finally, the entity is transferred from this module to a destination module or station. TYPICAL USES The end of a part's production in a series of parallel processes where the part needs a forklift to be transferred to shipping PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Allocation Type of category to which the entity's incurred delay time and cost will be added. **Delay** Specifies a load time incurred after getting a transfer device. Units Time units used for the delay time. Transfer Out Determines whether a resource (**Seize Resource**), transporter (**Request Transporter**), or conveyor (**Access Conveyor**) is required prior to transferring the entity out of this module. Priority Indicates the priority of the module when either seizing a resource or requesting a transporter when there are entities waiting for that resource/transporter from other modules. This field is not visible when the Transfer Type is None or **Access Conveyor**. Transporter Name Name of the transporter to request. **Queue** Type Type of queue, either a single **Queue**, queue **Set**, Internal queue, **Attribute**, or Expression. **Queue** Name Name of the individual queue. **Queue Set** Name Name of the queue set. **Set Index** Defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. **Queue Attribute** Name The attribute name that will be evaluated to indicate which queue is to be used. **Queue** Expression The expression that will be evaluated to indicate which queue is to be used. Selection Rule Method of selecting among available transporters in a set. Cyclical will cycle through available members. Random will randomly select a member. Preferred Order will always select the first available member. Specific Member requires an input attribute value to specify which member of the set (previously saved in the Save **Attribute** field). Largest Distance selects the transporter farthest away, and Smallest Distance selects the closest transporter. Save **Attribute** Attribute name used to store the index number into the set of the member that is chosen. This attribute can later be referenced with the Specific Member selection rule. Active when Transfer Out is Request Transporter. Index **Set Attribute** name whose value identifies the index number into the set of the member requested. The entity must have a value for the attribute before utilizing this option. **Resource** Type Type of resource for seizing, either specifying a particular **Resource**, selecting from a pool of resources (that is, a resource **Set**), **Attribute**, or Expression. **Resource** Name Name of the resource to seize. Conveyor Name Name of the conveyor to access.

of Cells Number of contiguous cells the entity requires.

Connect Type Determines if the entity is to **Route**, Convey, or Transport to another station or Connect to another module. Move Time Time to route from this module to the destination station. Units Time units used for the move time. **Station** Type The entity's destination station type either an individual **Station**, a station based on an **Attribute** or Expression value, or By **Sequence**. Station Name Name of the individual destination station. **Attribute** Name The attribute name that will be evaluated to indicate the station. Expression The expression that will be evaluated to indicate the station.

Definition at line 94 of file [Leave.h](#).

8.55.2 Constructor & Destructor Documentation

8.55.2.1 Leave() `Leave::Leave (`
 `Model * model,`
 `std::string name = "")`

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

```
00017 : ModelComponent(model, Util::TypeOf<Leave>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.55.2.2 ~Leave() `virtual Leave::~Leave () [virtual], [default]`

8.55.3 Member Function Documentation

8.55.3.1 _check() `bool Leave::_check (`
 `std::string * errorMessage) [protected], [virtual]`

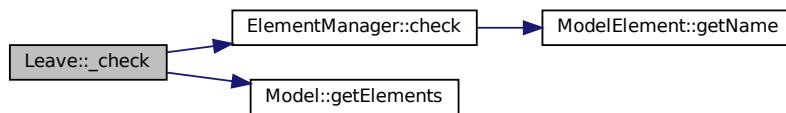
Reimplemented from [ModelElement](#).

Definition at line 69 of file [Leave.cpp](#).

```
00069 {  
00070     bool resultAll = true;  
00071     resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",  
00072         errorMessage);  
00073 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



8.55.3.2 `_createInternalElements()` void Leave::_createInternalElements () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

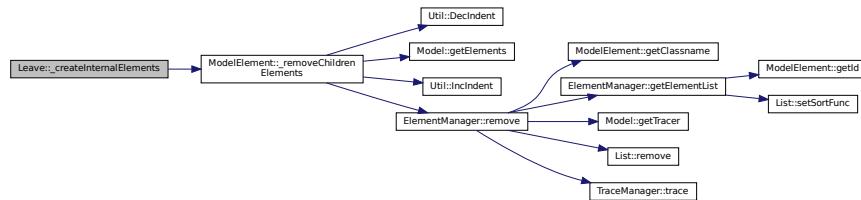
Reimplemented from [ModelComponent](#).

Definition at line 81 of file [Leave.cpp](#).

```
00081     if (_reportStatistics)
00082         if (_numberIn == nullptr) {
00083             _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00084             _childrenElements->insert({"CountNumberIn", _numberIn});
00085         } else
00086             if (_numberIn != nullptr)
00087                 _removeChildrenElements();
00088         }
00089     }
00090 }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



8.55.3.3 `_execute()` void Leave::_execute (Entity * entity) [protected], [virtual]

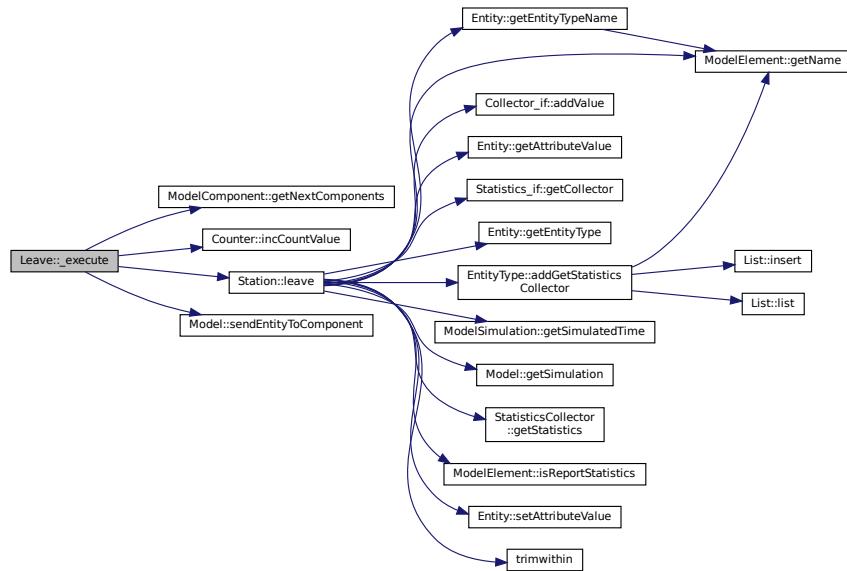
Implements [ModelComponent](#).

Definition at line 42 of file [Leave.cpp](#).

```
00042     if (_reportStatistics)
00043         _numberIn->incCountValue();
00044     _station->leave(entity);
00045     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00046 }
00047 }
```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [ModelComponent::getNextComponents\(\)](#), [Counter::incCountValue\(\)](#), [Station::leave\(\)](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.55.3.4 `_initBetweenReplications()`

```
void Leave::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 59 of file [Leave.cpp](#).

```
00059     {
00060         _numberIn->clear();
00061     }
```

References [Counter::clear\(\)](#).

Here is the call graph for this function:



```
8.55.3.5 _loadInstance() bool Leave::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

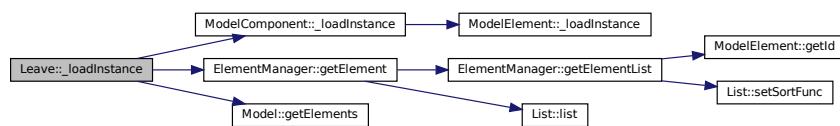
Definition at line 49 of file [Leave.cpp](#).

```
00049
00050     bool res = ModelComponent::_loadInstance(fields);
00051     if (res) {
00052         std::string stationName = ((*fields->find("stationName"))).second;
00053         Station* station = dynamic_cast<Station*>
00054             (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00055         this->station = station;
00056     }
00057     return res;
00058 }
```

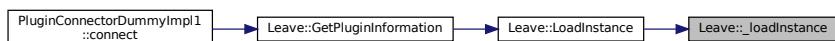
References [ModelComponent::_loadInstance\(\)](#), [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), and [Model::getElements\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.55.3.6 _saveInstance() std::map< std::string, std::string > * Leave::_saveInstance ( ) [protected], [virtual]
```

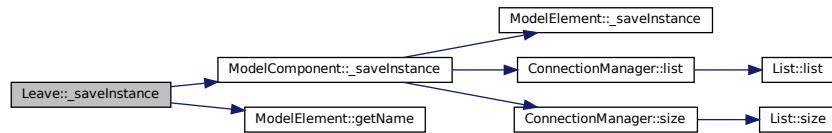
Reimplemented from [ModelComponent](#).

Definition at line 63 of file [Leave.cpp](#).

```
00063
00064     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00065     fields->emplace("stationName", "\"" + (this->station->getName()) + "\"");
00066     return fields;
00067 }
```

References [ModelComponent::_saveInstance\(\)](#), and [ModelElement::getName\(\)](#).

Here is the call graph for this function:



8.55.3.7 GetPluginInformation() [PluginInformation * Leave::GetPluginInformation \(\) \[static\]](#)

Definition at line 75 of file [Leave.cpp](#).

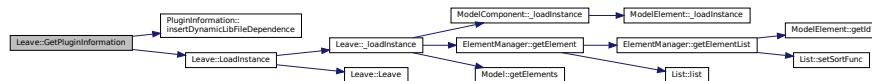
```

00075     {
00076     PluginInformation* info = new PluginInformation(Util::TypeOf<Leave>(), &Leave::LoadInstance);
00077     info->insertDynamicLibFileDependence("station.so");
00078     return info;
00079 }
  
```

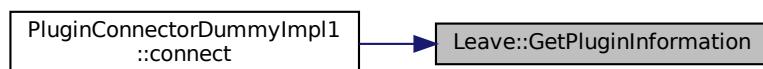
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), and [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.55.3.8 getStation() [Station * Leave::getStation \(\) const](#)

Definition at line 38 of file [Leave.cpp](#).

```

00038     {
00039     return _station;
00040 }
  
```

8.55.3.9 LoadInstance()

```
ModelComponent * Leave::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

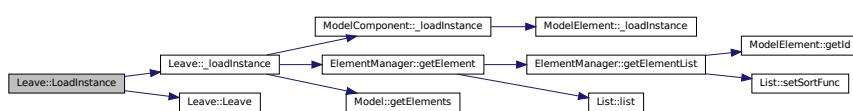
Definition at line 24 of file [Leave.cpp](#).

```
00024
00025     Leave* newComponent = new Leave(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
```

References [_loadInstance\(\)](#), and [Leave\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.55.3.10 setStation()

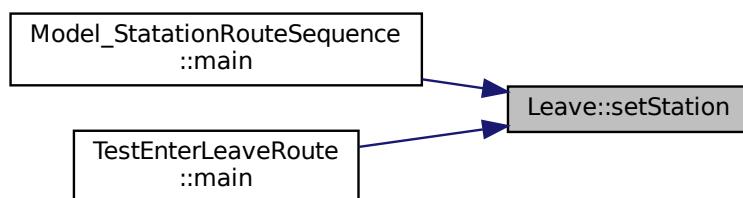
```
void Leave::setStation (
    Station * _station )
```

Definition at line 34 of file [Leave.cpp](#).

```
00034
00035     this->_station = _station;
00036 }
```

Referenced by [Model_StatationRouteSequence::main\(\)](#), and [TestEnterLeaveRoute::main\(\)](#).

Here is the caller graph for this function:



8.55.3.11 show() std::string Leave::show () [virtual]

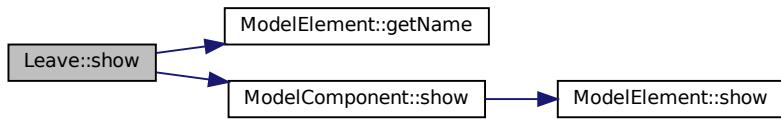
Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Leave.cpp](#).

```
00020     {
00021         return ModelComponent::show() + ",station=" + this->_station->getName();
00022     }
```

References [ModelElement::getName\(\)](#), and [ModelComponent::show\(\)](#).

Here is the call graph for this function:



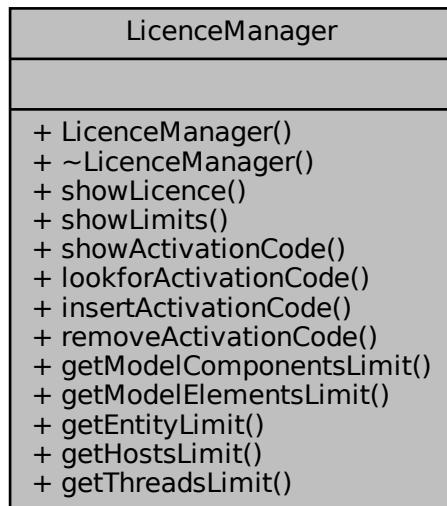
The documentation for this class was generated from the following files:

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

8.56 LicenceManager Class Reference

```
#include <LicenceManager.h>
```

Collaboration diagram for LicenceManager:



Public Member Functions

- `LicenceManager (Simulator *simulator)`
- `virtual ~LicenceManager ()=default`
- `const std::string showLicence () const`
- `const std::string showLimits () const`
- `const std::string showActivationCode () const`
- `bool lookforActivationCode ()`
- `bool insertActivationCode ()`
- `void removeActivationCode ()`
- `unsigned int getModelComponentsLimit ()`
- `unsigned int getModelElementsLimit ()`
- `unsigned int getEntityLimit ()`
- `unsigned int getHostsLimit ()`
- `unsigned int getThreadsLimit ()`

8.56.1 Detailed Description

Definition at line 22 of file [LicenceManager.h](#).

8.56.2 Constructor & Destructor Documentation

8.56.2.1 `LicenceManager()` `LicenceManager:::LicenceManager (Simulator * simulator)`

Definition at line 18 of file [LicenceManager.cpp](#).

```
00018     {  
00019         _simulator = simulator;  
00020         this->setDefaultLicenceAndLimits();  
00021     }
```

8.56.2.2 `~LicenceManager()` `virtual LicenceManager:::~LicenceManager () [virtual], [default]`

8.56.3 Member Function Documentation

8.56.3.1 `getEntityLimit()` `unsigned int LicenceManager:::getEntityLimit ()`

Definition at line 74 of file [LicenceManager.cpp](#).

```
00074     {  
00075         return this->_entities;  
00076     }
```

8.56.3.2 getHostsLimit() `unsigned int LicenceManager::getHostsLimit ()`

Definition at line 78 of file [LicenceManager.cpp](#).

```
00078     {  
00079         return this->_hosts;  
00080     }
```

8.56.3.3 getModelComponentsLimit() `unsigned int LicenceManager::getModelComponentsLimit ()`

Definition at line 66 of file [LicenceManager.cpp](#).

```
00066     {  
00067         return this->_components;  
00068     }
```

Referenced by [ModelCheckerDefaultImpl1::checkLimits\(\)](#).

Here is the caller graph for this function:

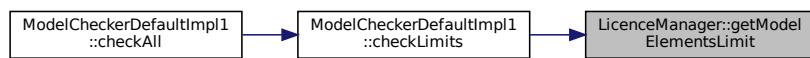
**8.56.3.4 getModelElementsLimit()** `unsigned int LicenceManager::getModelElementsLimit ()`

Definition at line 70 of file [LicenceManager.cpp](#).

```
00070     {  
00071         return this->_elements;  
00072     }
```

Referenced by [ModelCheckerDefaultImpl1::checkLimits\(\)](#).

Here is the caller graph for this function:

**8.56.3.5 getThreadsLimit()** `unsigned int LicenceManager::getThreadsLimit ()`

Definition at line 82 of file [LicenceManager.cpp](#).

```
00082     {  
00083         return this->_threads;  
00084     }
```

8.56.3.6 insertActivationCode() bool LicenceManager::insertActivationCode ()

Definition at line 57 of file LicenceManager.cpp.

```
00057 {  
00058     // \todo: Not implemented yet  
00059     return false;  
00060 }
```

8.56.3.7 lookforActivationCode() bool LicenceManager::lookforActivationCode ()

Definition at line 52 of file LicenceManager.cpp.

```
00052 {  
00053     // \todo: Not implemented yet  
00054     return false;  
00055 }
```

8.56.3.8 removeActivationCode() void LicenceManager::removeActivationCode ()

Definition at line 62 of file LicenceManager.cpp.

```
00062 {  
00063     this->setDefaultLicenceAndLimits();  
00064 }
```

8.56.3.9 showActivationCode() const std::string LicenceManager::showActivationCode () const

Definition at line 47 of file LicenceManager.cpp.

```
00047 {  
00048     std::string msg = "ACTIVATION CODE: Not found.";  
00049     return msg;  
00050 }
```

8.56.3.10 showLicence() const std::string LicenceManager::showLicence () const

Definition at line 33 of file LicenceManager.cpp.

```
00033 {  
00034     return _licence;  
00035 }
```

Referenced by [Simulator::Simulator\(\)](#).

Here is the caller graph for this function:



8.56.3.11 showLimits() const std::string LicenceManager::showLimits () const

Definition at line 37 of file [LicenceManager.cpp](#).

```
00037      {
00038         std::string msg = "LIMITS: Based on your licence and activation code, your simulator is running
00039             under the following limits" +
00040             std::string(": ") + std::to_string(_components) + " components" +
00041             ", " + std::to_string(_elements) + " elements" +
00042             ", " + std::to_string(_entities) + " entities" +
00043             ", " + std::to_string(_hosts) + " hosts" +
00044             ", " + std::to_string(_threads) + " threads.";
00045     return msg;
00046 }
```

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

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

8.57 LinkedBy Class Reference

```
#include <LinkedBy.h>
```

Collaboration diagram for LinkedBy:

LinkedBy
+ LinkedBy() + LinkedBy() + ~LinkedBy() + addLink() + removeLink() + isLinked()

Public Member Functions

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

8.57.1 Detailed Description

Definition at line 17 of file [LinkedBy.h](#).

8.57.2 Constructor & Destructor Documentation

8.57.2.1 **LinkedBy()** [1/2] `LinkedBy::LinkedBy ()`

Definition at line 16 of file [LinkedBy.cpp](#).

```
00016          {  
00017 }
```

8.57.2.2 **LinkedBy()** [2/2] `LinkedBy::LinkedBy (const LinkedBy & orig)`

Definition at line 19 of file [LinkedBy.cpp](#).

```
00019          {  
00020 }
```

8.57.2.3 **~LinkedBy()** `LinkedBy::~LinkedBy () [virtual]`

Definition at line 22 of file [LinkedBy.cpp](#).

```
00022          {  
00023 }
```

8.57.3 Member Function Documentation

8.57.3.1 **addLink()** `void LinkedBy::addLink ()`

Definition at line 25 of file [LinkedBy.cpp](#).

```
00025          {  
00026     this->_linkedBy++;  
00027 }
```

8.57.3.2 **isLinked()** `bool LinkedBy::isLinked ()`

Definition at line 33 of file [LinkedBy.cpp](#).

```
00033          {  
00034     return this->_linkedBy > 0;  
00035 }
```

8.57.3.3 removeLink() void LinkedBy::removeLink ()

Definition at line 29 of file [LinkedBy.cpp](#).

```
00029     {
00030     this->_linkedBy--;
00031 }
```

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

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

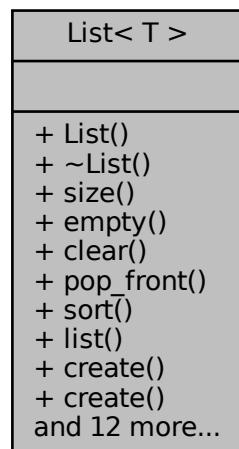
8.58 List< T > Class Template Reference

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

Inheritance diagram for List< T >:



Collaboration diagram for List< T >:



Public Types

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

Public Member Functions

- `List()`
- virtual `~List()=default`
- `unsigned int size()`
- `bool empty()`
- `void clear()`
- `void pop_front()`
- template<class Compare>
 `void sort(Compare comp)`
- `std::list<T>* list() 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)`
- `void setAtRank(unsigned int rank, T element)`
- `T getAtRank(unsigned int rank)`
- `T next()`
- `T front()`
- `T last()`
- `T previous()`
- `T current()`
- `void setSortFunc(CompFunct _sortFunc)`

8.58.1 Detailed Description

```
template<typename T>
class List<T>
```

`List` corresponds to an extended version of the list that must guarantee the consistency of the elements that make up the simulation model.

Definition at line 32 of file [List.h](#).

8.58.2 Member Typedef Documentation

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

Definition at line 34 of file [List.h](#).

8.58.3 Constructor & Destructor Documentation

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

Definition at line 74 of file [List.h](#).

```
00074         {
00075     // _map = new std::map<Util::identification, T>();
00076     _list = new std::list<T>();
00077     _it = _list->begin();
00078 }
```

8.58.3.2 ~List() template<typename T >
`virtual List< T >::~List () [virtual], [default]`

8.58.4 Member Function Documentation

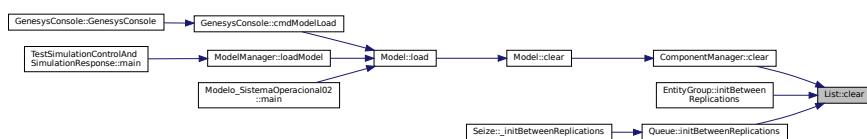
8.58.4.1 clear() template<typename T >
`void List< T >::clear`

Definition at line 142 of file [List.h](#).

```
00142         {
00143     _list->clear();
00144 }
```

Referenced by [ComponentManager::clear\(\)](#), [EntityGroup::initBetweenReplications\(\)](#), and [Queue::initBetweenReplications\(\)](#).

Here is the caller graph for this function:



8.58.4.2 create() [1/2] template<typename T >
`T List< T >::create`

Definition at line 137 of file [List.h](#).

```
00137         {
00138     return new T();
00139 }
```

8.58.4.3 `create()` [2/2] `template<typename T >`
`template<typename U >`
`T List< T >::create (`
`U arg)`

Definition at line 244 of file [List.h](#).

```
00244     {
00245         return T(arg);
00246     }
```

8.58.4.4 `current()` `template<typename T >`
`T List< T >::current`

Definition at line 232 of file [List.h](#).

```
00232     {
00233     /* \todo: To implement (i thing it's just to check). Must actualize _it on other methods when
00234     other elements are accessed */
00234     return (*_it);
00235 }
```

Referenced by [Entity::show\(\)](#).

Here is the caller graph for this function:



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

Definition at line 115 of file [List.h](#).

```
00115     {
00116     return _list->empty();
00117 }
```

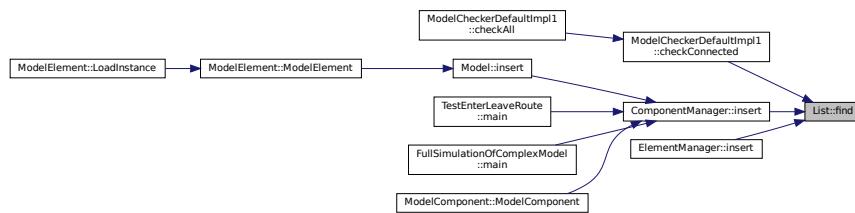
```
8.58.4.6 find() template<typename T >
std::list< T >::iterator List< T >::find (
    T element )
```

Definition at line 183 of file [List.h](#).

```
00183     {
00184         for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00185             if ((*it) == element) {
00186                 return it;
00187             }
00188         }
00189         return _list->end(); /* \todo:+: check nullptr or invalid iterator when not found */
00190         //return nullptr;
00191     }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ComponentManager::insert\(\)](#), and [ElementManager::insert\(\)](#).

Here is the caller graph for this function:



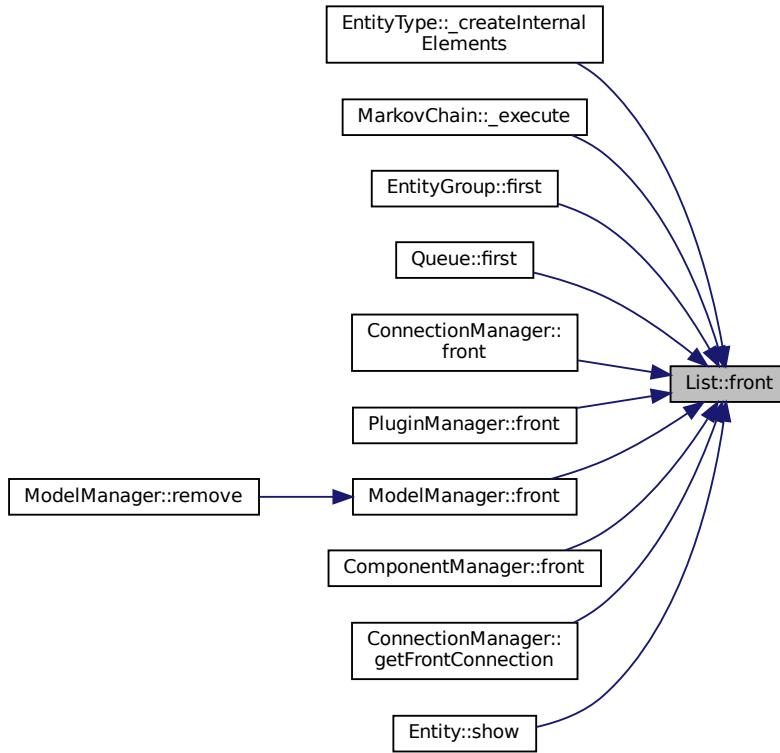
```
8.58.4.7 front() template<typename T >
T List< T >::front
```

Definition at line 208 of file [List.h](#).

```
00208     {
00209         _it = _list->begin();
00210         //if (_it != _list->end())
00211         return (*_it);
00212         //else
00213         //return dynamic_cast<T>(nullptr);
00214     }
```

Referenced by [EntityType::_createInternalElements\(\)](#), [MarkovChain::_execute\(\)](#), [EntityGroup::first\(\)](#), [Queue::first\(\)](#), [ConnectionManager::front\(\)](#), [ModelManager::front\(\)](#), [PluginManager::front\(\)](#), [ComponentManager::front\(\)](#), [ConnectionManager::getFrontConnection\(\)](#), and [Entity::show\(\)](#).

Here is the caller graph for this function:



8.58.4.8 `getAtRank()` template<typename T >

```
T List< T >::getAtRank (
    unsigned int rank )
```

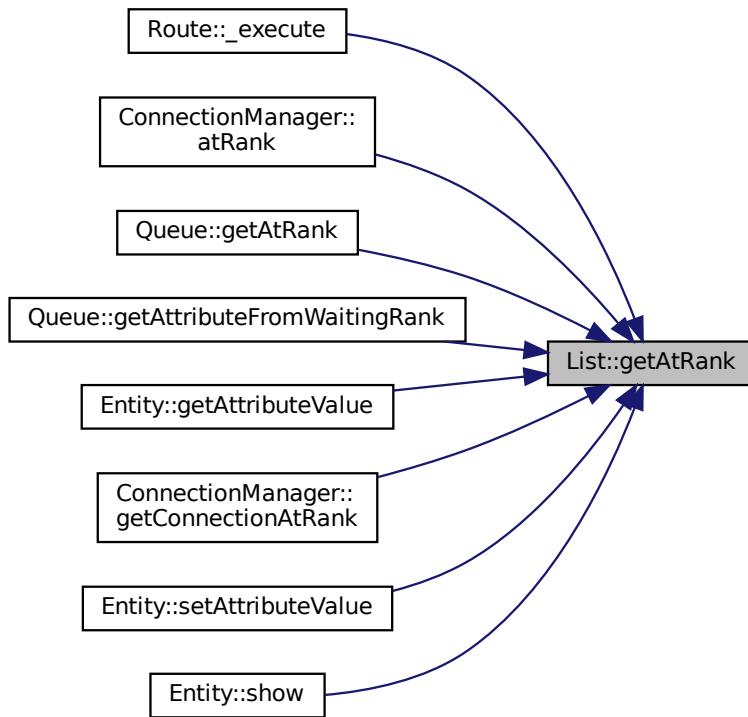
Definition at line 147 of file [List.h](#).

```

00147
00148     unsigned int thisRank = 0;
00149     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00150         if (rank == thisRank) {
00151             return (*it);
00152         } else {
00153             thisRank++;
00154         }
00155     }
00156     /* \todo: Invalid return depends on T. If T is pointer, nullptr works fine. If T is
        double, it does not. I just let (*it), buut it is not nice*/
00157 }
```

Referenced by [Route::_execute\(\)](#), [ConnectionManager::atRank\(\)](#), [Queue::getAtRank\(\)](#), [Queue::getAttributeFromWaitingRank\(\)](#), [Entity::getAttributeValue\(\)](#), [ConnectionManager::getConnectionAtRank\(\)](#), [Entity::setAttributeValue\(\)](#), and [Entity::show\(\)](#).

Here is the caller graph for this function:



8.58.4.9 insert() template<typename T >
`void List< T >::insert (`
 `T element)`

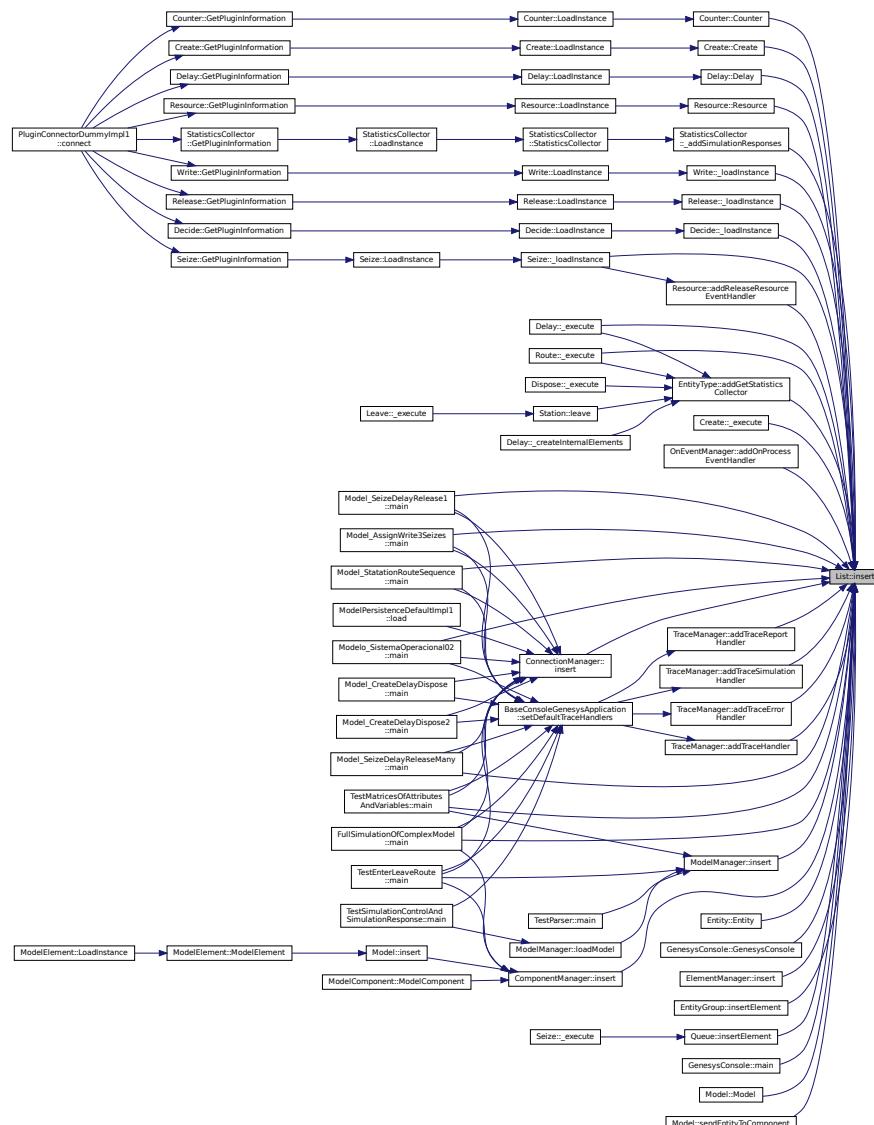
Definition at line 110 of file [List.h](#).

```

00110           {
00111     _list->insert(std::upper_bound(_list->begin(), _list->end(), element, _sortFunc), element);
00112 }
```

Referenced by [StatisticsCollector::_addSimulationResponses\(\)](#), [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Write::_loadInstance\(\)](#), [Release::_loadInstance\(\)](#), [Decide::_loadInstance\(\)](#), [Seize::_loadInstance\(\)](#), [EntityType::addGetStatisticsCollector\(\)](#), [OnEventManager::addOnProcessEventHandler\(\)](#), [Resource::addReleaseResourceEventHandler\(\)](#), [TraceManager::addTraceErrorHandler\(\)](#), [TraceManager::addTraceHandler\(\)](#), [TraceManager::addTraceReportHandler\(\)](#), [TraceManager::addTraceSimulationHandler\(\)](#), [Counter::Counter\(\)](#), [Create::Create\(\)](#), [Delay::Delay\(\)](#), [Entity::Entity\(\)](#), [GenesysConsole::GenesysConsole\(\)](#), [ComponentManager::insert\(\)](#), [ModelManager::insert\(\)](#), [ConnectionManager::insert\(\)](#), [ElementManager::insert\(\)](#), [EntityGroup::insertElement\(\)](#), [Queue::insertElement\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_AsignWrite3Seizes::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [GenesysConsole::main\(\)](#), [Model::Model\(\)](#), [Resource::Resource\(\)](#), and [Model::sendEntityToComponent\(\)](#).

Here is the caller graph for this function:



8.58.4.10 last() template<typename T >
T [List](#)< T >::last

Definition at line 217 of file [List.h](#).

```
00217      {
00218      _it = _list->end();
00219      _it--;
00220      //if (_it != _list->end()) // \todo: CHECK!!!
00221      return (*_it);
00222      //else return nullptr;
00223 }
```

Referenced by [PluginManager::last\(\)](#).

Here is the caller graph for this function:



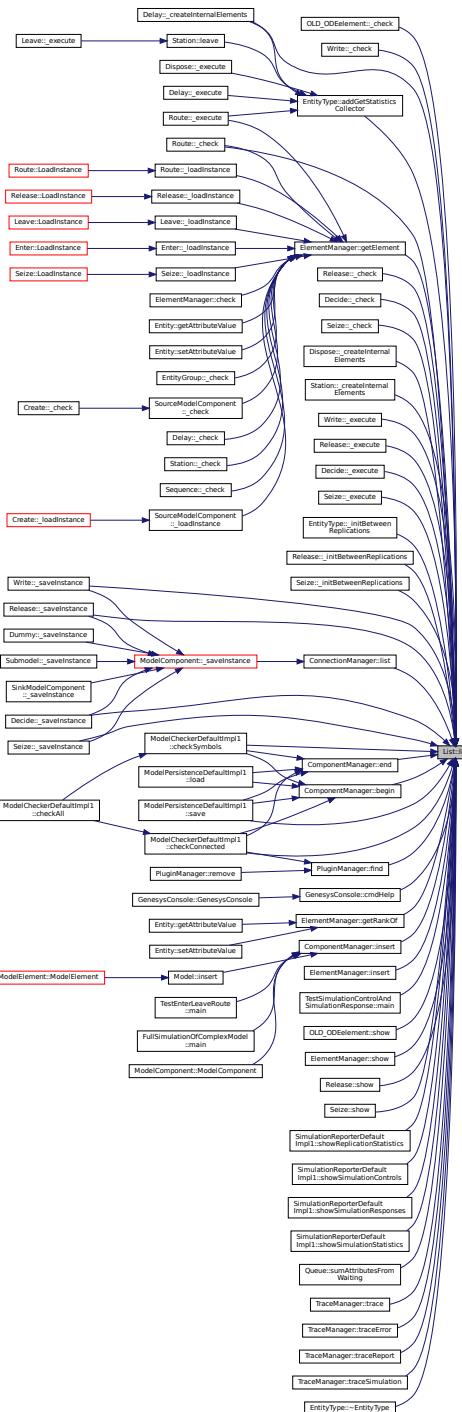
8.58.4.11 list() template<typename T >
std::list< T > * [List](#)< T >::list

Definition at line 81 of file [List.h](#).

```
00081     {  
00082         return _list;  
00083     }
```

Referenced by [OLD_ODElement::_check\(\)](#), [Write::_check\(\)](#), [Route::_check\(\)](#), [Release::_check\(\)](#), [Decide::_check\(\)](#), [Seize::_check\(\)](#), [Dispose::_createInternalElements\(\)](#), [Delay::_createInternalElements\(\)](#), [Station::_createInternalElements\(\)](#), [Write::_execute\(\)](#), [Release::_execute\(\)](#), [Decide::_execute\(\)](#), [Seize::_execute\(\)](#), [EntityType::_initBetweenReplications\(\)](#), [Release::_initBetweenReplications\(\)](#), [Seize::_initBetweenReplications\(\)](#), [Write::_saveInstance\(\)](#), [Release::_saveInstance\(\)](#), [Decide::_saveInstance\(\)](#), [Seize::_saveInstance\(\)](#), [EntityType::addGetStatisticsCollector\(\)](#), [ComponentManager::begin\(\)](#), [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [GenesysConsole::cmdHelp\(\)](#), [ComponentManager::end\(\)](#), [PluginManager::find\(\)](#), [ElementManager::getElement\(\)](#), [ElementManager::getRankOf\(\)](#), [ComponentManager::insert\(\)](#), [ElementManager::insert\(\)](#), [ConnectionManager::list\(\)](#), [TestSimulationControlAndSimulationResponse::ModelPersistenceDefaultImpl1::save\(\)](#), [OLD_ODElement::show\(\)](#), [ElementManager::show\(\)](#), [Release::show\(\)](#), [Seize::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [SimulationReporterDefaultImpl1::showSimulationControl\(\)](#), [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#), [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#), [Queue::sumAttributesFromWaiting\(\)](#), [TraceManager::trace\(\)](#), [TraceManager::traceError\(\)](#), [TraceManager::traceReport\(\)](#), [TraceManager::traceSimulation\(\)](#), and [EntityType::~EntityType\(\)](#).

Here is the caller graph for this function:



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

Definition at line 173 of file [List.h](#).
00173 {

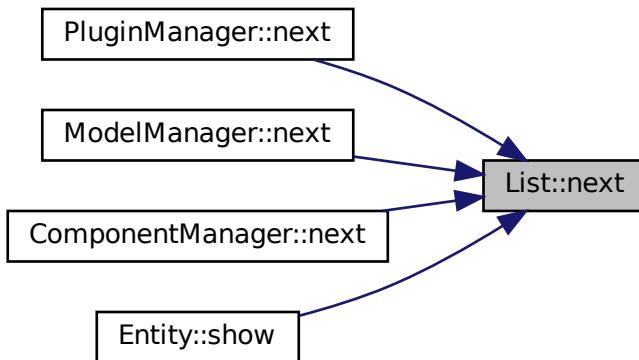
```

00174     _it++;
00175     if (_it != _list->end())
00176         return (*_it);
00177     else
00178         return nullptr;
00179
00180 }

```

Referenced by [PluginManager::next\(\)](#), [ModelManager::next\(\)](#), [ComponentManager::next\(\)](#), and [Entity::show\(\)](#).

Here is the caller graph for this function:



8.58.4.13 pop_front() template<typename T >

```
void List< T >::pop_front
```

Definition at line 120 of file [List.h](#).

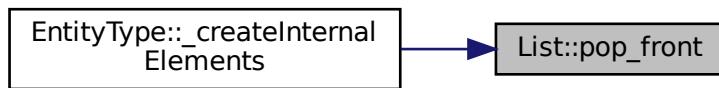
```

00120     {
00121     typename std::list<T>::iterator itTemp = _list->begin();
00122     _list->pop_front();
00123     if (_it == itTemp) { /* \todo: :: check this */
00124         _it = _list->begin(); // if it points to the removed element, then changes to begin
00125     }
00126 }

```

Referenced by [EntityType::_createInternalElements\(\)](#).

Here is the caller graph for this function:



8.58.4.14 previous() template<typename T >
T List< T >::previous

Definition at line 226 of file List.h.

```
00226     {
00227     _it--; // \todo: CHECK!!!
00228     return (*_it);
00229 }
```

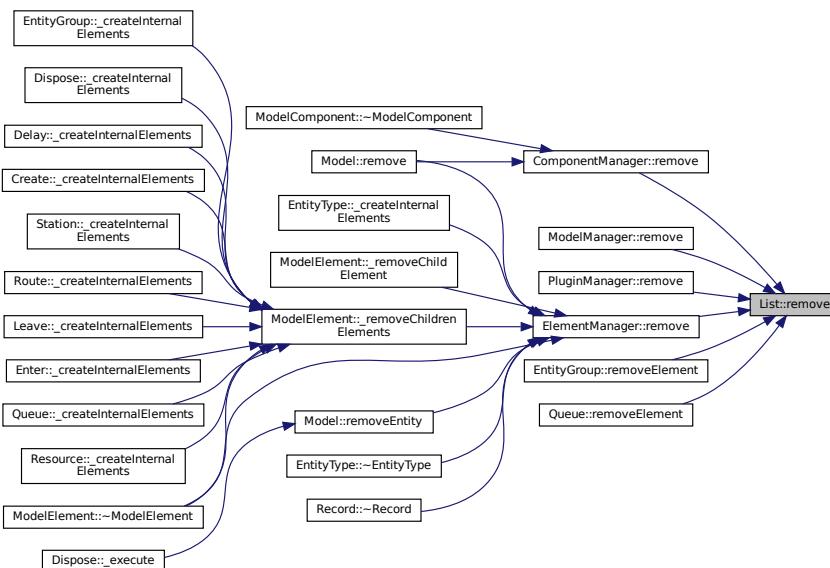
8.58.4.15 remove() template<typename T >
void List< T >::remove (
 T element)

Definition at line 129 of file List.h.

```
00129     _list->remove(element);
00130     if ((*_it) == element) { /* \todo: +: check this */
00131         _it = _list->begin(); // if it points to the removed element, then changes to begin
00132     }
00134 }
```

Referenced by [ComponentManager::remove\(\)](#), [ModelManager::remove\(\)](#), [PluginManager::remove\(\)](#), [ElementManager::remove\(\)](#), [EntityGroup::removeElement\(\)](#), and [Queue::removeElement\(\)](#).

Here is the caller graph for this function:



8.58.4.16 setAtRank() template<typename T >
void List< T >::setAtRank (
 unsigned int rank,
 T element)

Definition at line 160 of file [List.h](#).

```
00160                                         {  

00161     unsigned int thisRank = 0;  

00162     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {  

00163         if (rank == thisRank) {  

00164             *it = element;  

00165             return;  

00166         } else {  

00167             thisRank++;  

00168         }  

00169     }  

00170 }
```

Referenced by [ConnectionManager::insertAtRank\(\)](#).

Here is the caller graph for this function:



8.58.4.17 setSortFunc() template<typename T >
void List< T >::setSortFunc (
 CompFunct _sortFunc)

Definition at line 238 of file [List.h](#).

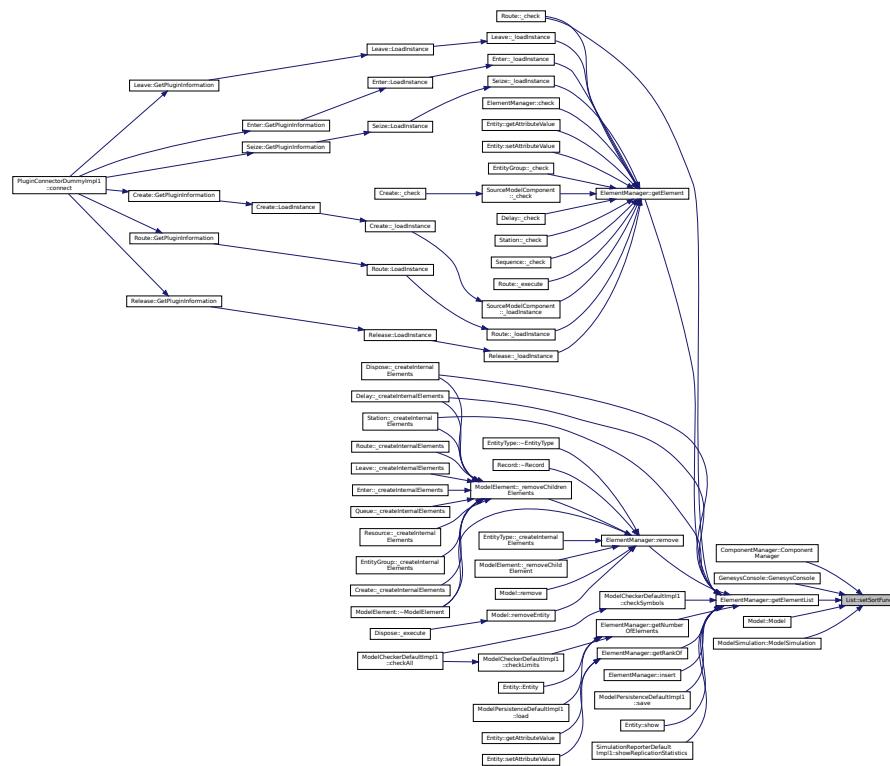
```
00238                                         {  

00239     this->_sortFunc = _sortFunc;  

00240 }
```

Referenced by [ComponentManager::ComponentManager\(\)](#), [GenesysConsole::GenesysConsole\(\)](#), [ElementManager::getElementList\(\)](#), [Model::Model\(\)](#), and [ModelSimulation::ModelSimulation\(\)](#).

Here is the caller graph for this function:



8.58.4.18 show() template<typename T >
 std::string List< T >::show

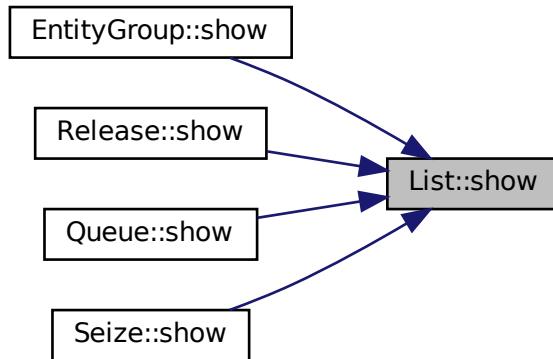
Definition at line 99 of file [List.h](#).

```

00099     {
00100     int i = 0;
00101     std::string text = "{";
00102     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++, i++) {
00103         text += "[" + std::to_string(i) + "]=" + (*it)->show() + ",";
00104     }
00105     text += "}";
00106     return text;
00107 }
```

Referenced by [EntityGroup::show\(\)](#), [Release::show\(\)](#), [Queue::show\(\)](#), and [Seize::show\(\)](#).

Here is the caller graph for this function:



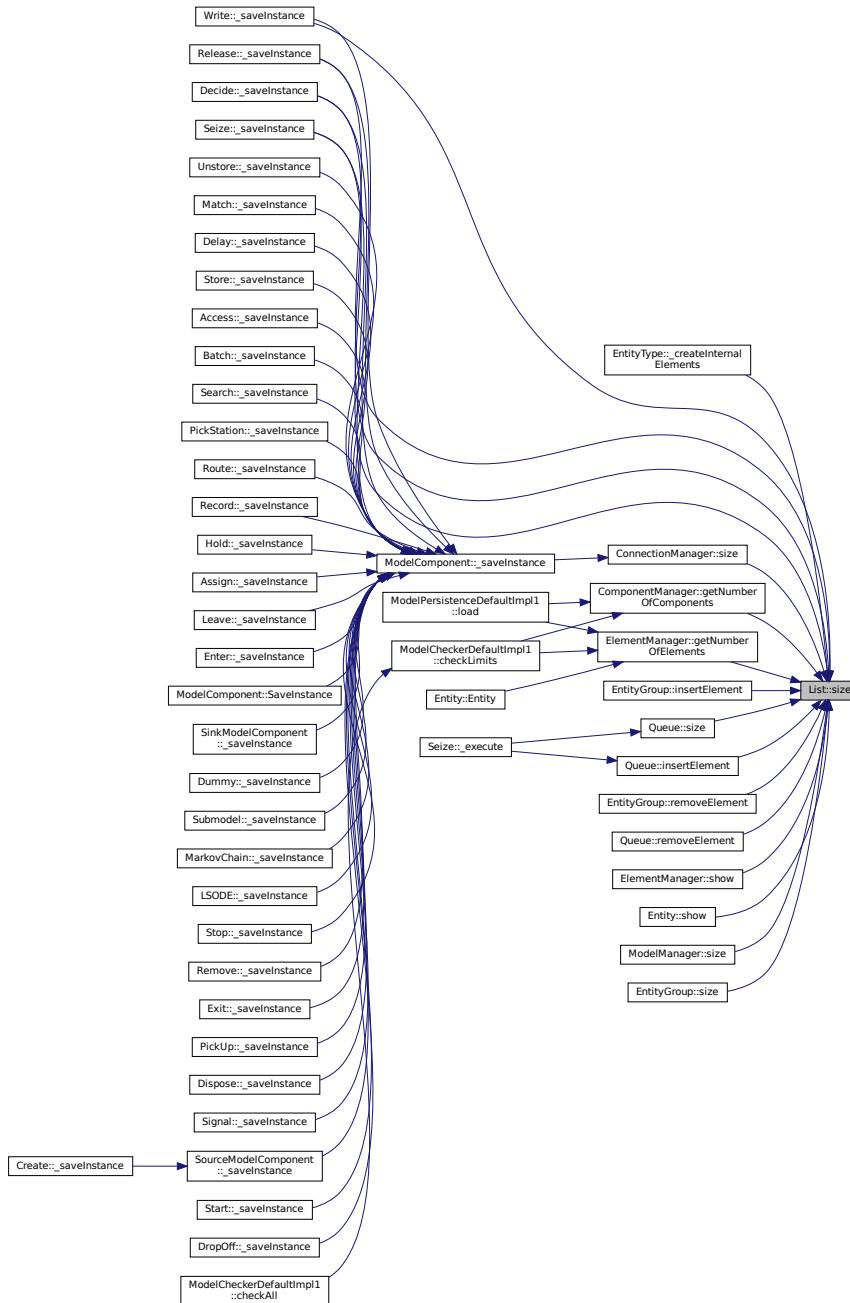
8.58.4.19 size() template<typename T >
unsigned int List< T >::size

Definition at line 86 of file [List.h](#).

```
00086     {  
00087         return _list->size();  
00088     }
```

Referenced by [EntityType::_createInternalElements\(\)](#), [Write::_saveInstance\(\)](#), [Release::_saveInstance\(\)](#), [Decide::_saveInstance\(\)](#), [Seize::_saveInstance\(\)](#), [ComponentManager::getNumberOfComponents\(\)](#), [ElementManager::getNumberOfComponents\(\)](#), [EntityGroup::insertElement\(\)](#), [Queue::insertElement\(\)](#), [EntityGroup::removeElement\(\)](#), [Queue::removeElement\(\)](#), [ElementManager::show\(\)](#), [Entity::show\(\)](#), [ConnectionManager::size\(\)](#), [ModelManager::size\(\)](#), [EntityGroup::size\(\)](#), and [Queue::size\(\)](#).

Here is the caller graph for this function:



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

Definition at line 250 of file [List.h](#).
00250 {

```
00251     _list->sort (comp);
00252 }
```

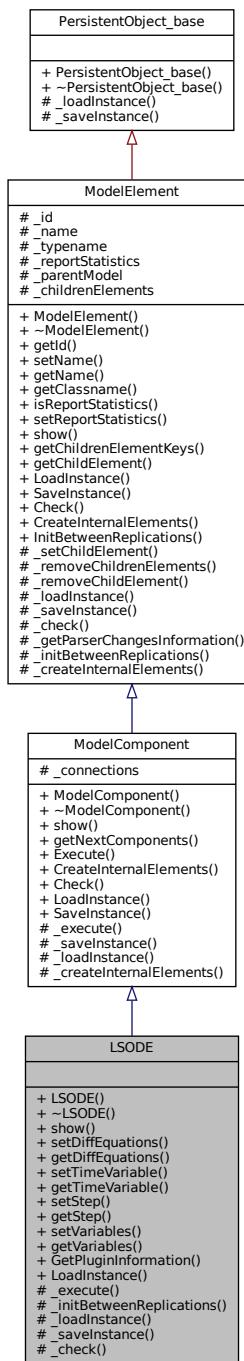
The documentation for this class was generated from the following file:

- [List.h](#)

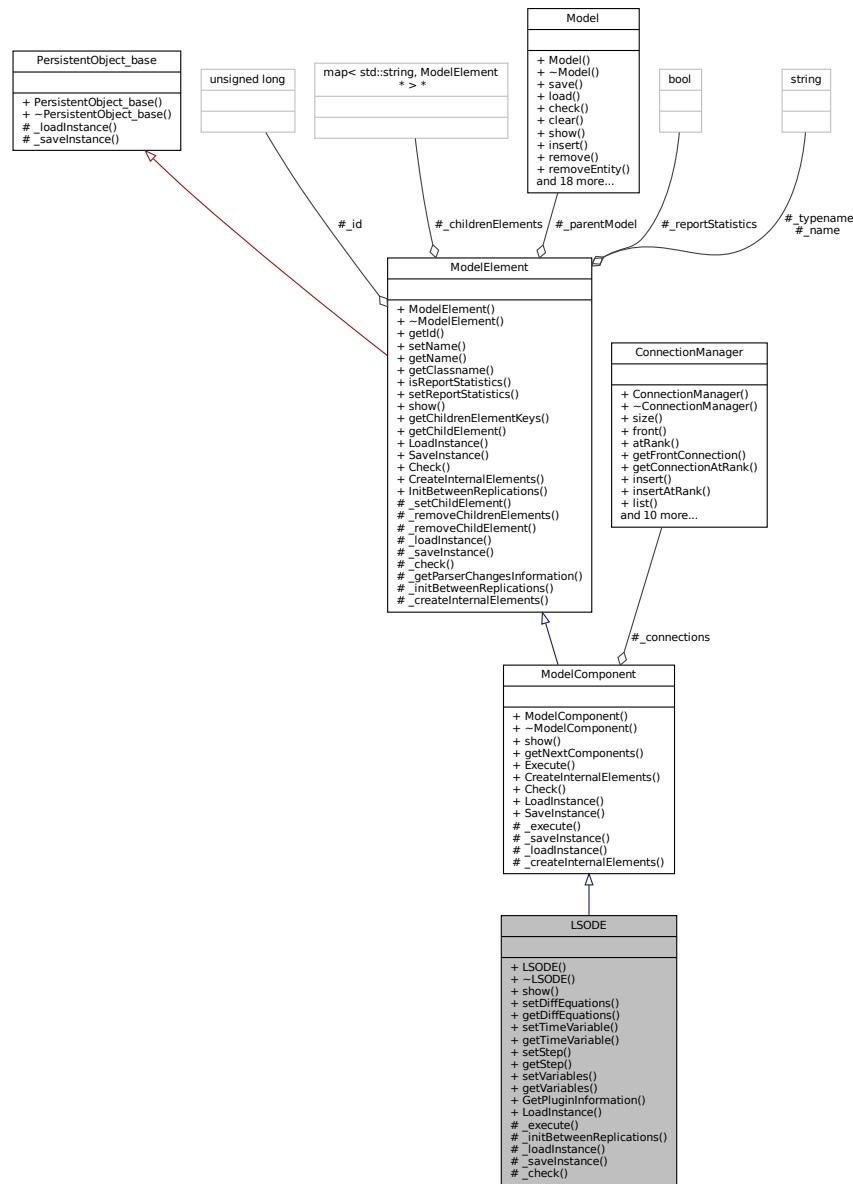
8.59 LSODE Class Reference

```
#include <LSODE.h>
```

Inheritance diagram for LSODE:



Collaboration diagram for LSODE:



Public Member Functions

- `LSODE (Model *model, std::string name="")`
- virtual `~LSODE ()=default`
- virtual `std::string show ()`
- `void setDiffEquations (Formula *formula)`
- `Formula * getDiffEquations () const`
- `void setTimeVariable (Variable *_timeVariable)`
- `Variable * getTimeVariable () const`
- `void setStep (double _step)`
- `double getStep () const`
- `void setVariables (Variable *_variables)`
- `Variable * getVariables () const`

Static Public Member Functions

- static [PluginInformation * GetPluginInformation \(\)](#)
- static [ModelComponent * LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

8.59.1 Detailed Description

This component ...

Definition at line [24](#) of file [LSODE.h](#).

8.59.2 Constructor & Destructor Documentation

```
8.59.2.1 LSODE() LSODE::LSODE (
    Model * model,
    std::string name = "")
```

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

```
00017 : ModelComponent(model, Util::TypeOf<LSODE>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.59.2.2 ~LSODE() virtual LSODE::~LSODE () [virtual], [default]

8.59.3 Member Function Documentation

8.59.3.1 `_check()` `bool LSODE::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 149 of file [LSODE.cpp](#).

```
00149
00150     bool resultAll = true;
00151     //...
00152     return resultAll;
00153 }
```

8.59.3.2 `_execute()` `void LSODE::_execute (Entity * entity) [protected], [virtual]`

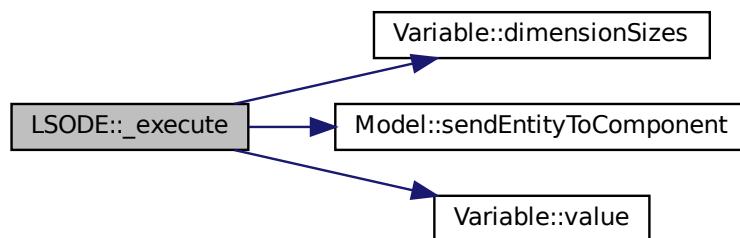
Implements [ModelComponent](#).

Definition at line 116 of file [LSODE.cpp](#).

```
00116
00117     //_parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00118     //for (std::list<std::string>::iterator it =
00119     //_diffEquations->getFormulaExpressions()->list()->begin(); it != _diffEquations->getFormulaExpressions()->list()->end(); it++) {
00119     //double value = _parentModel->parseExpression((*it));
00120     //_parentModel->tracer()->trace("Expression \"" + (*it) + "\" evaluates to " +
00121     //std::to_string(value));
00121     //}
00122     while (_doStep()) { // execute solve ODE step by step until reach TNOW
00123         std::string message = "time=" + std::to_string(_timeVariable->value());
00124         for (unsigned int i = 0; i < _variables->dimensionSizes()->front(); i++) {
00125             message += " ,y[" + std::to_string(i) + "]=" +
00126             std::to_string(_variables->value(std::to_string(i)));
00127         }
00128         _parentModel->tracer()->trace(message);
00129     }
00129     _parentModel->sendEntityToComponent(entity, nextComponents()->frontConnection(), 0.0);
00130 }
```

References [ModelElement::_parentModel](#), [Variable::dimensionSizes\(\)](#), [Model::sendEntityToComponent\(\)](#), and [Variable::value\(\)](#).

Here is the call graph for this function:



8.59.3.3 `_initBetweenReplications()` void LSODE::_initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 140 of file [LSODE.cpp](#).

```
00140
00141 }
```

8.59.3.4 `_loadInstance()` bool LSODE::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelComponent](#).

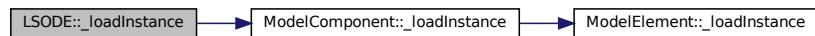
Definition at line 132 of file [LSODE.cpp](#).

```
00132
00133     bool res = ModelComponent::_loadInstance(fields);
00134     if (res) {
00135         //...
00136     }
00137     return res;
00138 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.59.3.5 `_saveInstance()` `std::map< std::string, std::string > * LSODE::_saveInstance () [protected], [virtual]`

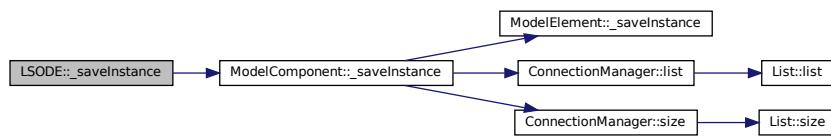
Reimplemented from [ModelComponent](#).

Definition at line 143 of file [LSODE.cpp](#).

```
00143 {  
00144     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);  
00145     //...  
00146     return fields;  
00147 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.59.3.6 `getDiffEquations()` `Formula * LSODE::getDiffEquations () const`

Definition at line 38 of file [LSODE.cpp](#).

```
00038 {  
00039     return _diffEquations;  
00040 }
```

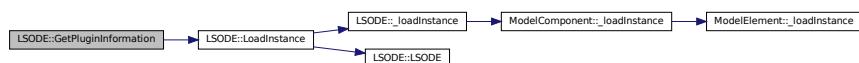
8.59.3.7 `GetPluginInformation()` `PluginInformation * LSODE::GetPluginInformation () [static]`

Definition at line 155 of file [LSODE.cpp](#).

```
00155 {  
00156     PluginInformation* info = new PluginInformation(Util::TypeOf<LSODE>(), &LSODE::LoadInstance);  
00157     // ...  
00158     return info;  
00159 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.59.3.8 getStep() double LSODE::getStep () constDefinition at line 54 of file [LSODE.cpp](#).

```
00054     {
00055     return _step;
00056 }
```

8.59.3.9 getTimeVariable() Variable * LSODE::getTimeVariable () constDefinition at line 46 of file [LSODE.cpp](#).

```
00046     {
00047     return _timeVariable;
00048 }
```

8.59.3.10 getVariables() Variable * LSODE::getVariables () constDefinition at line 62 of file [LSODE.cpp](#).

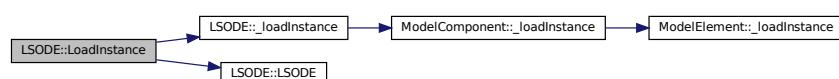
```
00062     {
00063     return _variables;
00064 }
```

8.59.3.11 LoadInstance() ModelComponent * LSODE::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]Definition at line 24 of file [LSODE.cpp](#).

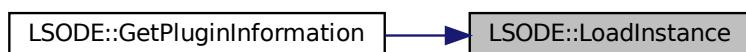
```
00024     {
00025     LSODE* newComponent = new LSODE(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00031     return newComponent;
00032 }
```

References [_loadInstance\(\)](#), and [LSODE\(\)](#).Referenced by [GetPluginInformation\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.59.3.12 setDiffEquations() void LSODE::setDiffEquations (
 Formula * formula)

Definition at line 34 of file [LSODE.cpp](#).

```
00034 {  
00035     _diffEquations = formula;  
00036 }
```

8.59.3.13 setStep() void LSODE::setStep (
 double _step)

Definition at line 50 of file [LSODE.cpp](#).

```
00050 {  
00051     _step = step;  
00052 }
```

8.59.3.14 setTimeVariable() void LSODE::setTimeVariable (
 Variable * _timeVariable)

Definition at line 42 of file [LSODE.cpp](#).

```
00042 {  
00043     _timeVariable = timeVariable;  
00044 }
```

8.59.3.15 setVariables() void LSODE::setVariables (
 Variable * _variables)

Definition at line 58 of file [LSODE.cpp](#).

```
00058 {  
00059     _variables = variables;  
00060 }
```

8.59.3.16 show() std::string LSODE::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [LSODE.cpp](#).

```
00020 {  
00021     return ModelComponent::show() + "";  
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



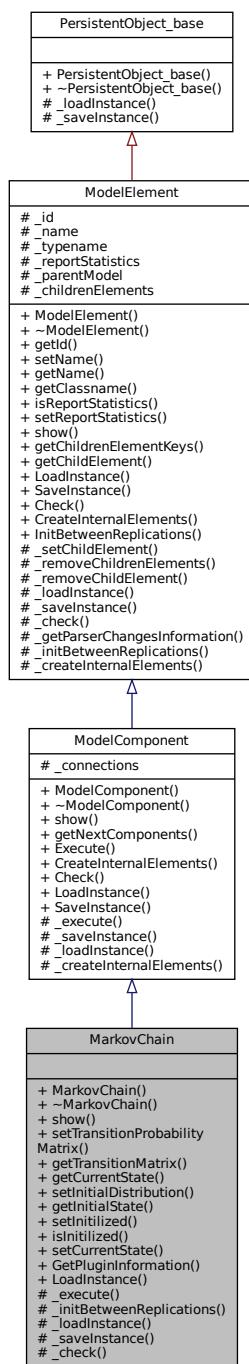
The documentation for this class was generated from the following files:

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

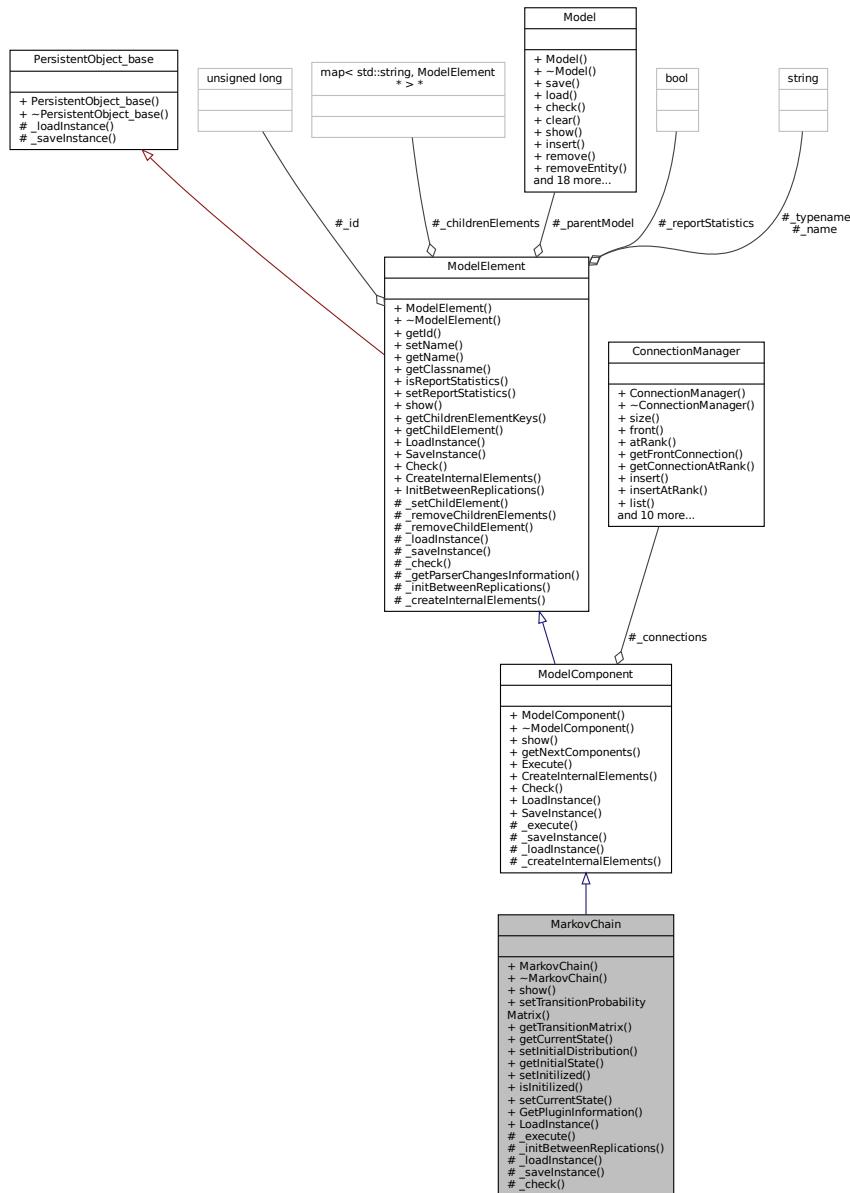
8.60 MarkovChain Class Reference

```
#include <MarkovChain.h>
```

Inheritance diagram for MarkovChain:



Collaboration diagram for MarkovChain:



Public Member Functions

- **MarkovChain (Model *model, std::string name="")**
- virtual **~MarkovChain ()=default**
- virtual std::string **show ()**
- void **setTransitionProbabilityMatrix (Variable *_transitionMatrix)**
- **Variable * getTransitionMatrix () const**
- **Variable * getCurrentState () const**
- void **setInitialDistribution (Variable *_initialDistribution)**
- **Variable * getInitialState () const**
- void **setInitialized (bool _initialized)**
- bool **isInitialized () const**
- void **setCurrentState (Variable *_currentState)**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.60.1 Detailed Description

Definition at line 21 of file [MarkovChain.h](#).

8.60.2 Constructor & Destructor Documentation

```
8.60.2.1 MarkovChain() MarkovChain::MarkovChain (
    Model * model,
    std::string name = "")
```

Definition at line 22 of file [MarkovChain.cpp](#).

```
00022     Util::TypeOf<MarkovChain>(), name) {
00023         : ModelComponent(model,
00024             _sampler = new Traits<Sampler_if>::Implementation());
00024 }
```

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

Here is the caller graph for this function:



8.60.2.2 ~MarkovChain() virtual MarkovChain::~MarkovChain () [virtual], [default]

8.60.3 Member Function Documentation

8.60.3.1 `_check()` `bool MarkovChain::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 127 of file [MarkovChain.cpp](#).

```
00127     {
00128     bool resultAll = true;
00129     //...
00130     return resultAll;
00131 }
```

8.60.3.2 `_execute()` `void MarkovChain::_execute (Entity * entity) [protected], [virtual]`

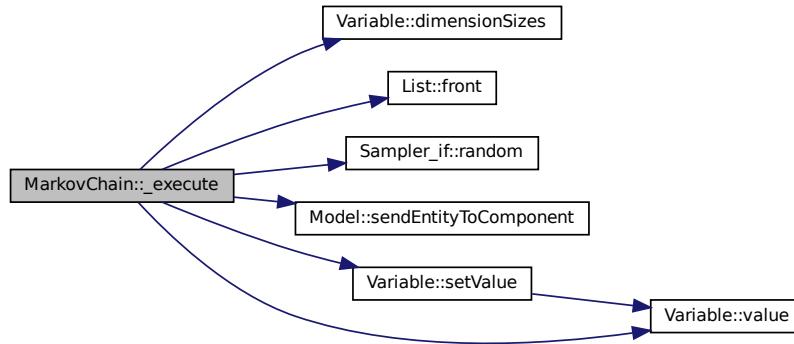
Implements [ModelComponent](#).

Definition at line 72 of file [MarkovChain.cpp](#).

```
00072     {
00073     //_parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00074     unsigned int size;
00075     double rnd, sum, value;
00076     if (!initialized) {
00077         // define the initial state based on initial probabilities
00078         size = _initialDistribution->dimensionSizes()->front();
00079         rnd = _sampler->random(); //parentSimulator()->tools()->sampler()->random();
00080         double sum = 0.0;
00081         for (unsigned int i = 0; i < size; i++) {
00082             value = _initialDistribution->value(std::to_string(i));
00083             sum += value;
00084             if (sum > rnd) {
00085                 _currentState->setValue(i); // _currentState = i;
00086                 break;
00087             }
00088         }
00089         _parentModel->tracer()->trace("Initial current state=" +
00090             std::to_string(_currentState->value()));
00091         initialized = true;
00092     } else {
00093         size = _transitionProbMatrix->dimensionSizes()->front();
00094         rnd = _sampler->random(); //parentSimulator()->tools()->sampler()->random();
00095         sum = 0.0;
00096         for (unsigned int i = 0; i < size; i++) {
00097             std::string index = std::to_string(static_cast<unsigned int> (_currentState->value()) +
00098                 "," + std::to_string(i));
00099             value = _transitionProbMatrix->value(index);
00100             sum += value;
00101             if (sum > rnd) {
00102                 _currentState->setValue(i);
00103                 break;
00104             }
00105         }
00106         _parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00107     }
```

References [ModelElement::_parentModel](#), [Variable::dimensionSizes\(\)](#), [List< T >::front\(\)](#), [Sampler_if::random\(\)](#), [Model::sendEntityToComponent\(\)](#), [Variable::setValue\(\)](#), and [Variable::value\(\)](#).

Here is the call graph for this function:



8.60.3.3 `_initBetweenReplications()` void `MarkovChain::_initBetweenReplications ()` [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 117 of file [MarkovChain.cpp](#).

```

00117
00118     this->_initialized = false;
00119 }
```

8.60.3.4 `_loadInstance()` bool `MarkovChain::_loadInstance (std::map< std::string, std::string > * fields)` [protected], [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 109 of file [MarkovChain.cpp](#).

```

00109
00110     bool res = ModelComponent::\_loadInstance(fields);
00111     if (res) {
00112         //...
00113     }
00114     return res;
00115 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.60.3.5 `_saveInstance()` `std::map< std::string, std::string > * MarkovChain::_saveInstance ()`
 [protected], [virtual]

Reimplemented from [ModelComponent](#).

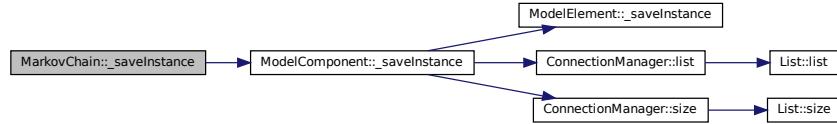
Definition at line 121 of file [MarkovChain.cpp](#).

```

00121
00122     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00123     //...
00124     return fields;
00125 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.60.3.6 `getCurrentState()` `Variable * MarkovChain::getCurrentState () const`

Definition at line 48 of file [MarkovChain.cpp](#).

```

00048
00049     return _currentState;
00050 }
```

8.60.3.7 `getInitialState()` `Variable * MarkovChain::getInitialState () const`

Definition at line 60 of file [MarkovChain.cpp](#).

```

00060
00061     return _initialDistribution;
00062 }
```

8.60.3.8 GetPluginInformation() `PluginInformation * MarkovChain::GetPluginInformation () [static]`

Definition at line 133 of file [MarkovChain.cpp](#).

```
00133                                         {
00134     PluginInformation* info = new PluginInformation(Util::TypeOf<MarkovChain>(),
00135     &MarkovChain::LoadInstance);
00136     // ...
00136     return info;
00137 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:

**8.60.3.9 getTransitionMatrix()** `Variable * MarkovChain::getTransitionMatrix () const`

Definition at line 44 of file [MarkovChain.cpp](#).

```
00044                                         {
00045     return _transitionProbMatrix;
00046 }
```

8.60.3.10 isInitialized() `bool MarkovChain::isInitialized () const`

Definition at line 68 of file [MarkovChain.cpp](#).

```
00068                                         {
00069     return _initialized;
00070 }
```

8.60.3.11 LoadInstance() `ModelComponent * MarkovChain::LoadInstance (`

```
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

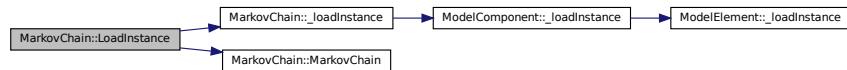
Definition at line 30 of file [MarkovChain.cpp](#).

```
00030                                         {
00031     MarkovChain* newComponent = new MarkovChain(model);
00032     try {
00033         newComponent->_loadInstance(fields);
00034     } catch (const std::exception& e) {
00035     }
00036     return newComponent;
00037 }
00038 }
```

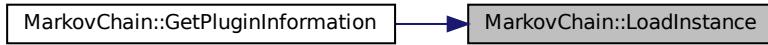
References [_loadInstance\(\)](#), and [MarkovChain\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.60.3.12 `setCurrentState()` `void MarkovChain::setCurrentState (Variable * _currentState)`

Definition at line 52 of file [MarkovChain.cpp](#).

```
00052
00053     this->_currentState = _currentState;
00054 }
```

8.60.3.13 `setInitialDistribution()` `void MarkovChain::setInitialDistribution (Variable * _initialDistribution)`

Definition at line 56 of file [MarkovChain.cpp](#).

```
00056
00057     this->_initialDistribution = _initialDistribution;
00058 }
```

8.60.3.14 `setInitialized()` `void MarkovChain::setInitialized (bool _initialized)`

Definition at line 64 of file [MarkovChain.cpp](#).

```
00064
00065     this->_initialized = _initialized;
00066 }
```

```
8.60.3.15 setTransitionProbabilityMatrix() void MarkovChain::setTransitionProbabilityMatrix (
    Variable * _transitionMatrix )
```

Definition at line 40 of file [MarkovChain.cpp](#).

```
00040
00041     this->_transitionProbMatrix = _transitionMatrix;
00042 }
```

```
8.60.3.16 show() std::string MarkovChain::show () [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 26 of file [MarkovChain.cpp](#).

```
00026
00027     {
00028         return ModelComponent::show() + "";
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



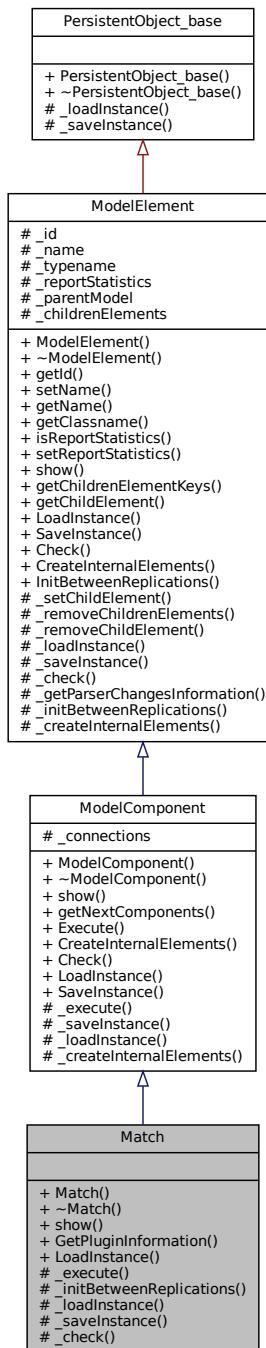
The documentation for this class was generated from the following files:

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

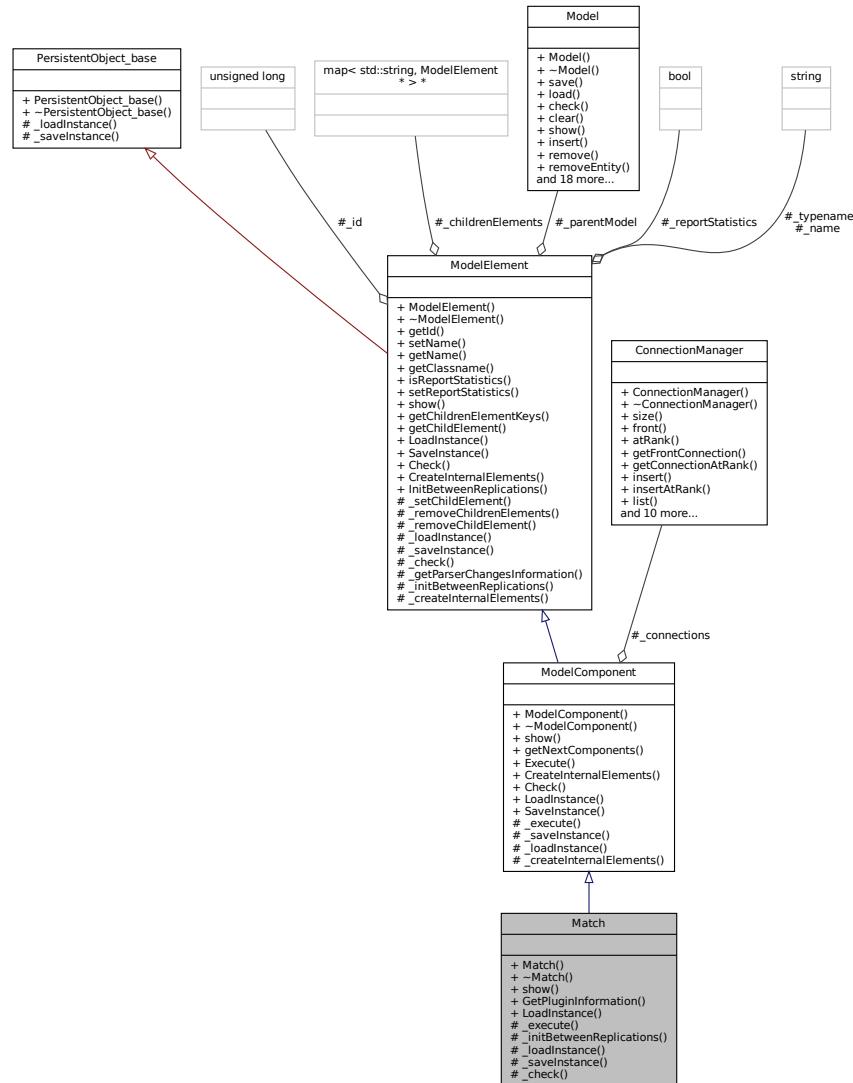
8.61 Match Class Reference

```
#include <Match.h>
```

Inheritance diagram for Match:



Collaboration diagram for Match:



Public Member Functions

- **Match** (`Model` *model, `std::string` name="")
- virtual **~Match** ()=default
- virtual `std::string` **show** ()

Static Public Member Functions

- static `PluginInformation` * **GetPluginInformation** ()
- static `ModelComponent` * **LoadInstance** (`Model` *model, `std::map< std::string, std::string >` *fields)

Protected Member Functions

- virtual void **_execute** (`Entity` *entity)
- virtual void **_initBetweenReplications** ()
- virtual bool **_loadInstance** (`std::map< std::string, std::string >` *fields)
- virtual `std::map< std::string, std::string >` * **_savelinstance** ()
- virtual bool **_check** (`std::string` *errorMessage)

Additional Inherited Members

8.61.1 Detailed Description

Match module DESCRIPTION The **Match** module brings together a specified number of entities waiting in different queues. The match may be accomplished when there is at least one entity in each of the desired queues. Additionally, an attribute may be specified such that the entities waiting in the queues must have the same attribute values before the match is initiated. When an entity arrives at the **Match** module, it is placed in one of up to five associated queues, based on the entry point to which it is connected. Entities will remain in their respective queues until a match exists. Once a match exists, one entity from each queue is released to be matched. The matched entities are then synchronized to depart from the module. TYPICAL USES Assembling a part Gathering various products for a customer order Synchronizing a customer exit with a filled order Prompt Description Name Unique module identifier displayed on the module shape. Number to **Match** Number of matching entities that must reside in different queues before a match may be completed. Type Method for matching the incoming entities. If Type is Any Entities, one entity must reside in each queue for a match to be made. If Type is Based on **Attribute**, one entity must reside in each queue with the same attribute value. **Attribute** Name **Attribute** name that is used for identifying an arriving entity's match value. Applies only when Type is Based on **Attribute**.

Definition at line 47 of file [Match.h](#).

8.61.2 Constructor & Destructor Documentation

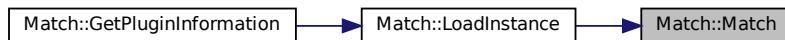
8.61.2.1 Match() `Match::Match (`
 `Model * model,`
 `std::string name = "")`

Definition at line 18 of file [Match.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Match>(), name) {  
00019 }
```

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

Here is the caller graph for this function:



8.61.2.2 ~Match() `virtual Match::~Match () [virtual], [default]`

8.61.3 Member Function Documentation

```
8.61.3.1 _check() bool Match::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Match.cpp](#).

```
00057
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
```

```
8.61.3.2 _execute() void Match::_execute (
    Entity * entity ) [protected], [virtual]
```

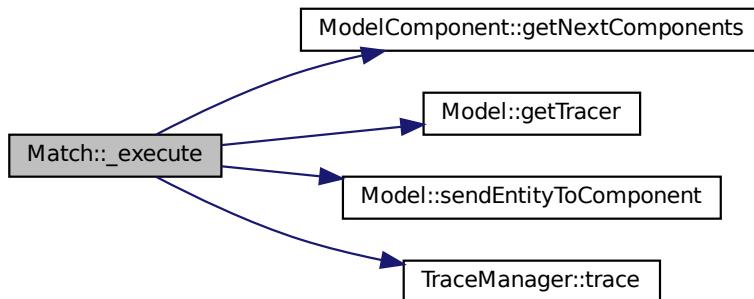
Implements [ModelComponent](#).

Definition at line 35 of file [Match.cpp](#).

```
00035
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
0.0);
00038 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent\(\)](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



```
8.61.3.3 _initBetweenReplications() void Match::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Match.cpp](#).

```
00048
00049 }
```

8.61.3.4 `_loadInstance()` `bool Match::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

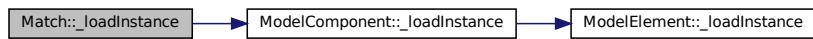
Definition at line 40 of file [Match.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

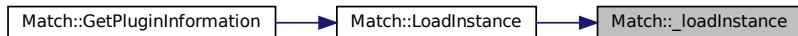
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.61.3.5 `_saveInstance()` `std::map< std::string, std::string > * Match::_saveInstance () [protected], [virtual]`

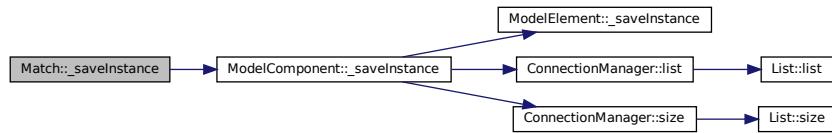
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Match.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



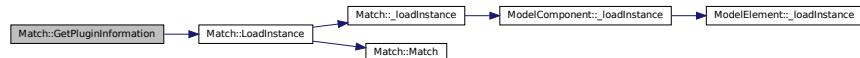
8.61.3.6 GetPluginInformation() `PluginInformation * Match::GetPluginInformation () [static]`

Definition at line 63 of file `Match.cpp`.

```
00063 {  
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Match>(), &Match::LoadInstance);  
00065     // ...  
00066     return info;  
00067 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.61.3.7 LoadInstance() `ModelComponent * Match::LoadInstance (`

```
Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file `Match.cpp`.

```
00025 {  
00026     Match* newComponent = new Match(model);  
00027     try {  
00028         newComponent->_loadInstance(fields);  
00029     } catch (const std::exception& e) {  
00030     }  
00032     return newComponent;  
00033 }
```

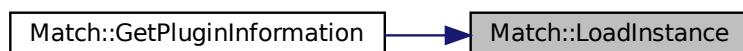
References `_loadInstance()`, and `Match()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.61.3.8 `show()` `std::string Match::show() [virtual]`

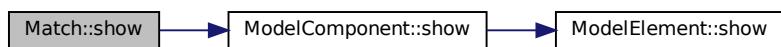
Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Match.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



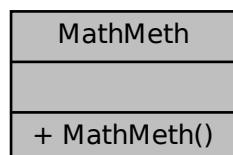
The documentation for this class was generated from the following files:

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

8.62 MathMeth Class Reference

```
#include <MathMeth.h>
```

Collaboration diagram for MathMeth:



Public Member Functions

- [MathMeth \(\)](#)

8.62.1 Detailed Description

Definition at line 19 of file [MathMeth.h](#).

8.62.2 Constructor & Destructor Documentation

8.62.2.1 MathMeth() MathMeth::MathMeth ()

Definition at line 16 of file [MathMeth.cpp](#).

```
00016 {  
00017 }
```

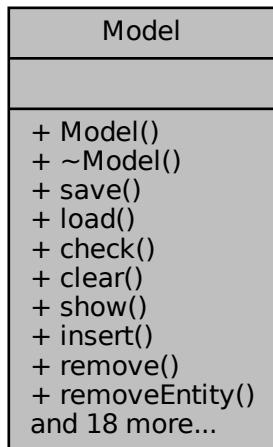
The documentation for this class was generated from the following files:

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

8.63 Model Class Reference

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

Collaboration diagram for Model:



Public Member Functions

- [Model \(Simulator *simulator\)](#)
- virtual [~Model \(\)=default](#)
- [bool save \(std::string filename\)](#)
- [bool load \(std::string filename\)](#)
- [bool check \(\)](#)

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

- void `clear ()`
- void `show ()`
- bool `insert (ModelElement *elemOrComp)`

Insert a new `ModelElement` or `ModelComponent` into the model (since 20191015). It's a generic access to `ComponentManager->insert()` or `ModelElement->insert()`
- void `remove (ModelElement *elemOrComp)`

Remove a new `ModelElement` or `ModelComponent` into the model (since 20191015). It's a generic access to `ComponentManager->remove()` or `ModelElement->remove()`
- void `removeEntity (Entity *entity, bool collectStatistics)`
- void `sendEntityToComponent (Entity *entity, Connection *connection, 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`).
- void `sendEntityToComponent (Entity *entity, ModelComponent *component, double timeDelay, unsigned int componentInputNumber=0)`

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)`
- bool `checkExpression (const std::string expression, const std::string expressionName, std::string *errorMessage)`
- `Util::identification getId () const`
- bool `hasChanged () const`
- `OnEventManager * getOnEvents () const`
- `ElementManager * getElements () const`

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).
- `ComponentManager * getComponents () 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`).
- `ModellInfo * getInfo () const`
- `Simulator * getParentSimulator () const`
- `ModelSimulation * getSimulation () const`

Provides access to the class that manages the model simulation.
- `List< Event * > * getFutureEvents () 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.
- void `setTracer (TraceManager *_traceManager)`
- `TraceManager * getTracer () const`

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

8.63.1 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 configurated and connected, and a collection of under layered element (`ModelElement`).

Definition at line 44 of file `Model.h`.

8.63.2 Constructor & Destructor Documentation

8.63.2.1 Model() Model::Model (

```
    Simulator * simulator )
```

The future events list must be chronologicaly sorted

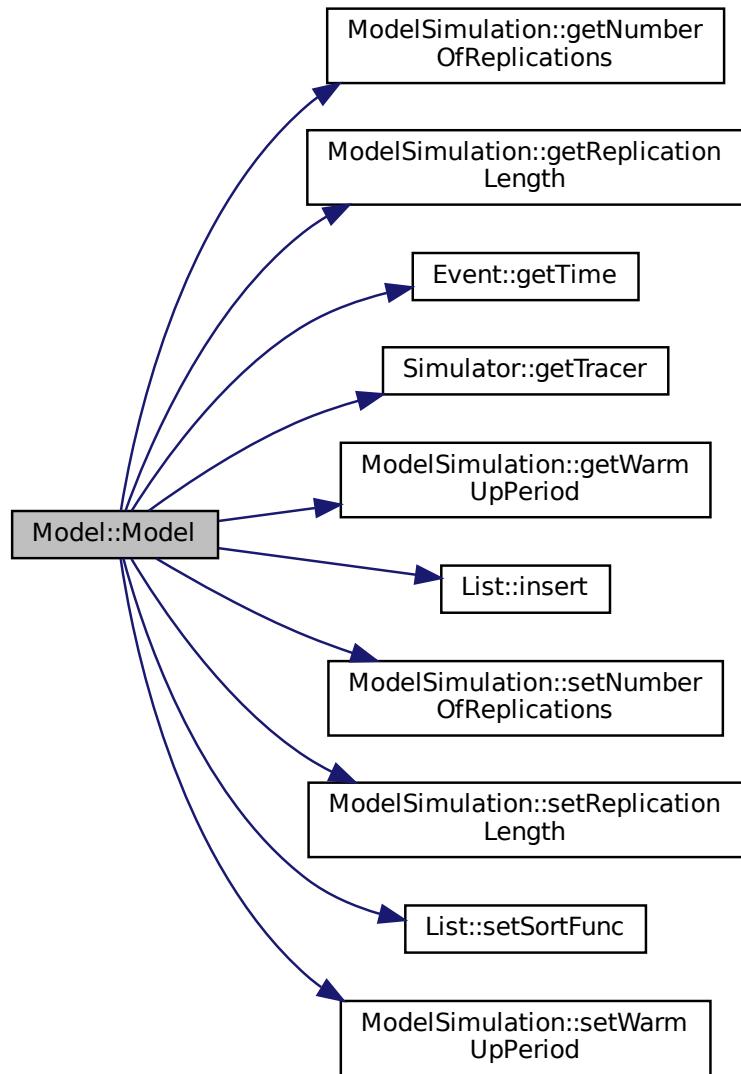
Events are sorted chronologically

Definition at line 32 of file [Model.cpp](#).

```
00032     {
00033     _parentSimulator = simulator; // a simulator is the "parent" of a model
00034     // 1:1 associations (no Traits)
00035     _traceManager = simulator->getTracer(); // every model starts with the same tracer, unless a
00036     // specific one is set
00037     _modelInfo = new ModelInfo();
00038     _eventManager = new OnEventManager(); // should be on .h (all that does not depends on THIS)
00039     _elementManager = new ElementManager(this);
00040     _componentManager = new ComponentManager(this);
00041     _simulation = new ModelSimulation(this);
00042     // 1:1 associations (Traits)
00043     //Sampler_if* sampler = new Traits<Sampler_if>::Implementation();
00044     _parser = new Traits<Parser_if>::Implementation(this, new Traits<Sampler_if>::Implementation());
00045     _modelChecker = new Traits<ModelChecker_if>::Implementation(this);
00046     _modelPersistence = new Traits<ModelPersistence_if>::Implementation(this);
00047     // 1:n associations
00048     _futureEvents = new List<Event*>();
00049     //_events->setSortFunc(&EventCompare); // It works too
00050     _futureEvents->setSortFunc([](const Event* a, const Event* b) {
00051         return a->getTime() < b->getTime();
00052     });
00053     // for process analyser
00054     _responses = new List<SimulationResponse*>();
00055     _controls = new List<SimulationControl*>();
00056     // insert controls
00057     _controls->insert(new SimulationControl("ModelSimulation", "NumberOfReplications",
00058         DefineGetterMember<ModelSimulation>(this->_simulation,
00059         &ModelSimulation::getNumberOfReplications),
00060         DefineSetterMember<ModelSimulation>(this->_simulation,
00061         &ModelSimulation::setNumberOfReplications))
00062     );
00063     _controls->insert(new SimulationControl("ModelSimulation", "ReplicationLength",
00064         DefineGetterMember<ModelSimulation>(this->_simulation,
00065         &ModelSimulation::getReplicationLength),
00066         DefineSetterMember<ModelSimulation>(this->_simulation,
00067         &ModelSimulation::setReplicationLength))
00068 );
```

References [ModelSimulation::getNumberOfReplications\(\)](#), [ModelSimulation::getReplicationLength\(\)](#), [Event::getTime\(\)](#), [Simulator::getTracer\(\)](#), [ModelSimulation::getWarmUpPeriod\(\)](#), [List< T >::insert\(\)](#), [ModelSimulation::setNumberOfReplications\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [List< T >::setSortFunc\(\)](#), and [ModelSimulation::setWarmUpPeriod\(\)](#).

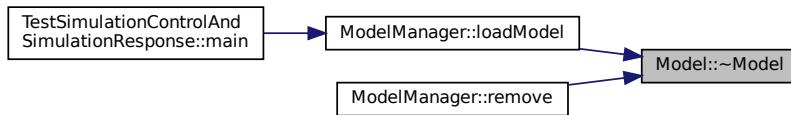
Here is the call graph for this function:



8.63.2.2 ~Model() `virtual Model::~Model () [virtual], [default]`

Referenced by [ModelManager::loadModel\(\)](#), and [ModelManager::remove\(\)](#).

Here is the caller graph for this function:



8.63.3 Member Function Documentation

8.63.3.1 `check()` `bool Model::check()`

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

Definition at line 272 of file [Model.cpp](#).

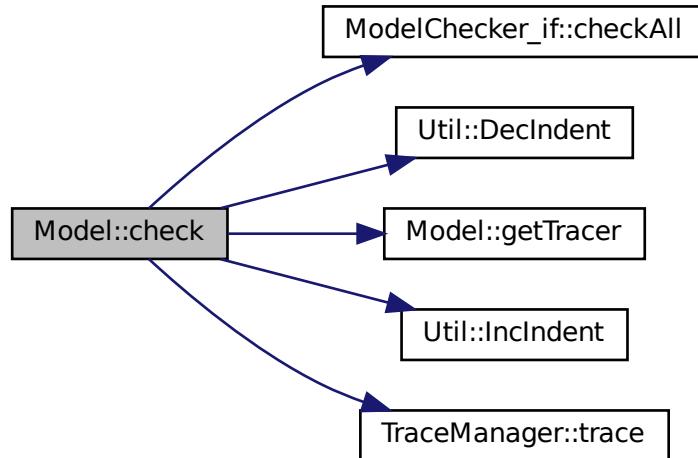
```

00272     {
00273         getTracer() -> trace(Util::TraceLevel::modelInternal, "Checking model consistency");
00274         Util::IncIndent();
00275         // before checking the model, creates all necessary internal ModelElements
00276         _createModelInternalElements();
00277         bool res = this->_modelChecker->checkAll();
00278         Util::DecIndent();
00279         if (res) {
00280             getTracer() -> trace(Util::TraceLevel::modelResult, "End of Model checking: Success");
00281         } else {
00282             //std::exception e = new std::exception();
00283             //getTrace()->traceError();
00284             getTracer() -> trace(Util::TraceLevel::modelResult, "End of Model checking: Failed");
00285         }
00286         return res;
00287     }
  
```

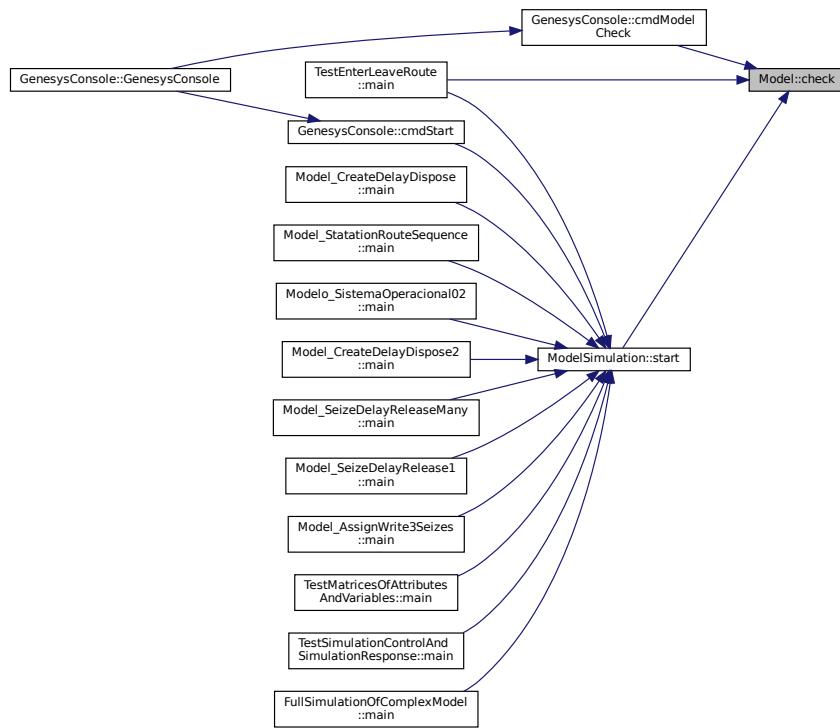
References [ModelChecker_if::checkAll\(\)](#), [Util::DecIndent\(\)](#), [getTracer\(\)](#), [Util::IncIndent\(\)](#), [Util::modelInternal](#), [Util::modelResult](#), and [TraceManager::trace\(\)](#).

Referenced by [GenesysConsole::cmdModelCheck\(\)](#), [TestEnterLeaveRoute::main\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.63.3.2 checkExpression() bool Model::checkExpression (
    const std::string expression,
    const std::string expressionName,
    std::string * errorMessage )
```

Definition at line 118 of file [Model.cpp](#).

```
00118
00119     {
00120         bool result;
00121         parseExpression(expression, &result, errorMessage);
00122         if (!result) {
00123             std::string msg = "Expression \"\" + expression + "\" for '" + expressionName + "' is
incorrect. ";
00124             errorMessage->append(msg);
00125         }
00126     }
00127 }
```

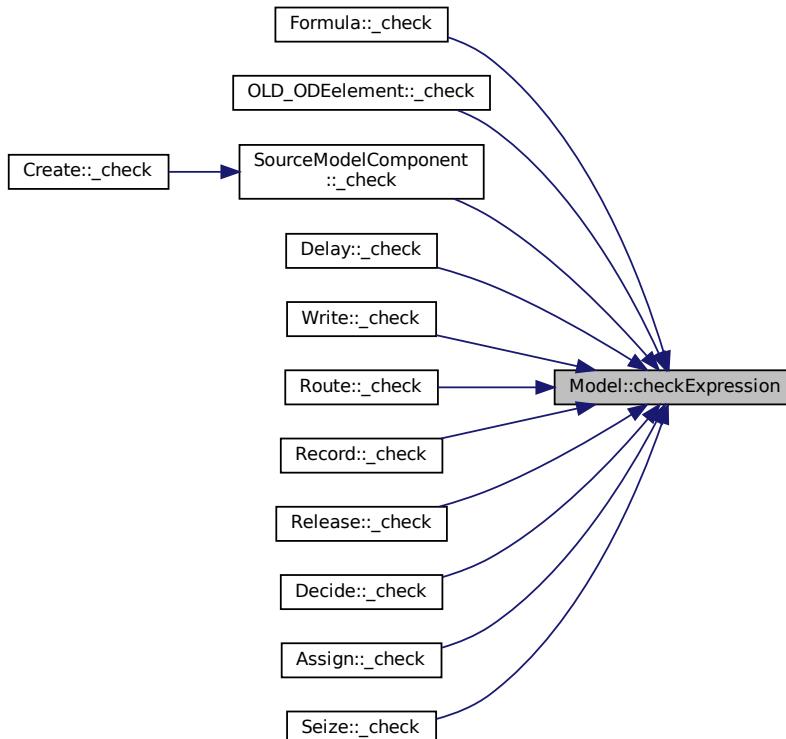
References [parseExpression\(\)](#).

Referenced by [Formula::_check\(\)](#), [OLD_ODElement::_check\(\)](#), [SourceModelComponent::_check\(\)](#), [Delay::_check\(\)](#), [Write::_check\(\)](#), [Route::_check\(\)](#), [Release::_check\(\)](#), [Record::_check\(\)](#), [Decide::_check\(\)](#), [Assign::_check\(\)](#), and [Seize::_check\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.3 `clear()` `void Model::clear ()`

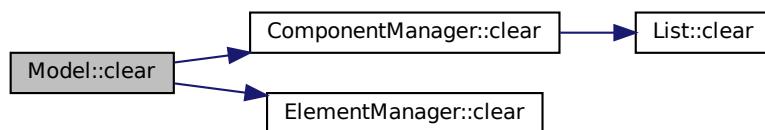
Definition at line 222 of file `Model.cpp`.

```
00222     {
00223         this->_componentManager->clear();
00224         this->elementManager->clear();
00225         //Util::ResetAllIds(); // \todo: To implement
00226     }
```

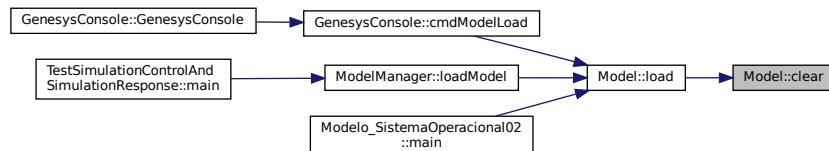
References `ComponentManager::clear()`, and `ElementManager::clear()`.

Referenced by `load()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.4 `getComponents()` `ComponentManager * Model::getComponents () const`

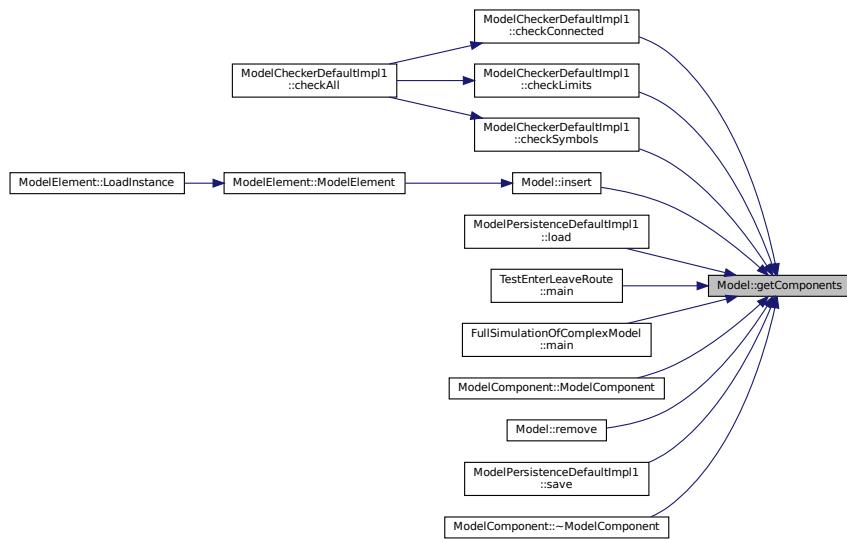
The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (`SourceComponentModel`).

Definition at line 322 of file `Model.cpp`.

```
00322     {
00323         return _componentManager;
00324     }
```

Referenced by `ModelCheckerDefaultImpl1::checkConnected()`, `ModelCheckerDefaultImpl1::checkLimits()`, `ModelCheckerDefaultImpl1::checkSymbols()`, `insert()`, `ModelPersistenceDefaultImpl1::load()`, `TestEnterLeaveRoute::main()`, `FullSimulationOfComplexModel::main()`, `ModelComponent::ModelComponent()`, `remove()`, `ModelPersistenceDefaultImpl1::save()`, and `ModelComponent::~ModelComponent()`.

Here is the caller graph for this function:



8.63.3.5 `getControls()`

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.

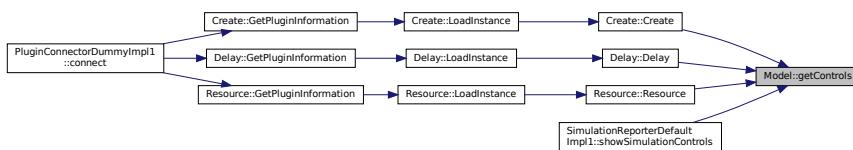
Definition at line 326 of file [Model.cpp](#).

```

00326
00327     return _controls;
00328 }
```

Referenced by [Create::Create\(\)](#), [Delay::Delay\(\)](#), [Resource::Resource\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationControls](#)

Here is the caller graph for this function:



8.63.3.6 getElements() `ElementManager * Model::getElements () const`

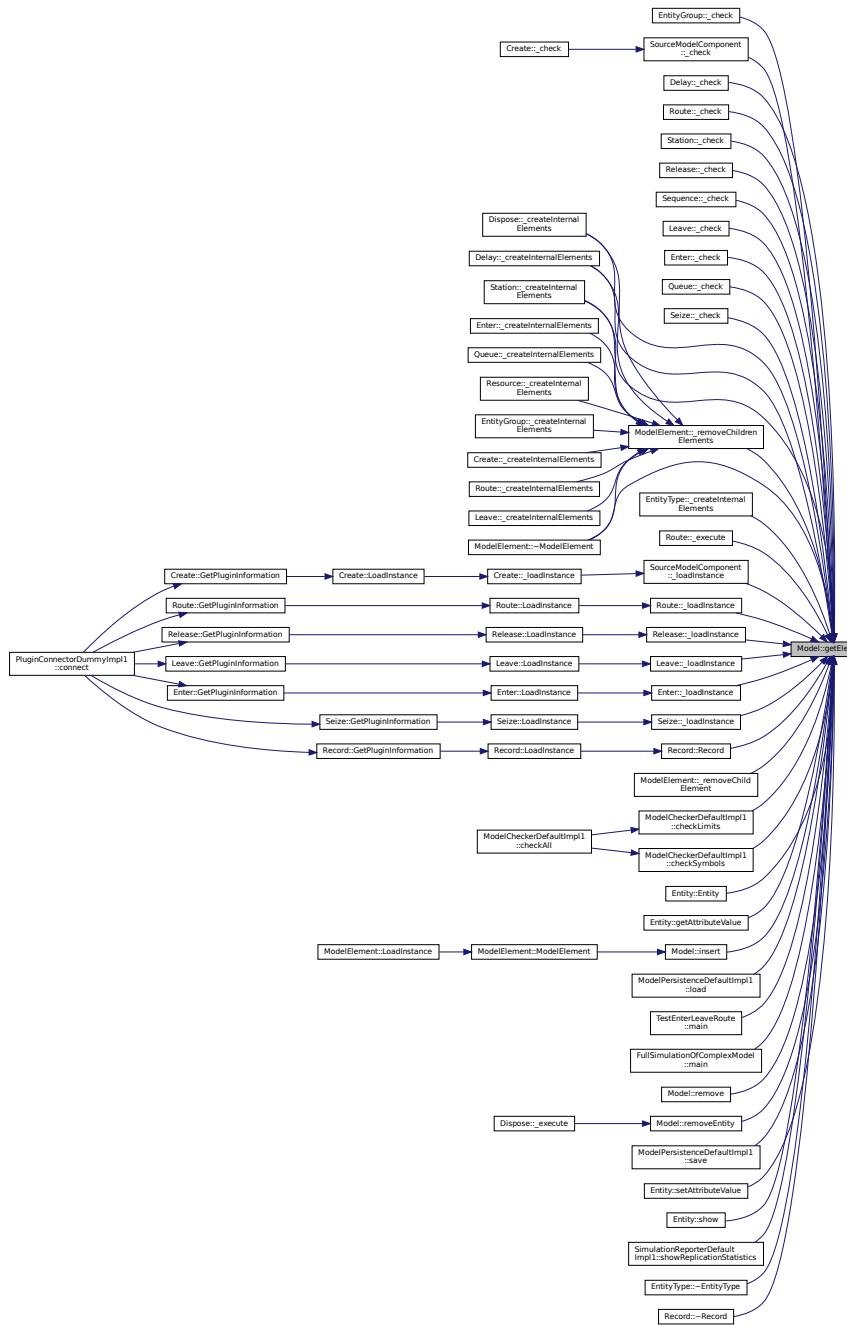
Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).

Definition at line 338 of file [Model.cpp](#).

```
00338     {  
00339         return _elementManager;  
00340     }
```

Referenced by [EntityGroup::_check\(\)](#), [SourceModelComponent::_check\(\)](#), [Delay::_check\(\)](#), [Route::_check\(\)](#), [Station::_check\(\)](#), [Release::_check\(\)](#), [Sequence::_check\(\)](#), [Leave::_check\(\)](#), [Enter::_check\(\)](#), [Queue::_check\(\)](#), [Seize::_check\(\)](#), [Dispose::_createInternalElements\(\)](#), [EntityType::_createInternalElements\(\)](#), [Delay::_createInternalElements\(\)](#), [Station::_createInternalElements\(\)](#), [Route::_execute\(\)](#), [SourceModelComponent::_loadInstance\(\)](#), [Route::_loadInstance\(\)](#), [Release::_loadInstance\(\)](#), [Leave::_loadInstance\(\)](#), [Enter::_loadInstance\(\)](#), [Seize::_loadInstance\(\)](#), [ModelElement::_removeChildElements\(\)](#), [ModelElement::_removeChildrenElements\(\)](#), [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [Entity::Entity\(\)](#), [Entity::getAttributeValue\(\)](#), [insert\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), [Record::Record\(\)](#), [remove\(\)](#), [removeEntity\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [Entity::setAttributeValue\(\)](#), [Entity::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [EntityType::~EntityType\(\)](#), [ModelElement::~ModelElement\(\)](#), and [Record::~Record\(\)](#).

Here is the caller graph for this function:



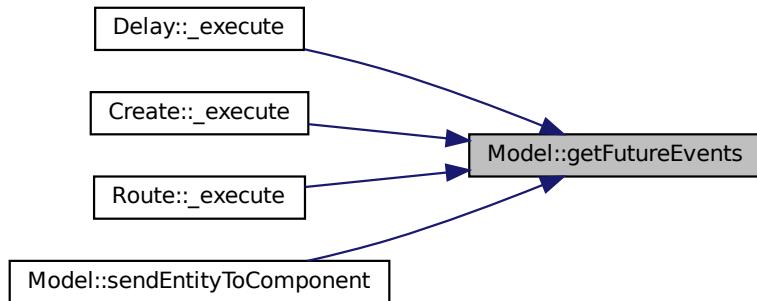
8.63.3.7 getFutureEvents() List< Event * > * Model::getFutureEvents () const

Definition at line 301 of file [Model.cpp](#).

```
0031
0032     return _futureEvents;
0033 }
```

Referenced by `Delay::execute()`, `Create::execute()`, `Route::execute()`, and `sendEntityToComponent()`.

Here is the caller graph for this function:



8.63.3.8 getId() [Util::identification](#) Model::getId () const

Definition at line 354 of file [Model.cpp](#).

```

00354
00355     return _id;
00356 }
```

8.63.3.9 getInfo() [ModelInfo](#) * Model::getInfo () const

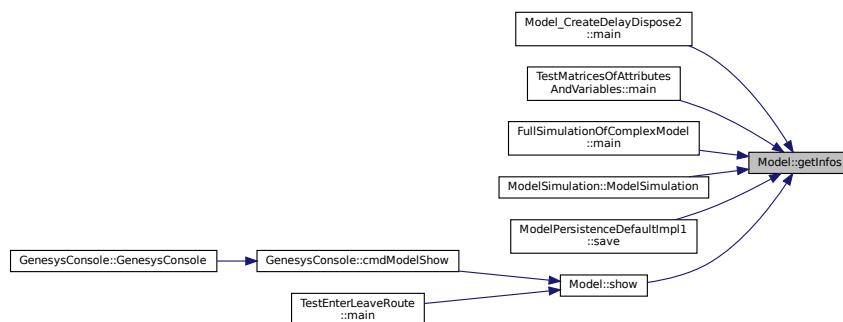
Definition at line 342 of file [Model.cpp](#).

```

00342
00343     return _modelInfo;
00344 }
```

Referenced by [Model_CreateDelayDispose2::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [FullSimulationOfComplexModel::ModelSimulation::ModelSimulation\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), and [show\(\)](#).

Here is the caller graph for this function:



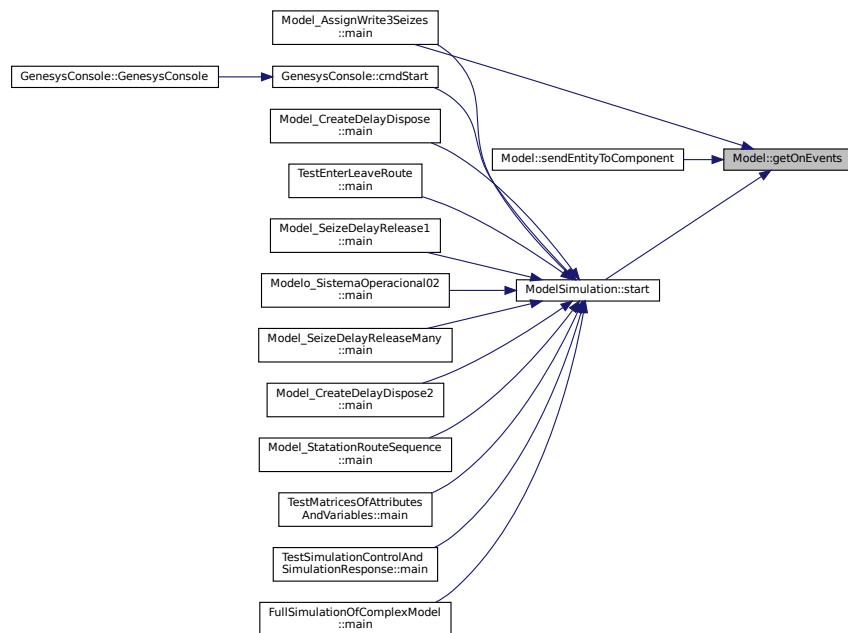
8.63.3.10 getOnEvents() `OnEventManager * Model::getOnEvents () const`

Definition at line 334 of file [Model.cpp](#).

```
00334                                     {
00335     return _eventManager;
00336 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [sendEntityToComponent\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



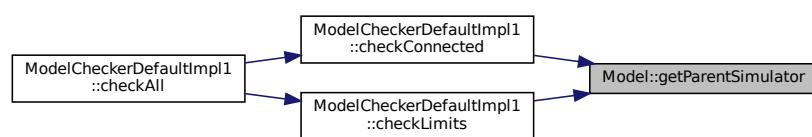
8.63.3.11 getParentSimulator() `Simulator * Model::getParentSimulator () const`

Definition at line 346 of file [Model.cpp](#).

```
00346                                     {
00347     return _parentSimulator;
00348 }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), and [ModelCheckerDefaultImpl1::checkLimits\(\)](#).

Here is the caller graph for this function:



8.63.3.12 `getResponses()` `List< SimulationResponse * > * Model::getResponses () const`

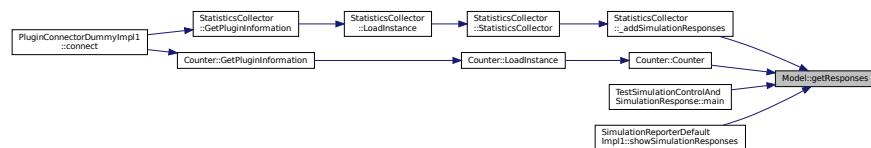
Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.

Definition at line 330 of file `Model.cpp`.

```
00330
00331     return _responses;
00332 }
```

Referenced by `StatisticsCollector::_addSimulationResponses()`, `Counter::Counter()`, `TestSimulationControlAndSimulationResponse::main()` and `SimulationReporterDefaultImpl1::showSimulationResponses()`.

Here is the caller graph for this function:



8.63.3.13 `getSimulation()` `ModelSimulation * Model::getSimulation () const`

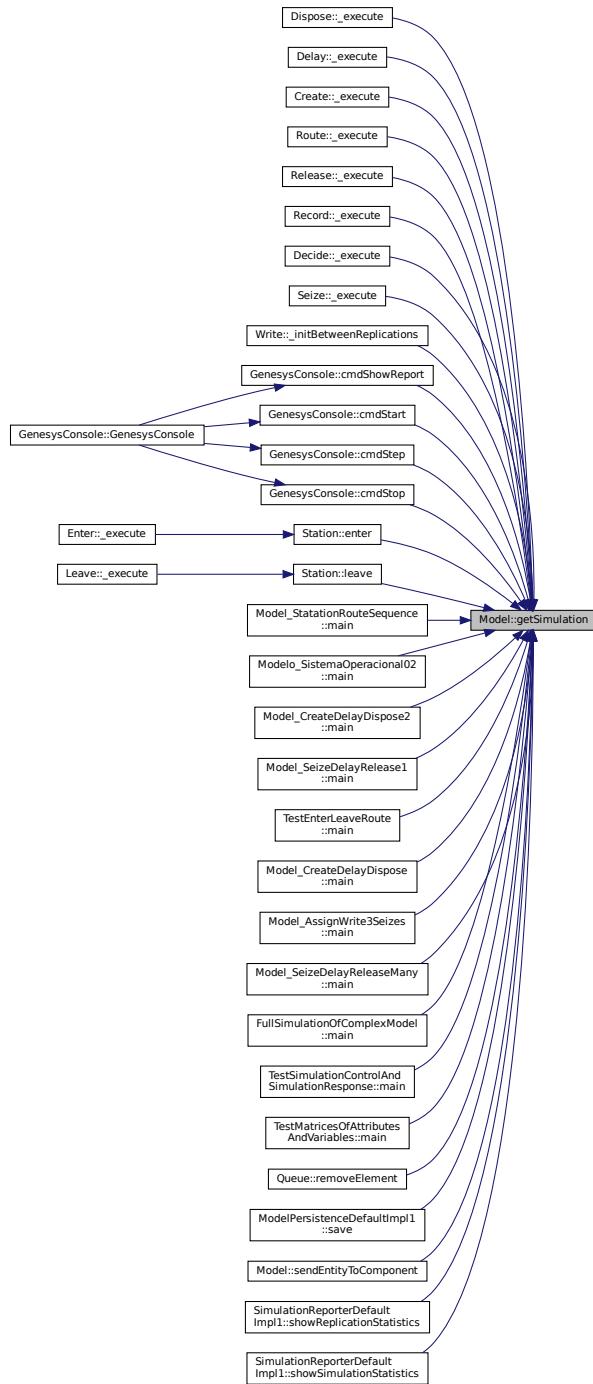
Provides access to the class that manages the model simulation.

Definition at line 350 of file `Model.cpp`.

```
00350
00351     return _simulation;
00352 }
```

Referenced by `Dispose::_execute()`, `Delay::_execute()`, `Create::_execute()`, `Route::_execute()`, `Release::_execute()`, `Record::_execute()`, `Decide::_execute()`, `Seize::_execute()`, `Write::_initBetweenReplications()`, `GenesysConsole::cmdShowReport()`, `GenesysConsole::cmdStart()`, `GenesysConsole::cmdStep()`, `GenesysConsole::cmdStop()`, `Station::enter()`, `Station::leave()`, `Model_StatationRouteSequence::main()`, `Modelo_SistemaOperacional02::main()`, `Model_CreateDelayDispose2::main()`, `Model_SeizeDelayReleaseMany::main()`, `TestEnterLeaveRoute::main()`, `Model_AssignWrite3Seizes::main()`, `Model_CreateDelayDispose::main()`, `Model_SeizeDelayRelease1::main()`, `TestMatricesOfAttributesAndVariables::main()`, `FullSimulationOfComplexModel::main()`, `TestSimulationControlAndSimulationResponse::main()`, `Queue::removeElement()`, `ModelPersistenceDefaultImpl1::save()`, `sendEntityToComponent()`, `SimulationReporterDefaultImpl1::showReplicationStatistics()`, and `SimulationReporterDefaultImpl1::showSimulationStatistics()`.

Here is the caller graph for this function:



8.63.3.14 `getTracer()`

`TraceManager * Model::getTracer() const`

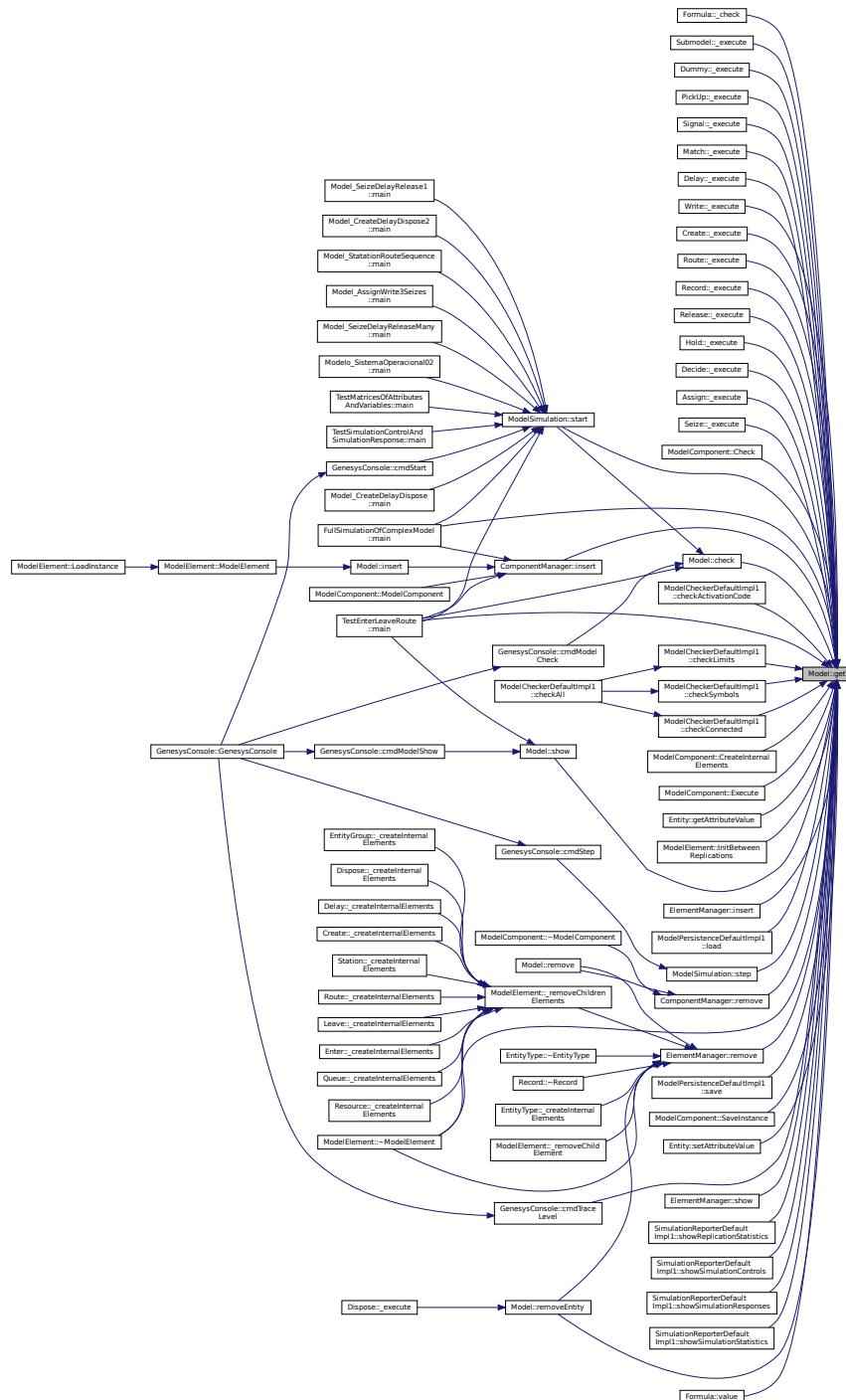
Provides access to the class that performs the trace of simulation and replications.

Definition at line 309 of file [Model.cpp](#).

```
00309         {
00310     return _traceManager;
00311 }
```

Referenced by [Formula::check\(\)](#), [Submodel::execute\(\)](#), [Dummy::execute\(\)](#), [PickUp::execute\(\)](#), [Signal::execute\(\)](#), [Match::execute\(\)](#), [Delay::execute\(\)](#), [Write::execute\(\)](#), [Create::execute\(\)](#), [Route::execute\(\)](#), [Record::execute\(\)](#), [Release::execute\(\)](#), [Hold::execute\(\)](#), [Decide::execute\(\)](#), [Assign::execute\(\)](#), [Seize::execute\(\)](#), [ModelComponent::Check\(\)](#), [check\(\)](#), [ModelCheckerDefaultImpl1::checkActivationCode\(\)](#), [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [GenesysConsole::cmdTraceLevel\(\)](#), [ModelComponent::CreateInternalElements\(\)](#), [ModelComponent::Execute\(\)](#), [Entity::getAttributeValue\(\)](#), [ModelElement::InitBetweenRe](#), [ComponentManager::insert\(\)](#), [ElementManager::insert\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), [ComponentManager::remove\(\)](#), [ElementManager::remove\(\)](#), [removeEntity\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [ModelComponent::SaveInstance\(\)](#), [Entity::setAttributeValue\(\)](#), [show\(\)](#), [ElementManager::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [SimulationReporterDefaultImpl1::showSimula](#), [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#), [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#), [ModelSimulation::start\(\)](#), [ModelSimulation::step\(\)](#), [Formula::value\(\)](#), and [ModelElement::~ModelElement\(\)](#).

Here is the caller graph for this function:



8.63.3.15 hasChanged() bool Model::hasChanged() const

Definition at line 313 of file [Model.cpp](#).

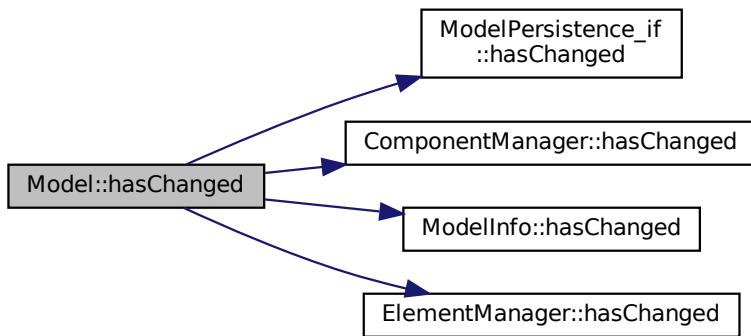
```
00313 {
00314     bool changed = _hasChanged;
```

```

00315     changed &= this->_componentManager->hasChanged\(\);
00316     changed &= this->_elementManager->hasChanged\(\);
00317     changed &= this->_modelInfo->hasChanged\(\);
00318     changed &= this->_modelPersistence->hasChanged\(\);
00319     return changed;
00320 }
```

References [ModelPersistence_if::hasChanged\(\)](#), [ComponentManager::hasChanged\(\)](#), [ModelInfo::hasChanged\(\)](#), and [ElementManager::hasChanged\(\)](#).

Here is the call graph for this function:



8.63.3.16 insert() `bool Model::insert (ModelElement * elemOrComp)`

Insert a new [ModelElement](#) or [ModelComponent](#) into the model (since 20191015). It's a generic access to [ComponentManager->insert\(\)](#) or [ModelElement->insert\(\)](#)

Definition at line 151 of file [Model.cpp](#).

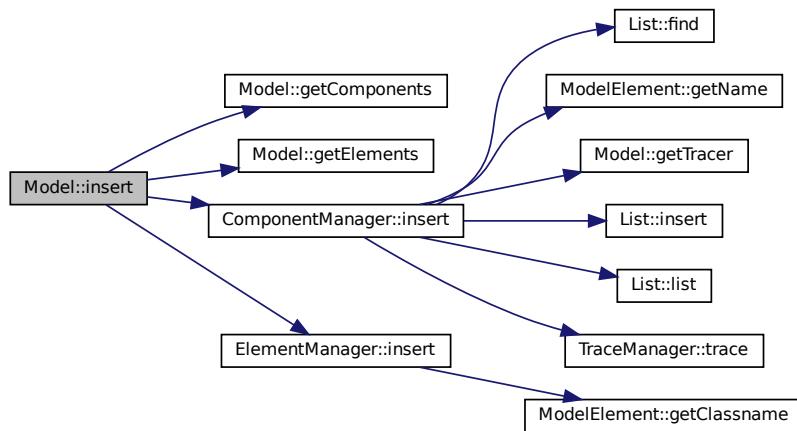
```

00151
00152     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00153     {
00154         if (comp == nullptr) // it's a ModelElement
00155             return this->getElements\(\)->insert (elemOrComp);
00156         else // it's a ModelComponent
00157             return this->getComponents\(\)->insert (comp);
  
```

References [getComponents\(\)](#), [getElements\(\)](#), [ComponentManager::insert\(\)](#), and [ElementManager::insert\(\)](#).

Referenced by [ModelElement::ModelElement\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.17 `load()` `bool Model::load (std::string filename)`

Definition at line 100 of file [Model.cpp](#).

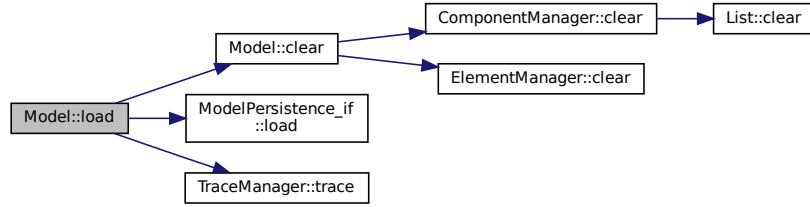
```

00100
00101     this->clear();
00102     bool res = this->_modelPersistence->load(filename);
00103     if (res)
00104         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model successfully loaded");
00105     else
00106         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model could not be loaded");
00107     return res;
00108 }
```

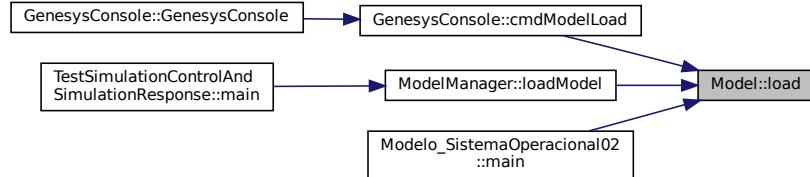
References [clear\(\)](#), [ModelPersistence_if::load\(\)](#), [Util::modelResult](#), and [TraceManager::trace\(\)](#).

Referenced by [GenesysConsole::cmdModelLoad\(\)](#), [ModelManager::loadModel\(\)](#), and [Modelo_SistemaOperacional02::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.18 parseExpression() [1/2] double Model::parseExpression (const std::string expression)

Definition at line 110 of file [Model.cpp](#).

```

00110
00111     try {
00112         return _parser->parse(expression);
00113     } catch (...) {
00114         return 0.0; // \todo: HOW SAY THERE WAS AN ERROR?
00115     }
00116 }
```

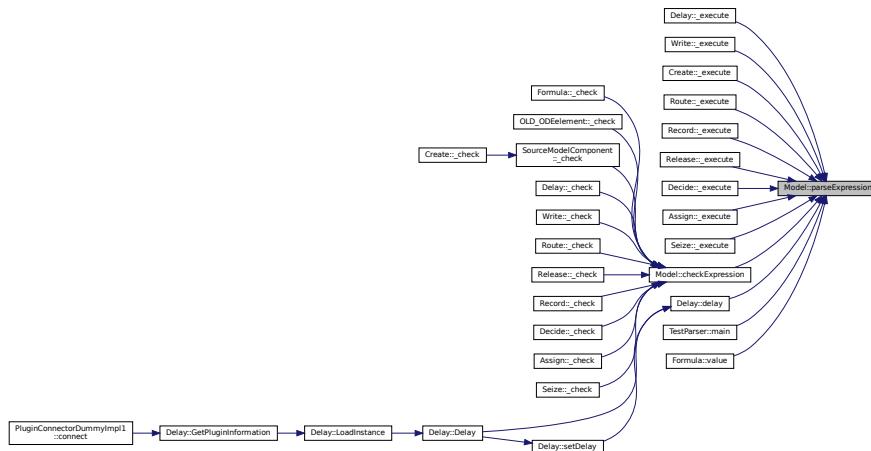
References [Parser_if::parse\(\)](#).

Referenced by [Delay::_execute\(\)](#), [Write::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Record::_execute\(\)](#), [Release::_execute\(\)](#), [Decide::_execute\(\)](#), [Assign::_execute\(\)](#), [Seize::_execute\(\)](#), [checkExpression\(\)](#), [Delay::delay\(\)](#), [TestParser::main\(\)](#), and [Formula::value\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.19 parseExpression() [2/2] `double Model::parseExpression (const std::string expression, bool * success, std::string * errorMessage)`

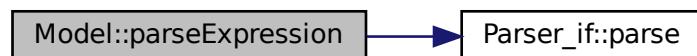
Definition at line 128 of file [Model.cpp](#).

```

00128     double value = _parser->parse (expression, success, errorMessage);
00129     return value;
00130 }
00131 }
```

References [Parser_if::parse\(\)](#).

Here is the call graph for this function:



8.63.3.20 remove() `void Model::remove (ModelElement * elemOrComp)`

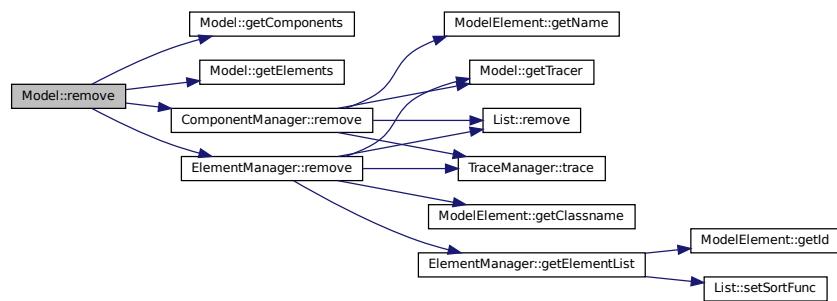
Remove a new [ModelElement](#) or [ModelComponent](#) into the model (since 20191015). It's a generic access to ComponentManager->[remove\(\)](#) or ModelElement->[remove\(\)](#)

Definition at line 159 of file [Model.cpp](#).

```
00159     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00160     if (comp == nullptr) // it's a ModelElement
00161         this->getElements ()->remove (elemOrComp);
00162     else // it's a ModelComponent
00163         this->getComponents ()->remove (comp);
00164
00165 }
```

References [getComponents\(\)](#), [getElements\(\)](#), [ComponentManager::remove\(\)](#), and [ElementManager::remove\(\)](#).

Here is the call graph for this function:



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

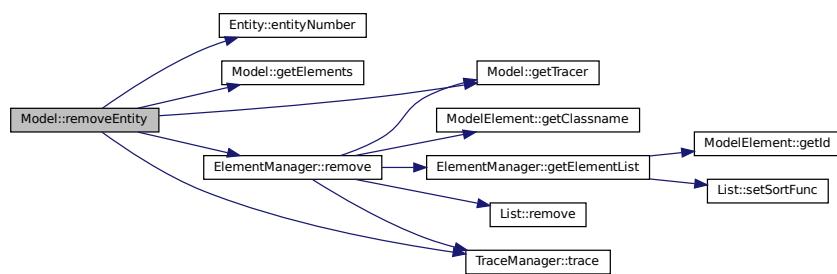
Definition at line 293 of file [Model.cpp](#).

```
00293
00294     /* \todo: -: event onEntityRemove */
00295     /* \todo: -: collectStatistics */
00296     std::string entId = std::to_string(entity->entityNumber ());
00297     this->getElements ()->remove (Util::TypeOf<Entity> (), entity);
00298     getTracer ()->trace ("Entity " + entId + " was removed from the system");
00299 }
```

References [Entity::entityNumber\(\)](#), [getElements\(\)](#), [getTracer\(\)](#), [ElementManager::remove\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [Dispose::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.63.3.22 save() `bool Model::save (std::string filename)`

Definition at line 89 of file [Model.cpp](#).

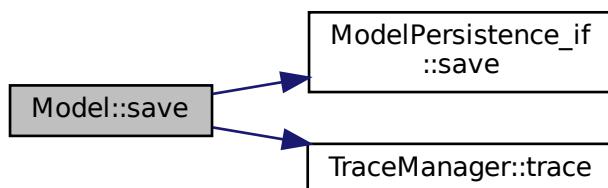
```

00089         {
00090     bool res = this->_modelPersistence->save(filename);
00091     if (res) {
00092         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model successfully saved");
00093     } else {
00094         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model could not be saved");
00095     }
00096 }
00097 return res;
00098 }
```

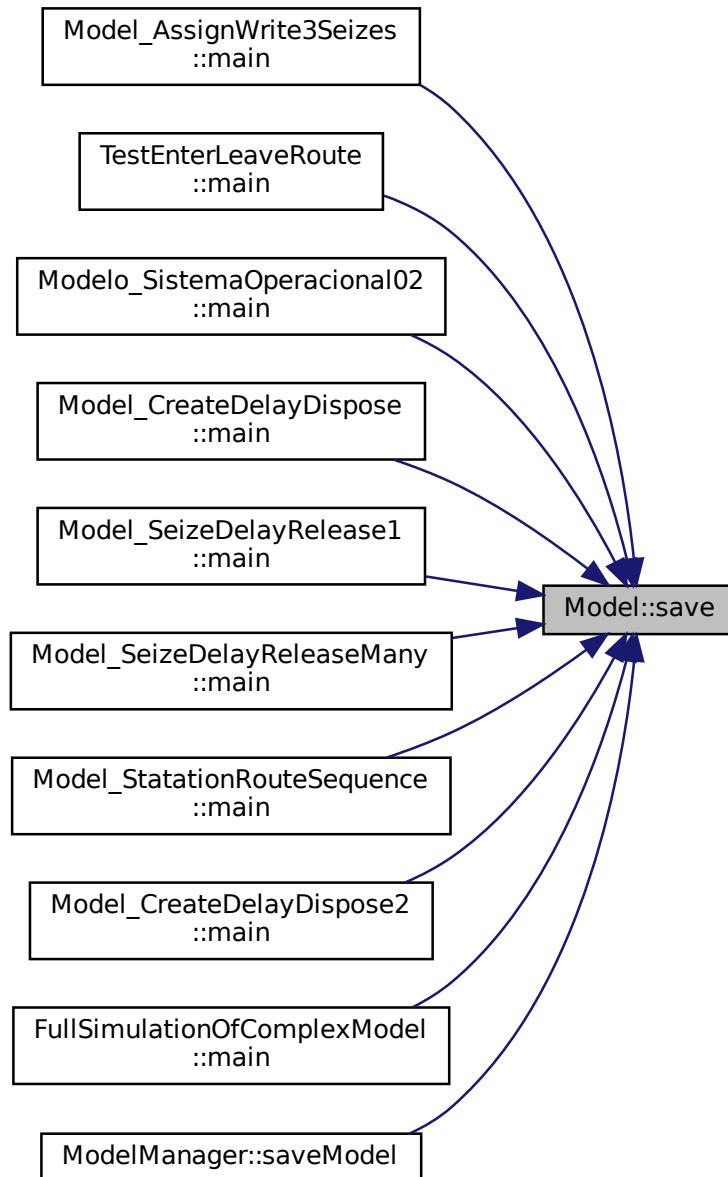
References [Util::modelResult](#), [ModelPersistence_if::save\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_StationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), and [ModelManager::saveModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```

8.63.3.23 sendEntityToComponent() [1/2] void Model::sendEntityToComponent (
    Entity * entity,
    Connection * connection,
    double timeDelay )
  
```

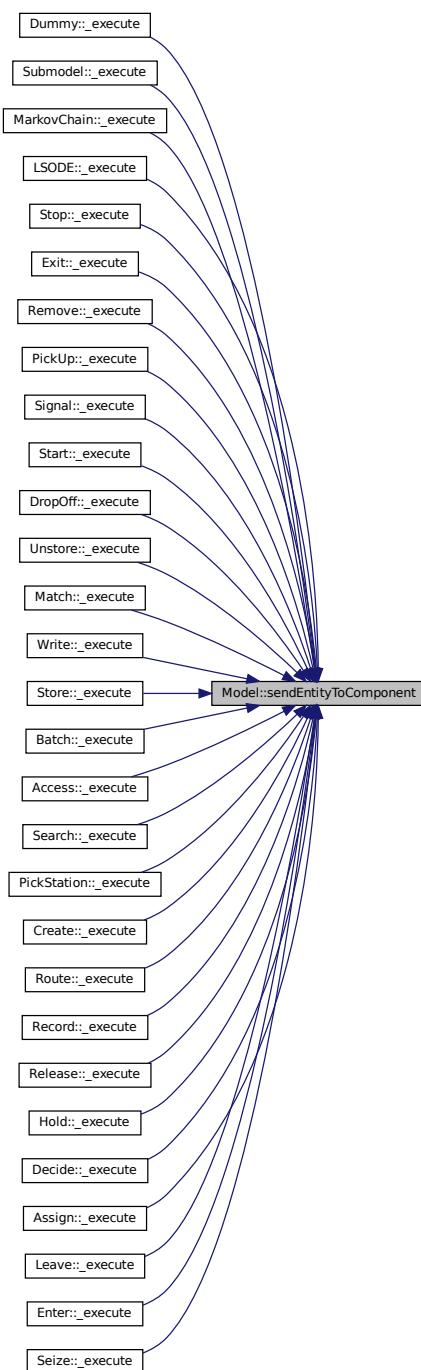
Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

Definition at line 70 of file [Model.cpp](#).

```
00070     this->sendEntityToComponent(entity, connection->first, timeDelay, connection->second);  
00071 }  
00072 }
```

Referenced by [Dummy::_execute\(\)](#), [Submodel::_execute\(\)](#), [MarkovChain::_execute\(\)](#), [LSODE::_execute\(\)](#), [Stop::_execute\(\)](#), [Exit::_execute\(\)](#), [Remove::_execute\(\)](#), [PickUp::_execute\(\)](#), [Signal::_execute\(\)](#), [Start::_execute\(\)](#), [DropOff::_execute\(\)](#), [Unstore::_execute\(\)](#), [Match::_execute\(\)](#), [Write::_execute\(\)](#), [Store::_execute\(\)](#), [Batch::_execute\(\)](#), [Access::_execute\(\)](#), [Search::_execute\(\)](#), [PickStation::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Release::_execute\(\)](#), [Record::_execute\(\)](#), [Hold::_execute\(\)](#), [Decide::_execute\(\)](#), [Assign::_execute\(\)](#), [Leave::_execute\(\)](#), [Enter::_execute\(\)](#), and [Seize::_execute\(\)](#).

Here is the caller graph for this function:



```
8.63.3.24 sendEntityToComponent() [2/2] void Model::sendEntityToComponent (
    Entity * entity,
    ModelComponent * component,
    double timeDelay,
    unsigned int componentInputNumber = 0 )
```

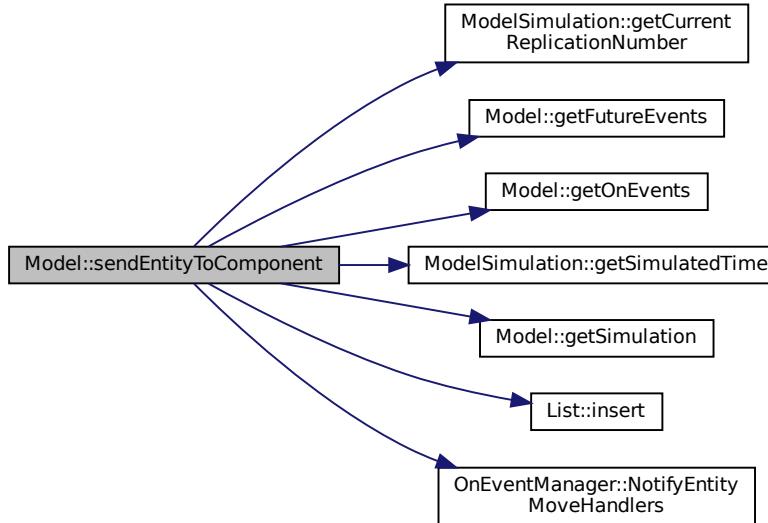
Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

Definition at line 74 of file [Model.cpp](#).

```
00074
00075     {
00076         this->getOnEvents()->NotifyEntityMoveHandlers(new
00077             SimulationEvent(_simulation->getCurrentReplicationNumber(), new
00078                 Event(_simulation->getSimulatedTime(), entity, component, componentInputNumber))); //\todo: Event
00079             should include information about "from component" and timeDelay, but it doesn't
00080             // schedule to send it
00081             Event* newEvent = new Event(this->getSimulation()->getSimulatedTime() + timeDelay, entity,
00082                 component, componentInputNumber);
00083             this->getFutureEvents()->insert(newEvent);
00084     }
```

References [ModelSimulation::getCurrentReplicationNumber\(\)](#), [getFutureEvents\(\)](#), [getOnEvents\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [getSimulation\(\)](#), [List< T >::insert\(\)](#), and [OnEventManager::NotifyEntityMoveHandlers\(\)](#).

Here is the call graph for this function:



```
8.63.3.25 setTracer() void Model::setTracer (
    TraceManager * _traceManager )
```

Definition at line 305 of file [Model.cpp](#).

```
00305
00306     this->_traceManager = _traceManager;
00307 }
```

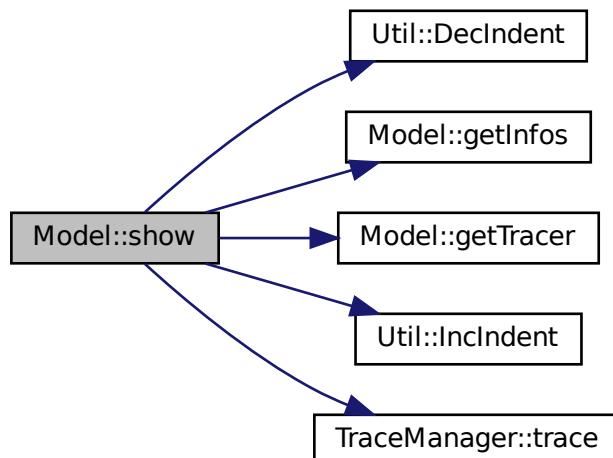
8.63.3.26 show() void Model::show ()Definition at line 133 of file [Model.cpp](#).

```

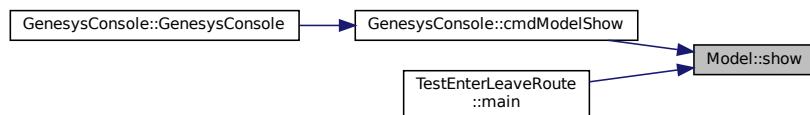
00133     {
00134         getTracer ()->trace(Util::TraceLevel::report, "Simulation Model:");
00135         Util::IncIndent ();
00136     {
00137         getTracer ()->trace(Util::TraceLevel::report, "Information:");
00138         Util::IncIndent ();
00139         getTracer ()->trace(Util::TraceLevel::report, this->getInfos ()->show ());
00140         Util::DecIndent ();
00141         _showConnections ();
00142         _showComponents ();
00143         _showElements ();
00144         _showSimulationControls ();
00145         _showSimulationResponses ();
00146     }
00147     Util::DecIndent ();
00148     getTracer ()->trace(Util::TraceLevel::report, "End of Simulation Model");
00149 }
```

References [Util::DecIndent\(\)](#), [getInfos\(\)](#), [getTracer\(\)](#), [Util::IncIndent\(\)](#), [Util::report](#), and [TraceManager::trace\(\)](#).Referenced by [GenesysConsole::cmdModelShow\(\)](#), and [TestEnterLeaveRoute::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



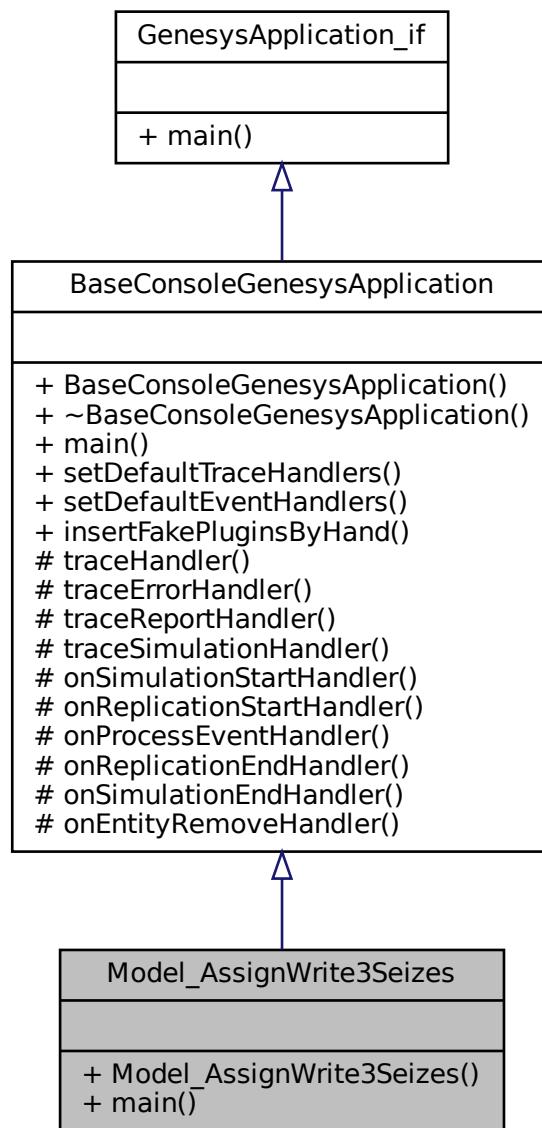
The documentation for this class was generated from the following files:

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

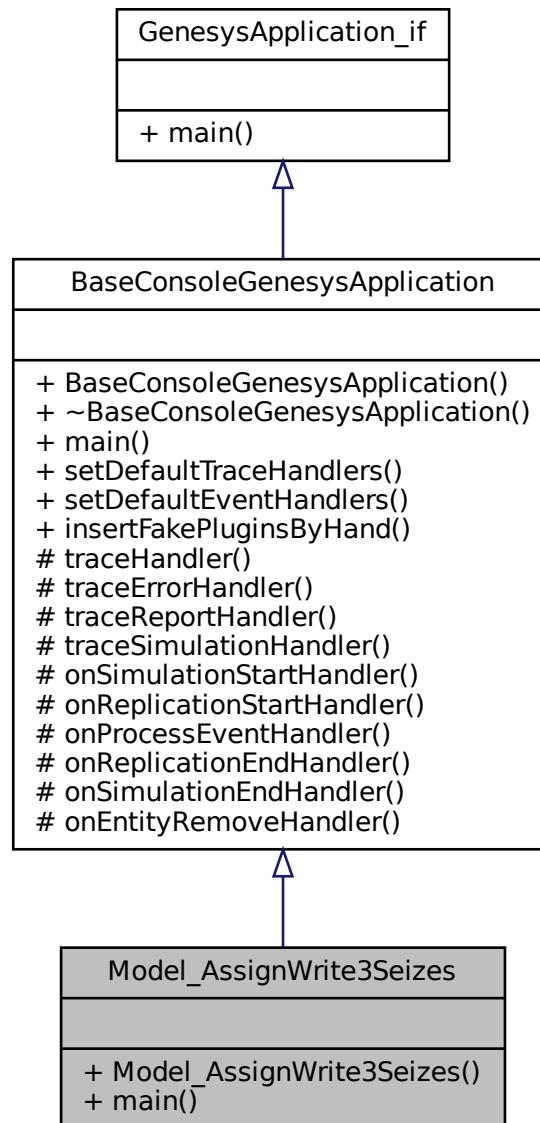
8.64 Model_AssignWrite3Seizes Class Reference

```
#include <Model_AssignWrite3Seizes.h>
```

Inheritance diagram for Model_AssignWrite3Seizes:



Collaboration diagram for Model_AssignWrite3Seizes:



Public Member Functions

- [Model_AssignWrite3Seizes \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

8.64.1 Detailed Description

Definition at line 19 of file [Model_AssignWrite3Seizes.h](#).

8.64.2 Constructor & Destructor Documentation

8.64.2.1 Model_AssignWrite3Seizes() Model_AssignWrite3Seizes::Model_AssignWrite3Seizes ()

Definition at line 39 of file [Model_AssignWrite3Seizes.cpp](#).

```
00039     {
00040 }
```

8.64.3 Member Function Documentation

8.64.3.1 main() int Model_AssignWrite3Seizes::main (

```
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 42 of file [Model_AssignWrite3Seizes.cpp](#).

```
00042     {
00043     Simulator* genesys = new Simulator();
00044     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed); //modelResult);
//componentArrival;
00045     this->setDefaultTraceHandlers(genesys->getTracer());
00046     this->insertFakePluginsByHand(genesys);
00047
00048     Model* model = genesys->getModels()->newModel();
00049     this->setDefaultEventHandlers(model->getOnEvents());
00050
00051     // build the simulation model
00052     ModelSimulation* sim = model->getSimulation();
00053     sim->setNumberOfReplications(5);
00054     sim->setReplicationLength(100);
00055     EntityType* part = new EntityType(model, "Part");
00056     // model->insert(part);
00057     Create* create1 = new Create(model);
00058     create1->setEntityType(part);
00059     create1-> setTimeBetweenCreationsExpression("norm(1.5,0.5)");
00060     create1-> setTimeUnit(Util::TimeUnit::second);
00061     create1->setEntitiesPerCreation(1);
00062     // model->insert(create1);
00063     Assign* assign1 = new Assign(model);
00064     assign1->getAssignments()->insert(new Assign::Assignment("varNextIndex", "varNextIndex + 1"));
00065     assign1->getAssignments()->insert(new Assign::Assignment("index", "varNextIndex"));
00066     // model->insert(assign1);
00067     Attribute* attr1 = new Attribute(model, "index");
00068     // model->insert(attr1);
00069     Variable* var1 = new Variable(model, "varNextIndex");
00070     // model->insert(var1);
00071     Write* write1 = new Write(model);
00072     write1->setWriteToType(Write::WriteToType::SCREEN);
00073     write1->writeElements()->insert(new WriteElement("Atributo index: "));
00074     write1->writeElements()->insert(new WriteElement("index", true, true));
00075     write1->writeElements()->insert(new WriteElement("Variável nextIndex: "));
00076     write1->writeElements()->insert(new WriteElement("varNextIndex", true, true));
00077     write1->writeElements()->insert(new WriteElement("Quantidade ocupada das máquinas: "));
00078     write1->writeElements()->insert(new WriteElement("NR(Machine_1)", true));
00079     write1->writeElements()->insert(new WriteElement(" ", ""));
00080     write1->writeElements()->insert(new WriteElement("NR(Machine_2)", true));
00081     write1->writeElements()->insert(new WriteElement(" ", ""));
00082     write1->writeElements()->insert(new WriteElement("NR(Machine_3)", true, true));
00083     write1->writeElements()->insert(new WriteElement("Estado das máquinas: "));
00084     write1->writeElements()->insert(new WriteElement("STATE(Machine_1)", true));
00085     write1->writeElements()->insert(new WriteElement(" ", ""));
00086     write1->writeElements()->insert(new WriteElement("STATE(Machine_2)", true));
00087     write1->writeElements()->insert(new WriteElement(" ", ""));
00088     write1->writeElements()->insert(new WriteElement("STATE(Machine_3)", true, true));
00089     write1->writeElements()->insert(new WriteElement("Quantidade de máquinas ocupadas no Set: "));
00090     write1->writeElements()->insert(new WriteElement("SETSUM(Machine_Set)", true, true));
```

```

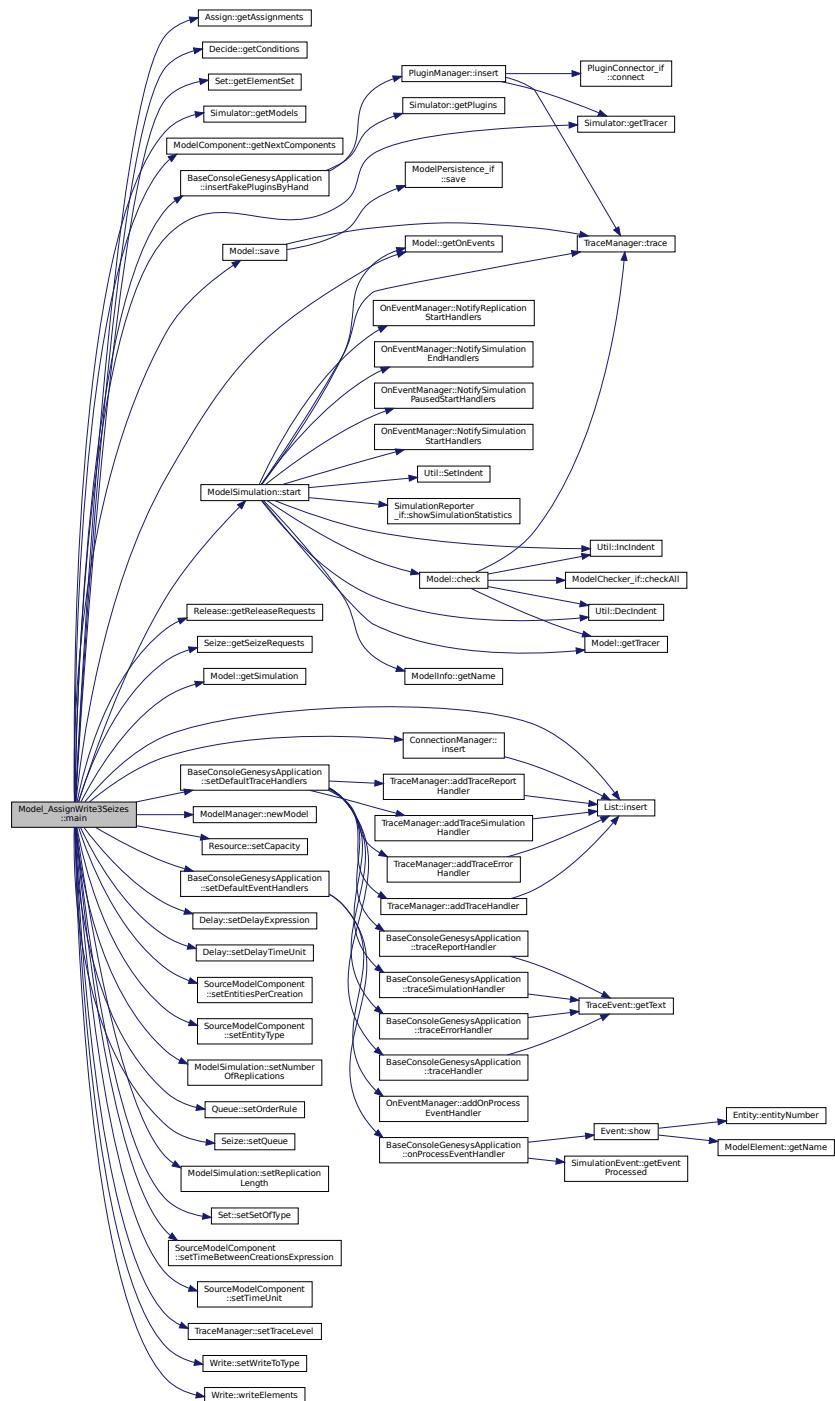
00091     write1->writeElements()->insert(new WriteElement("Quantidade de entidades na fila 3: "));
00092     write1->writeElements()->insert(new WriteElement("NQ(Queue_Seize_3)", true, true));
00093     write1->writeElements()->insert(new WriteElement("Somatório do atributo 'index' das entidades na
00094     fila 3: "));
00095     write1->writeElements()->insert(new WriteElement("SAQUE(Queue_Seize_3,index)", true, true));
00096     write1->writeElements()->insert(new WriteElement("Tempo médio das entidades na fila 3: "));
00097     write1->writeElements()->insert(new WriteElement("TAVG(Queue_Seize_3.TimeInQueue)", true, true));
00098     // model->insert(write1);
00099     //
00100    //
00101    Resource* machine1 = new Resource(model, "Machine_1");
00102    machine1->setCapacity(1);
00103    // model->insert(machine1);
00104    Resource* machine2 = new Resource(model, "Machine_2");
00105    machine2->setCapacity(2);
00106    // model->insert(machine2);
00107    Resource* machine3 = new Resource(model, "Machine_3");
00108    machine3->setCapacity(3);
00109    // model->insert(machine3);
00110    Set* machSet = new Set(model, "Machine_Set");
00111    machSet->setSetOfType(Util::TypeOf<Resource>());
00112    machSet->getElementSet()->insert(machine1);
00113    machSet->getElementSet()->insert(machine2);
00114    machSet->getElementSet()->insert(machine3);
00115    // model->insert(machSet);
00116    Decide* decide1 = new Decide(model);
00117    decide1->getConditions()->insert("NR(Machine_1) < MR(Machine_1)");
00118    decide1->getConditions()->insert("NR(Machine_2) < MR(Machine_2)");
00119    // model->insert(decide1);
00120    Queue* queueSeize1 = new Queue(model, "Queue_Seize_1");
00121    queueSeize1->setOrderRule(Queue::OrderRule::FIFO);
00122    // model->insert(queueSeize1);
00123    Seize* seize1 = new Seize(model);
00124    seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine1));
00125    seize1->setQueue(queueSeize1);
00126    // model->insert(seize1);
00127    Delay* delay1 = new Delay(model);
00128    delay1->setDelayExpression("norm(15,1)");
00129    delay1->setDelayTimeUnit(Util::TimeUnit::second);
00130    // model->insert(delay1);
00131    Release* release1 = new Release(model);
00132    release1->getReleaseRequests()->insert(new SeizableItemRequest(machine1));
00133    // model->insert(release1);
00134    Queue* queueSeize2 = new Queue(model, "Queue_Seize_2");
00135    queueSeize2->setOrderRule(Queue::OrderRule::FIFO);
00136    // model->insert(queueSeize2);
00137    Seize* seize2 = new Seize(model);
00138    seize2->getSeizeRequests()->insert(new SeizableItemRequest(machine2));
00139    seize2->setQueue(queueSeize2);
00140    // model->insert(seize2);
00141    Delay* delay2 = new Delay(model);
00142    delay2->setDelayExpression("norm(15,1)");
00143    delay2->setDelayTimeUnit(Util::TimeUnit::second);
00144    // model->insert(delay2);
00145    Release* release2 = new Release(model);
00146    release2->getReleaseRequests()->insert(new SeizableItemRequest(machine2));
00147    // model->insert(release2);
00148    Queue* queueSeize3 = new Queue(model, "Queue_Seize_3");
00149    queueSeize3->setOrderRule(Queue::OrderRule::FIFO);
00150    // model->insert(queueSeize3);
00151    Seize* seize3 = new Seize(model);
00152    seize3->getSeizeRequests()->insert(new SeizableItemRequest(machine3));
00153    seize3->setQueue(queueSeize3);
00154    // model->insert(seize3);
00155    Delay* delay3 = new Delay(model);
00156    delay3->setDelayExpression("norm(15,1)");
00157    delay3->setDelayTimeUnit(Util::TimeUnit::second);
00158    // model->insert(delay3);
00159    Release* release3 = new Release(model);
00160    release3->getReleaseRequests()->insert(new SeizableItemRequest(machine3));
00161    // model->insert(release3);
00162    Dispose* dispose1 = new Dispose(model);
00163    // model->insert(dispose1);
00164    //
00165    create1->getNextComponents()->insert(assign1);
00166    assign1->getNextComponents()->insert(write1);
00167    write1->getNextComponents()->insert(decide1);
00168    decide1->getNextComponents()->insert(seize1);
00169    decide1->getNextComponents()->insert(seize2);
00170    decide1->getNextComponents()->insert(seize3);
00171    seize1->getNextComponents()->insert(delay1);
00172    delay1->getNextComponents()->insert(release1);
00173    release1->getNextComponents()->insert(dispose1);
00174    seize2->getNextComponents()->insert(delay2);
00175    delay2->getNextComponents()->insert(release2);

```

```
00176     release2->getNextComponents()->insert(dispose1);
00177     seize3->getNextComponents()->insert(delay3);
00178     delay3->getNextComponents()->insert(release3);
00179     release3->getNextComponents()->insert(dispose1);
00180     //
00181     model->save("../temp/forthExampleOfSimulation.txt");
00182     sim->start();
00183     return 0;
00184 }
```

References Util::everythingMostDetailed, Queue::FIFO, Assign::getAssignments(), Decide::getConditions(), Set::getElementSet(), Simulator::getModels(), ModelComponent::getNextComponents(), Model::getOnEvents(), Release::getReleaseRequests(), Seize::getSeizeRequests(), Model::getSimulation(), Simulator::getTracer(), ConnectionManager::insert(), List< T >::insert(), BaseConsoleGenesysApplication::insertFakePluginsByHand(), ModelManager::newModel(), Model::save(), Write::SCREEN, Util::second, Resource::setCapacity(), BaseConsoleGenesysApplication::BaseConsoleGenesysApplication::setDefaultTraceHandlers(), Delay::setDelayExpression(), Delay::setDelayTimeUnit(), SourceModelComponent::setEntitiesPerCreation(), SourceModelComponent::setEntityType(), ModelSimulation::setNumberOfReplicas(), Queue::setOrderRule(), Seize::setQueue(), ModelSimulation::setReplicationLength(), Set::setSetOfType(), SourceModelComponent::setTimeBetweenCreationsExpression(), SourceModelComponent::setTimeUnit(), TraceManager::setTraceLevel(), Write::setWriteToType(), ModelSimulation::start(), and Write::writeElements().

Here is the call graph for this function:



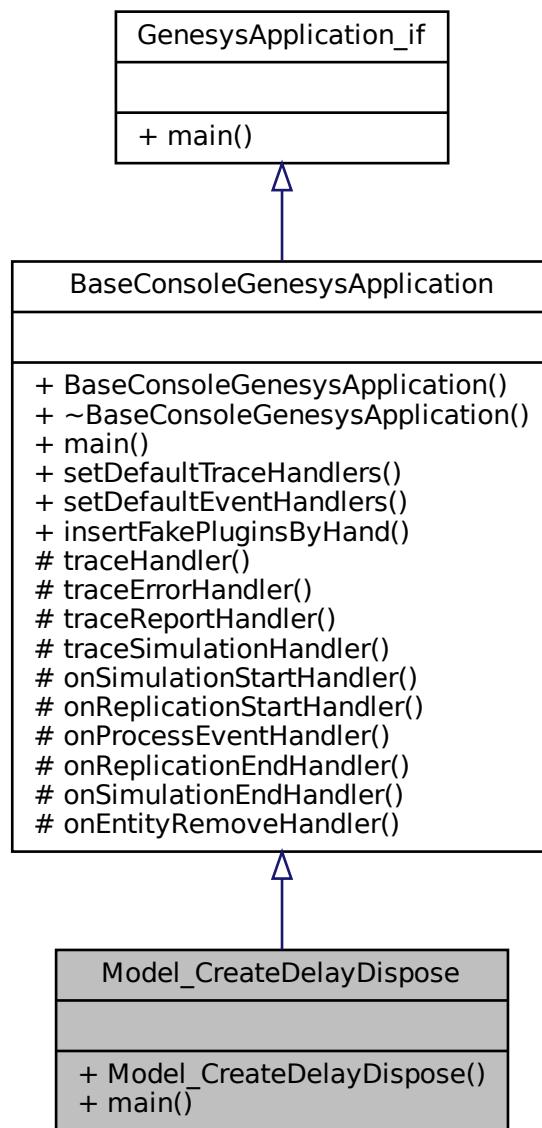
The documentation for this class was generated from the following files:

- [Model_AssignWrite3Seizes.h](#)
- [Model_AssignWrite3Seizes.cpp](#)

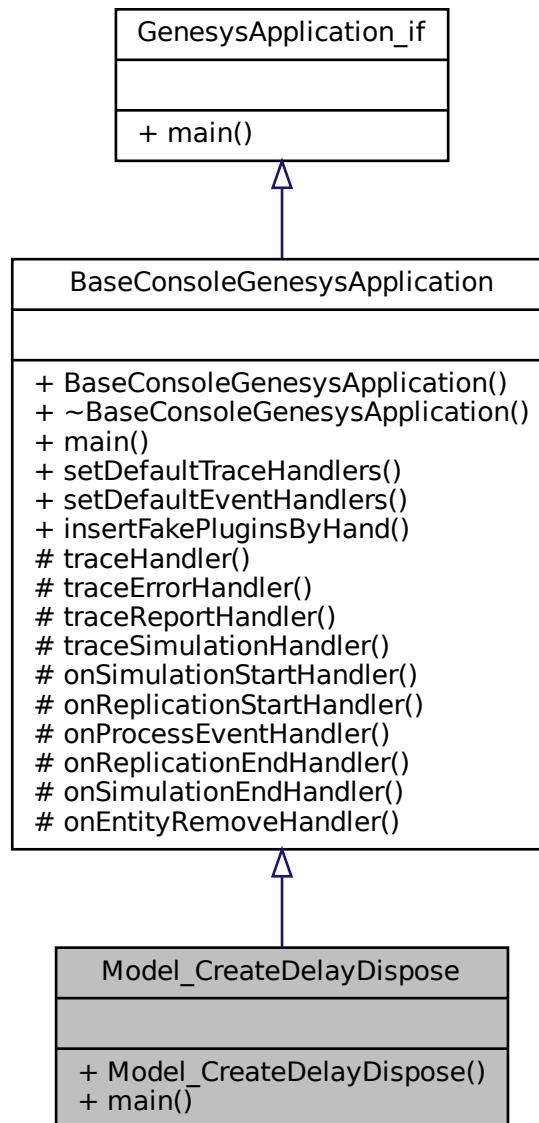
8.65 Model_CreateDelayDispose Class Reference

```
#include <Model_CreateDelayDispose.h>
```

Inheritance diagram for Model_CreateDelayDispose:



Collaboration diagram for Model_CreateDelayDispose:



Public Member Functions

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

Additional Inherited Members

8.65.1 Detailed Description

Definition at line 19 of file [Model_CreateDelayDispose.h](#).

8.65.2 Constructor & Destructor Documentation

8.65.2.1 Model_CreateDelayDispose() Model_CreateDelayDispose::Model_CreateDelayDispose ()

Definition at line 29 of file [Model_CreateDelayDispose.cpp](#).

```
00029     {  
00030 }
```

8.65.3 Member Function Documentation

8.65.3.1 main() int Model_CreateDelayDispose::main (

```
    int argc,  
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

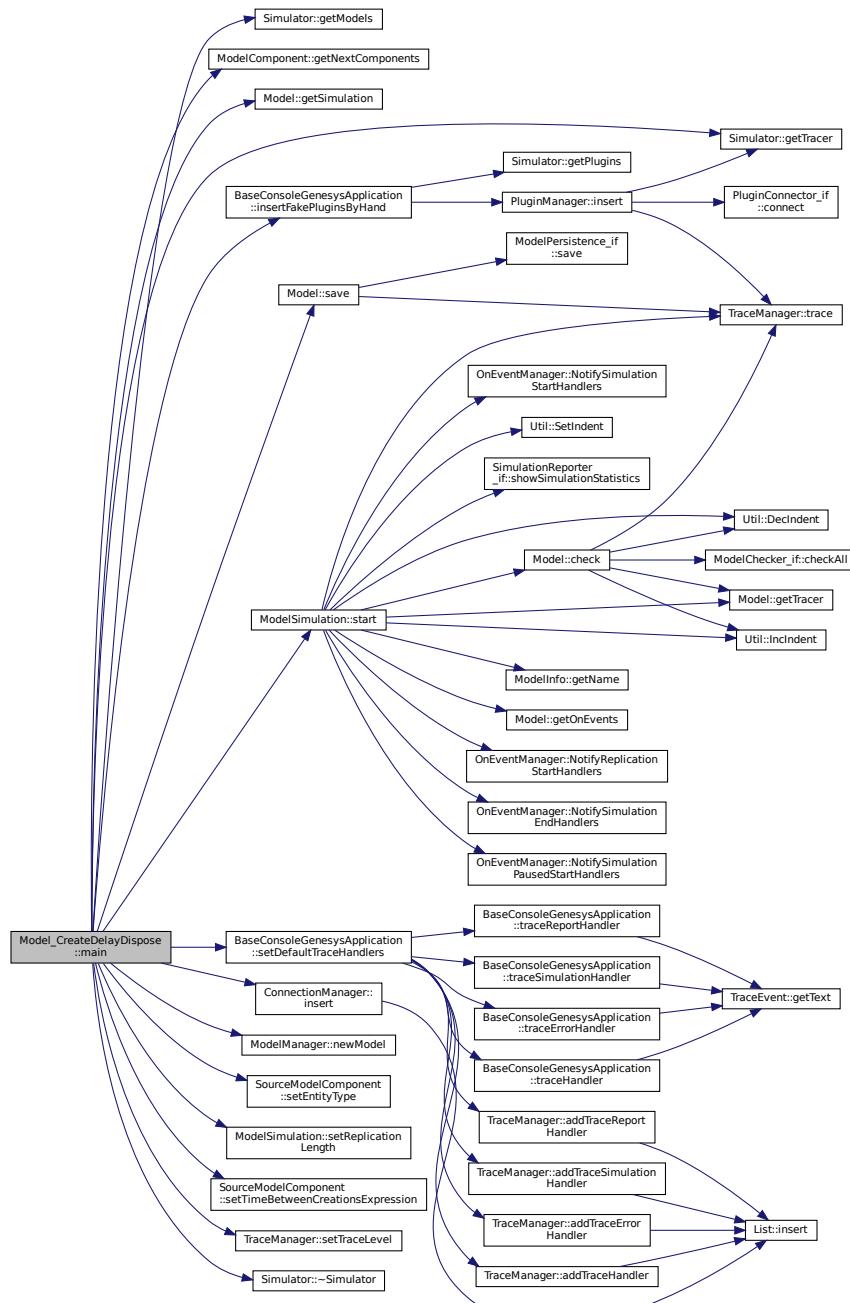
Implements [BaseConsoleGenesysApplication](#).

Definition at line 36 of file [Model_CreateDelayDispose.cpp](#).

```
00036     {  
00037     Simulator* genesys = new Simulator();  
00038     // Handle traces and simulation events to output them  
00039     this->setDefaultTraceHandlers(genesys->getTracer());  
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::componentDetailed);  
00041     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet  
00042     this->insertFakePluginsByHand(genesys);  
00043     Model* model = genesys->getModels()->newModel();  
00044     //  
00045     // build the simulation model  
00046     // if no ModelInfo is provided, then the model will be simulated once (one replication) and the  
replication length will be 3600 seconds (simulated time)  
00047     model->getSimulation()->setReplicationLength(60);  
00048     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows  
00049     EntityType* entityTypel = new EntityType(model, "Type_of_Representative_Entity");  
00050     // create a ModelComponent of type Create, used to insert entities into the model  
00051     Create* createl = new Create(model);  
00052     createl->setEntityType(entityTypel);  
00053     createl-> setTimeBetweenCreationsExpression("1.5"); // create one new entity every 1.5 seconds  
00054     // create a ModelComponent of type Delay, used to represent a time delay  
00055     Delay* delayl = new Delay(model);  
00056     // if nothing else is set, the delay will take 1 second  
00057     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model  
00058     Dispose* disposel = new Dispose(model); // insert the component into the model  
00059     // connect model components to create a "workflow" -- should always start from a  
SourceModelComponent and end at a SinkModelComponent (it will be checked)  
00060     createl->getNextComponents()->insert(delayl);  
00061     delayl->getNextComponents()->insert(disposel);  
00062     // save the model into a text file  
00063     model->save("./models/Model_CreateDelayDispose.txt");  
00064     // execute the simulation until completed and show the report  
00065     model->getSimulation()->start();  
00066     genesys->~Simulator();  
00067     return 0;  
00068 },
```

References [Util::componentDetailed](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [ConnectionManager::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [ModelManager::newModel\(\)](#), [Model::save\(\)](#), [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [SourceModelComponent::setTimeBetweenCreate\(\)](#), [TraceManager::setTraceLevel\(\)](#), [ModelSimulation::start\(\)](#), and [Simulator::~Simulator\(\)](#).

Here is the call graph for this function:



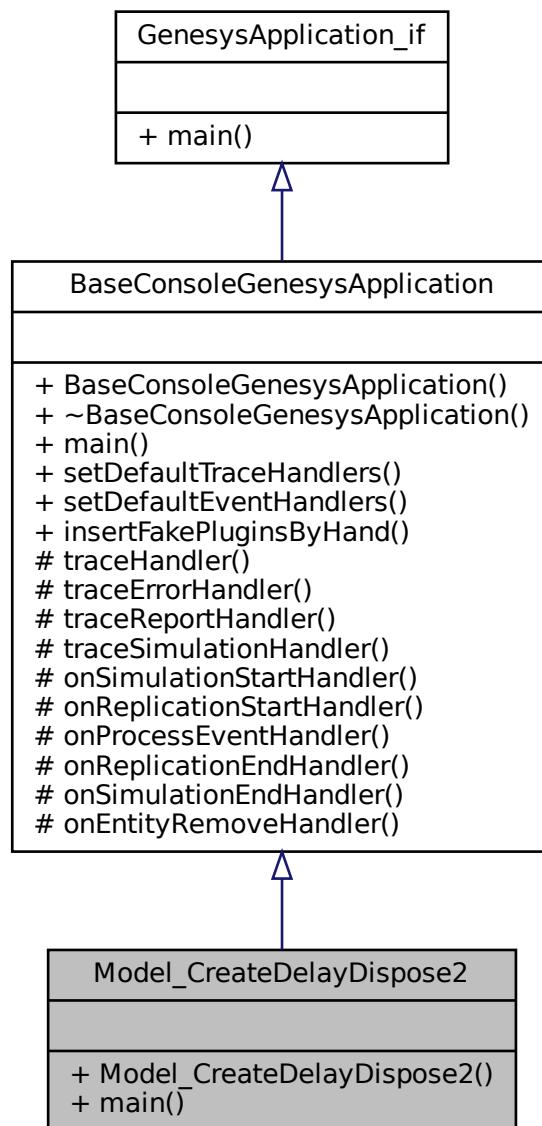
The documentation for this class was generated from the following files:

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

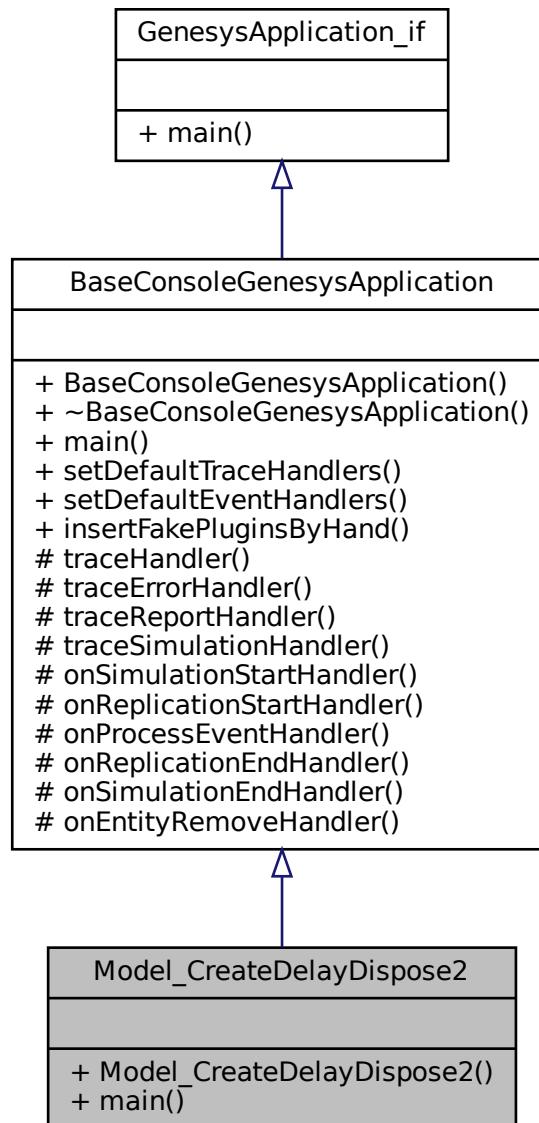
8.66 Model_CreateDelayDispose2 Class Reference

```
#include <Model_CreateDelayDispose2.h>
```

Inheritance diagram for Model_CreateDelayDispose2:



Collaboration diagram for Model_CreateDelayDispose2:



Public Member Functions

- [Model_CreateDelayDispose2 \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

8.66.1 Detailed Description

Definition at line 19 of file [Model_CreateDelayDispose2.h](#).

8.66.2 Constructor & Destructor Documentation

8.66.2.1 Model_CreateDelayDispose2() Model_CreateDelayDispose2::Model_CreateDelayDispose2 ()

Definition at line 30 of file [Model_CreateDelayDispose2.cpp](#).

```
00030 }  
00031 }
```

8.66.3 Member Function Documentation

8.66.3.1 main() int Model_CreateDelayDispose2::main (

```
    int argc,  
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

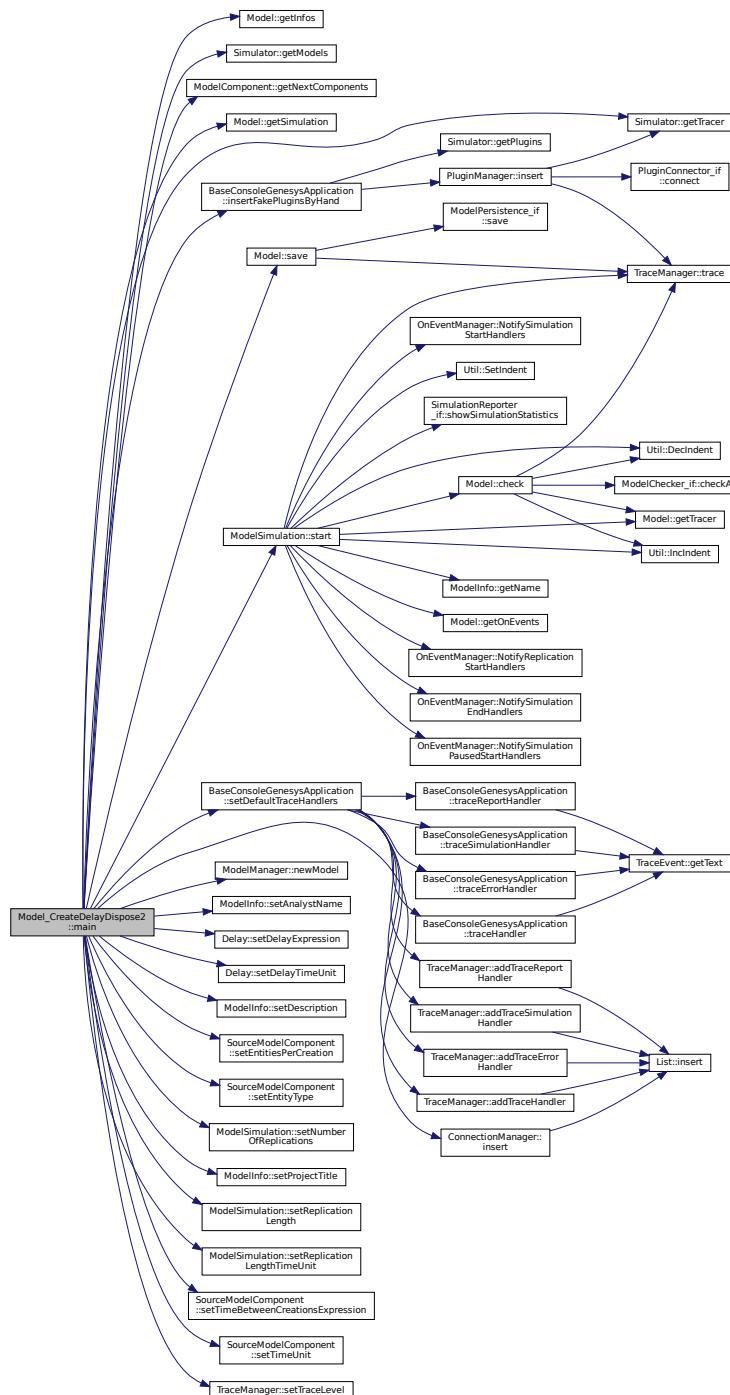
Implements [BaseConsoleGenesysApplication](#).

Definition at line 36 of file [Model_CreateDelayDispose2.cpp](#).

```
00036 {  
00037     Simulator* genesys = new Simulator();  
00038     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of  
00039     // tracing  
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::componentArrival);  
00041     // Handle traces and simulation events to output them  
00042     this->setDefaultTraceHandlers(genesys->getTracer());  
00043     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet  
00044     this->insertFakePluginsByHand(genesys);  
00045     Model* model = genesys->getModels()->newModel();  
00046     // build the simulation model  
00047     // set general info about the model  
00048     ModelInfo* infos = model->getInfos();  
00049     infos->setAnalystName("Your name");  
00050     infos->setProjectTitle("The title of the project");  
00051     infos->setDescription("This simulation model tests one of the most basic models possible.");  
00052     ModelSimulation* sim = model->getSimulation();  
00053     sim->setReplicationLength(30);  
00054     sim->setReplicationLengthTimeUnit(Util::TimeUnit::minute); // each replication will last 30  
00055     minutes (simulated time)  
00056     sim->setNumberOfReplications(3); // replicates the simulation 3 times  
00057     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows  
00058     // Entity Type  
00059     EntityType* entityType1 = new EntityType(model, "Type_of_Representative_Entity");  
00060     // create a ModelComponent of type Create, used to insert entities into the model  
00061     Create* create1 = new Create(model);  
00062     create1->setEntityType(entityType1);  
00063     create1-> setTimeBetweenCreationsExpression("Expo(2)");  
00064     create1->setTimeUnit(Util::TimeUnit::minute);  
00065     create1->setEntitiesPerCreation(1);  
00066     // create a ModelComponent of type Delay, used to represent a time delay  
00067     Delay* delay1 = new Delay(model);  
00068     delay1->setDelayExpression("NORM(1,0.2)");  
00069     delay1->setDelayTimeUnit(Util::TimeUnit::minute);  
00070     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model  
00071     Dispose* dispose1 = new Dispose(model);  
00072     // connect model components to create a "workflow"  
00073     create1->getNextComponents()->insert(delay1);  
00074     delay1->getNextComponents()->insert(dispose1);  
00075     // save the model into a text file  
00076     model->save("./models/Model_CreateDelayDispose2.txt");  
00077     // execute the simulation  
00078     sim->start();  
00079 }
```

References [Util::componentArrival](#), [Model::getInfos\(\)](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [ConnectionManager::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [Util::minute](#), [ModelManager::newModel\(\)](#), [Model::save\(\)](#), [ModelInfo::setAnalystName\(\)](#), [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#), [Delay::setDelayExpression\(\)](#), [Delay::setDelayTimeUnit\(\)](#), [ModelInfo::setDescription\(\)](#), [SourceModelComponent::setEntitiesPerCreation\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [ModelSimulation::setNumberOfReplications\(\)](#), [ModelInfo::setProjectTitle\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [ModelSimulation::setReplicationLengthTimeUnit\(\)](#), [SourceModelComponent::setTimeBetweenCreationsExpression\(\)](#), [SourceModelComponent::setTimeUnit\(\)](#), [TraceManager::setTraceLevel\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the call graph for this function:



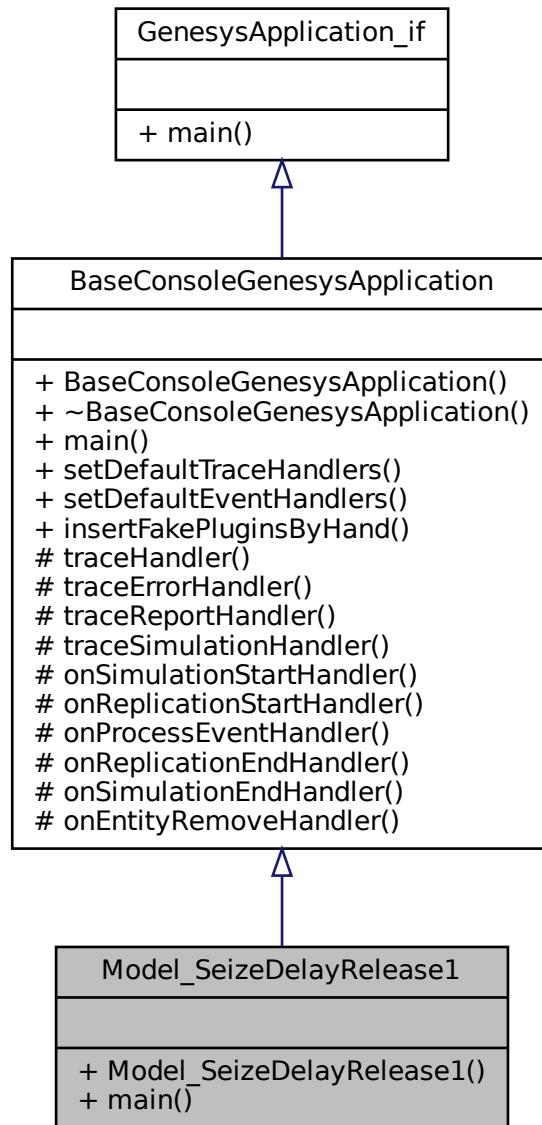
The documentation for this class was generated from the following files:

- [Model_CreteDelayDispose2.h](#)
- [Model_CreteDelayDispose2.cpp](#)

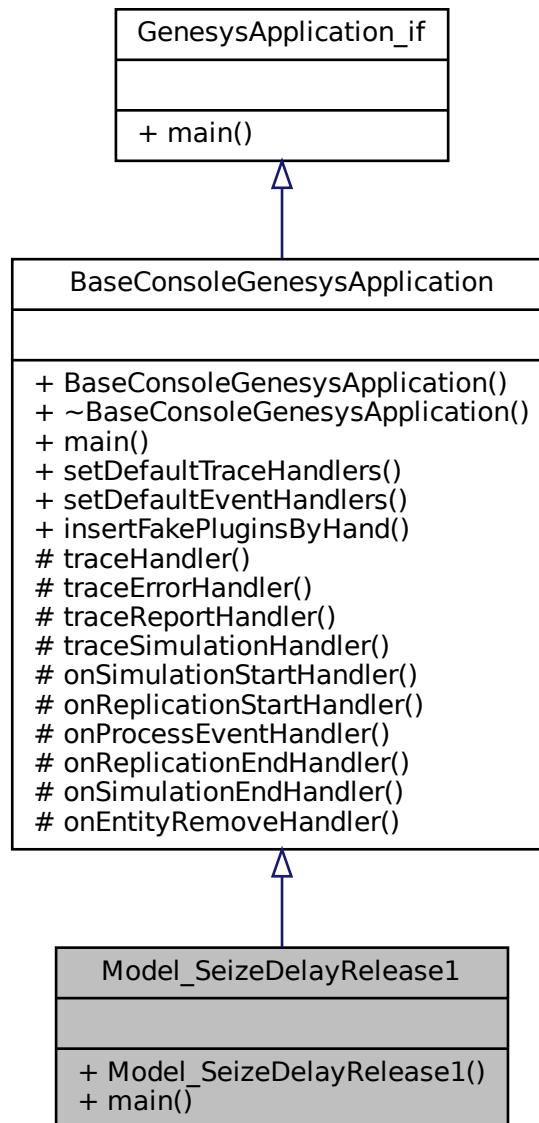
8.67 Model_SeizeDelayRelease1 Class Reference

```
#include <Model_SeizeDelayRelease1.h>
```

Inheritance diagram for Model_SeizeDelayRelease1:



Collaboration diagram for Model_SeizeDelayRelease1:



Public Member Functions

- [Model_SeizeDelayRelease1 \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

8.67.1 Detailed Description

Definition at line 19 of file [Model_SeizeDelayRelease1.h](#).

8.67.2 Constructor & Destructor Documentation

8.67.2.1 Model_SeizeDelayRelease1() Model_SeizeDelayRelease1::Model_SeizeDelayRelease1 ()

Definition at line 31 of file [Model_SeizeDelayRelease1.cpp](#).

```
00031 {  
00032 }
```

8.67.3 Member Function Documentation

8.67.3.1 main() int Model_SeizeDelayRelease1::main (

```
    int argc,  
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

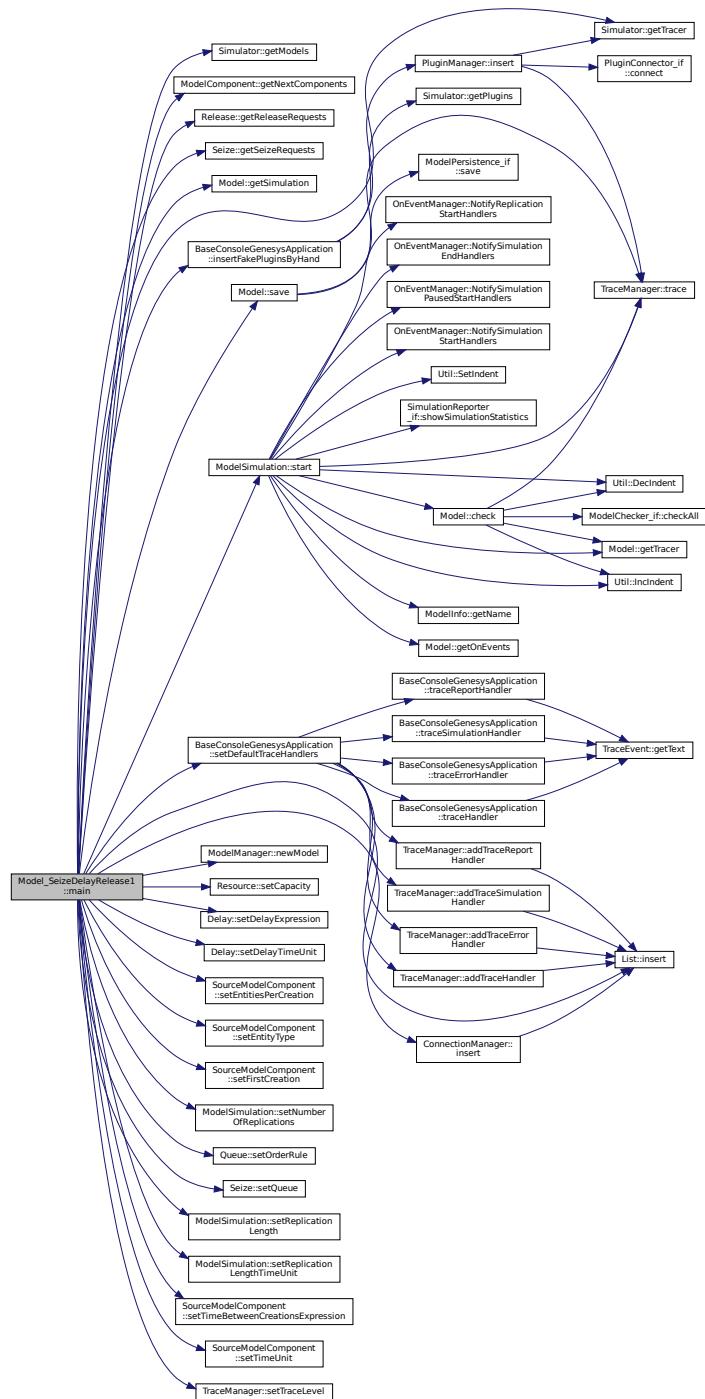
Definition at line 38 of file [Model_SeizeDelayRelease1.cpp](#).

```
00038 {  
00039     Simulator* genesys = new Simulator();  
00040     // Handle traces and simulation events to output them  
00041     this->setDefaultTraceHandlers(genesys->getTracer());  
00042     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);  
00043     // insert plugins  
00044     this->insertFakePluginsByHand(genesys);  
00045     Model* model = genesys->getModels()->newModel();  
00046     // build the simulation model  
00047     // set model infos  
00048     ModelSimulation* sim = model->getSimulation();  
00049     sim->setReplicationLength(3600);  
00050     sim->setReplicationLengthTimeUnit(Util::TimeUnit::second);  
00051     sim->setNumberOfReplications(30);  
00052     //  
00053     EntityType* customer = new EntityType(model, "Customer");  
00054     //  
00055     Create* createl = new Create(model);  
00056     createl->setEntityType(customer);  
00057     createl->setTimeBetweenCreationsExpression("expo(20)");  
00058     createl->setTimeUnit(Util::TimeUnit::second);  
00059     createl->setEntitiesPerCreation(1);  
00060     createl->setFirstCreation(0.0);  
00061     //  
00062     Resource* machine1 = new Resource(model, "Machine_1");  
00063     machine1->setCapacity(1);  
00064     //  
00065     Queue* queueSeize1 = new Queue(model, "Seize_1.Queue");  
00066     queueSeize1->setOrderRule(Queue::OrderRule::FIFO);  
00067     //  
00068     Seize* seize1 = new Seize(model);  
00069     //seize1->setResource(machine1);  
00070     //seize1->setQuantity("1");  
00071     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine1, "1"));  
00072     seize1->setQueue(queueSeize1);  
00073     //  
00074     Delay* delay1 = new Delay(model);  
00075     delay1->setDelayExpression("unif(15,30)");  
00076     delay1->setDelayTimeUnit(Util::TimeUnit::second);  
00077     //  
00078     Release* release1 = new Release(model);  
00079     //release1->setResource(machine1);  
00080     //release1->setQuantity("1");  
00081     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine1, "1"));  
00082     //  
00083     Dispose* dispose1 = new Dispose(model);
```

```
00084 // connect model components to create a "workflow"
00085 create1->getNextComponents()->insert(seize1);
00086 seize1->getNextComponents()->insert(delay1);
00087 delay1->getNextComponents()->insert(release1);
00088 release1->getNextComponents()->insert(dispose1);
00089 // save the model into a text file
00090 model->save("./models/Model_SeizeDelayRelease1.txt");
00091 // execute the simulation
00092 sim->start();
00093
00094 return 0;
00095 };
```

References [Util::everythingMostDetailed](#), [Queue::FIFO](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Release::getReleaseRequests\(\)](#), [Seize::getSeizeRequests\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [ConnectionManager::insert\(\)](#), [List< T >::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [ModelManager::newModel\(\)](#), [Model::save\(\)](#), [Util::second](#), [Resource::setCapacity\(\)](#), [BaseConsoleGenesysApplication::setDefaultTrace\(\)](#), [Delay::setDelayExpression\(\)](#), [Delay::setDelayTimeUnit\(\)](#), [SourceModelComponent::setEntitiesPerCreation\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [SourceModelComponent::setFirstCreation\(\)](#), [ModelSimulation::setNumberOfReplications\(\)](#), [Queue::setOrderRule\(\)](#), [Seize::setQueue\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [ModelSimulation::setReplicationLengthTimeUnit\(\)](#), [SourceModelComponent::setTimeBetweenCreationsExpression\(\)](#), [SourceModelComponent::setTimeUnit\(\)](#), [TraceManager::setTraceLevel\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the call graph for this function:



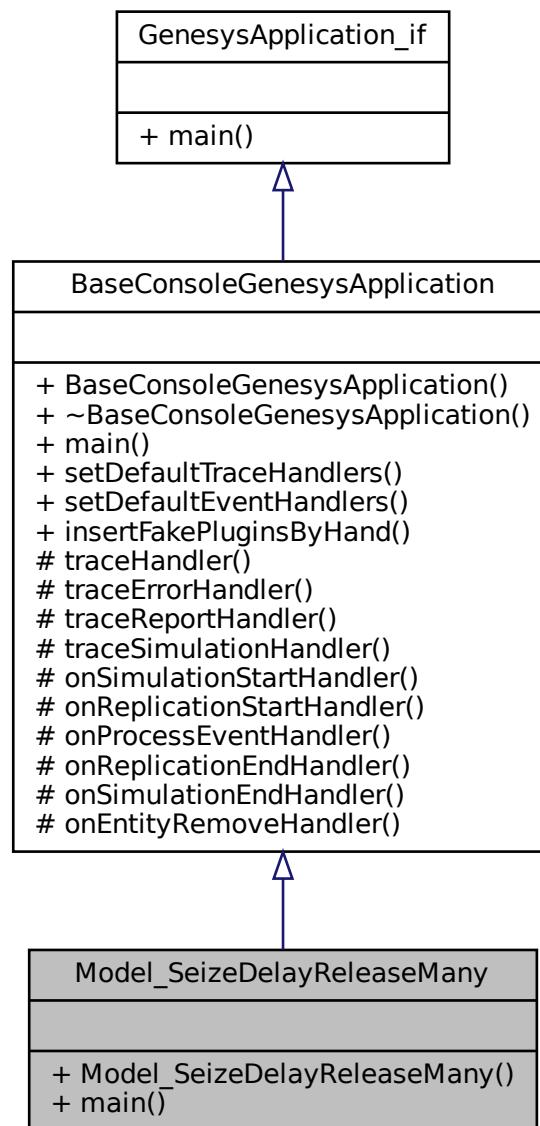
The documentation for this class was generated from the following files:

- [Model_SeizeDelayRelease1.h](#)
- [Model_SeizeDelayRelease1.cpp](#)

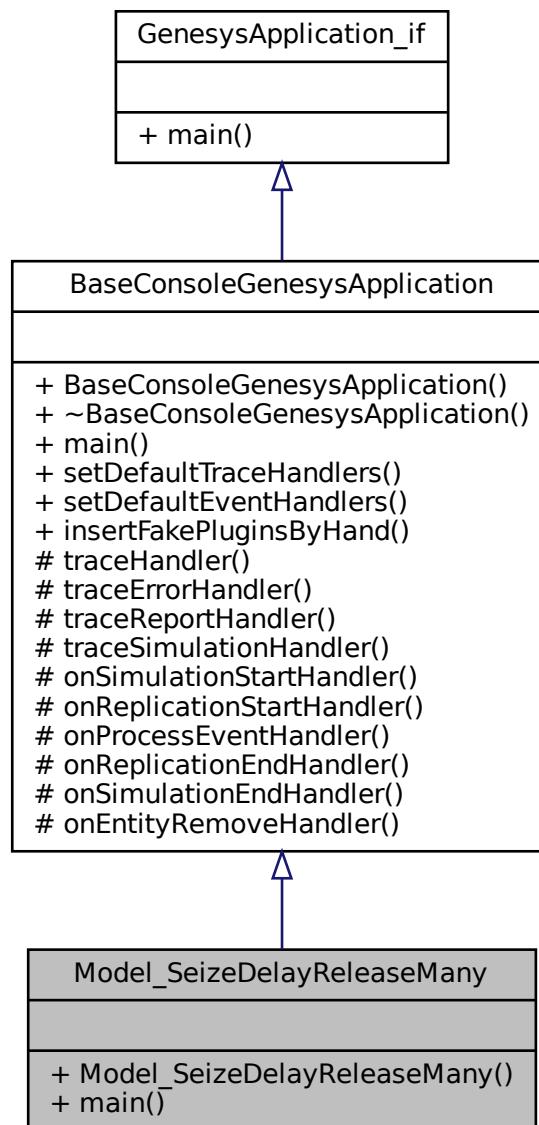
8.68 Model_SeizeDelayReleaseMany Class Reference

```
#include <Model_SeizeDelayReleaseMany.h>
```

Inheritance diagram for Model_SeizeDelayReleaseMany:



Collaboration diagram for Model_SeizeDelayReleaseMany:



Public Member Functions

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

Additional Inherited Members

8.68.1 Detailed Description

Definition at line 19 of file [Model_SeizeDelayReleaseMany.h](#).

8.68.2 Constructor & Destructor Documentation

8.68.2.1 Model_SeizeDelayReleaseMany() Model_SeizeDelayReleaseMany::Model_SeizeDelayReleaseMany()

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

```
00031 {  
00032 }
```

8.68.3 Member Function Documentation

8.68.3.1 main() int Model_SeizeDelayReleaseMany::main (int argc, char ** argv) [virtual]

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

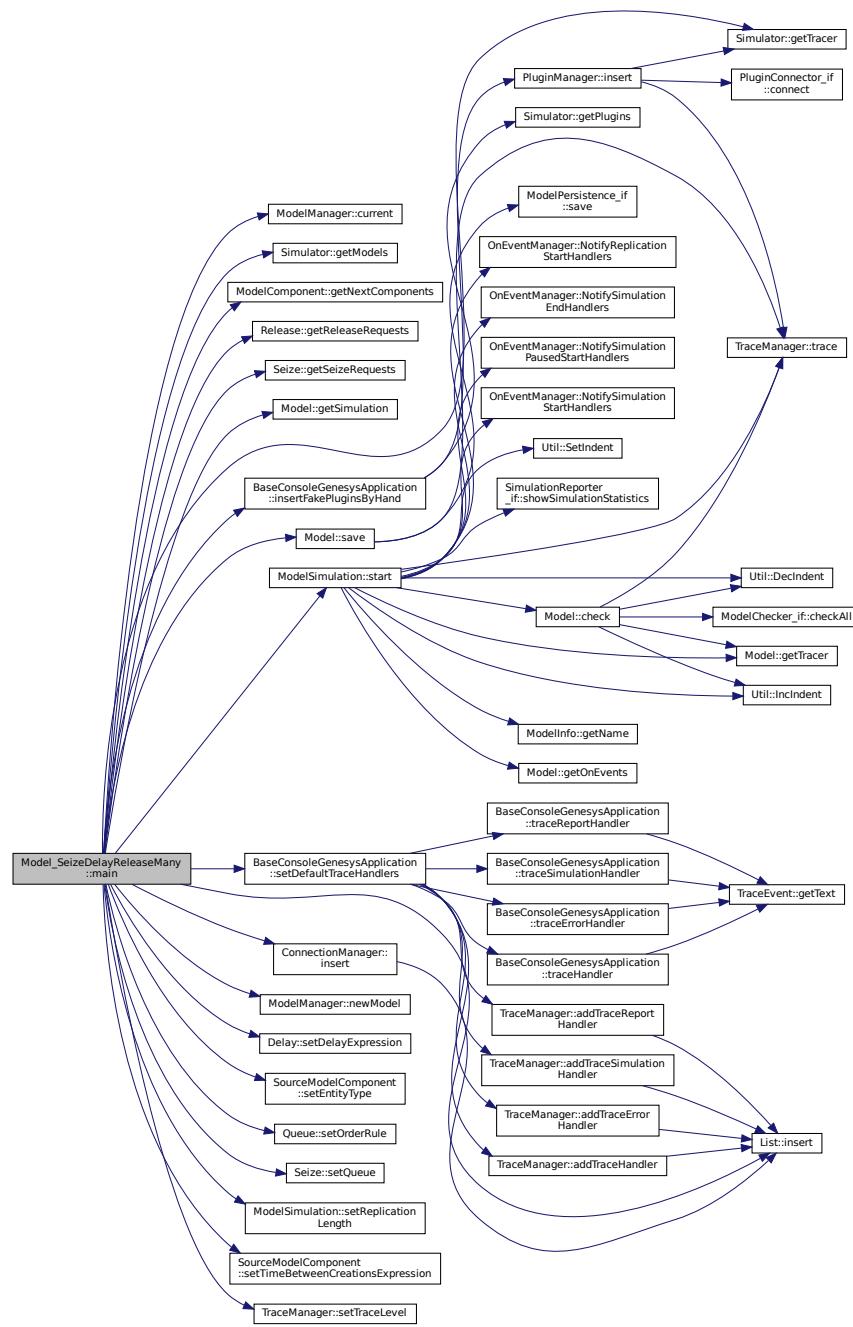
Definition at line 38 of file [Model_SeizeDelayReleaseMany.cpp](#).

```
00038 {  
00039     Simulator* genesys = new Simulator();  
00040     this->setDefaultTraceHandlers(genesys->getTracer());  
00041     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);  
00042     this->insertFakePluginsByHand(genesys);  
00043     Model* m = genesys->getModels()->newModel();  
00044     m->getSimulation()->setReplicationLength(60);  
00045     EntityType* customer = new EntityType(m);  
00046     Create* create1 = new Create(m);  
00047     create1->setEntityType(customer);  
00048     create1->setTimeBetweenCreationsExpression("unif(1,2)");  
00049     Resource* machine1 = new Resource(m);  
00050     Resource* machine2 = new Resource(m);  
00051     Resource* machine3 = new Resource(m);  
00052     Resource* machine4 = new Resource(m);  
00053     Queue* queueSeize1 = new Queue(m);  
00054     queueSeize1->setOrderRule(Queue::OrderRule::FIFO);  
00055     Seize* seize1 = new Seize(m);  
00056     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine1));  
00057     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine2));  
00058     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine3));  
00059     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine4));  
00060     seize1->setQueue(queueSeize1);  
00061     Delay* delay1 = new Delay(m);  
00062     delay1->setDelayExpression("unif(1,2)");  
00063     Release* release1 = new Release(m);  
00064     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine1));  
00065     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine2));  
00066     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine3));  
00067     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine4));  
00068     Dispose* dispose1 = new Dispose();  
00069     // connect model components to create a "workflow"  
00070     create1->getNextComponents()->insert(seize1);  
00071     seize1->getNextComponents()->insert(delay1);  
00072     delay1->getNextComponents()->insert(release1);  
00073     release1->getNextComponents()->insert(dispose1);  
00074     // save the model into a text file  
00075     m->save("./models/Model_SeizeDelayReleaseMany.txt");  
00076     genesys->getModels()->current()->getSimulation()->start();  
00077     return 0;  
00078 };
```

References [ModelManager::current\(\)](#), [Util::everythingMostDetailed](#), [Queue::FIFO](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Release::getReleaseRequests\(\)](#), [Seize::getSeizeRequests\(\)](#), [Model::getSimulation\(\)](#),

`Simulator::getTracer()`, `ConnectionManager::insert()`, `List< T >::insert()`, `BaseConsoleGenesysApplication::insertFakePluginsByHand()`, `ModelManager::newModel()`, `Model::save()`, `BaseConsoleGenesysApplication::setDefaultTraceHandlers()`, `Delay::setDelayExpression()`, `SourceModelComponent::setEntityType()`, `Queue::setOrderRule()`, `Seize::setQueue()`, `ModelSimulation::setReplicationLength()`, `SourceModelComponent::setTimeBetweenCreationsExpression()`, `TraceManager::setTraceLevel()`, and `ModelSimulation::start()`.

Here is the call graph for this function:



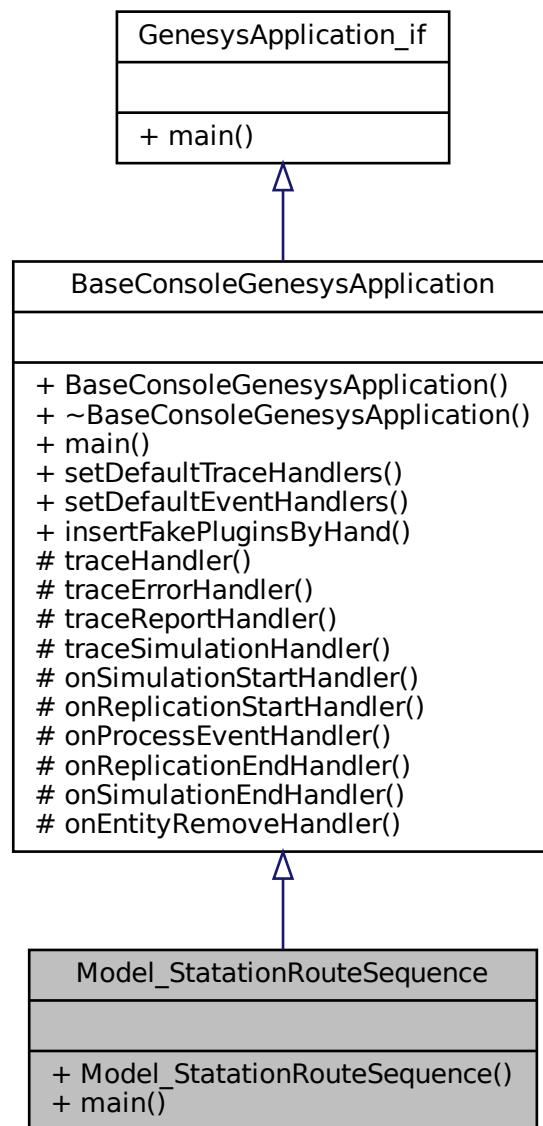
The documentation for this class was generated from the following files:

- `Model_SeizeDelayReleaseMany.h`
- `Model_SeizeDelayReleaseMany.cpp`

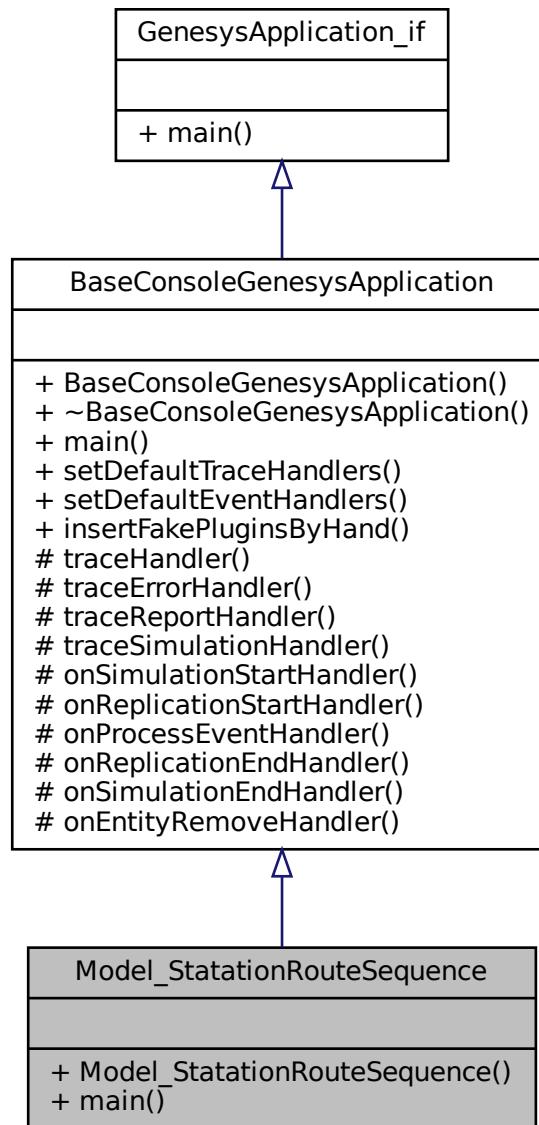
8.69 Model_StationRouteSequence Class Reference

```
#include <Model_StationRouteSequence.h>
```

Inheritance diagram for Model_StationRouteSequence:



Collaboration diagram for Model_StationRouteSequence:



Public Member Functions

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

Additional Inherited Members

8.69.1 Detailed Description

Definition at line 19 of file [Model_StationRouteSequence.h](#).

8.69.2 Constructor & Destructor Documentation

8.69.2.1 Model_StationRouteSequence() Model_StationRouteSequence::Model_StationRouteSequence ()

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

```
00028     {
00029 }
```

8.69.3 Member Function Documentation

8.69.3.1 main() int Model_StationRouteSequence::main (

```
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

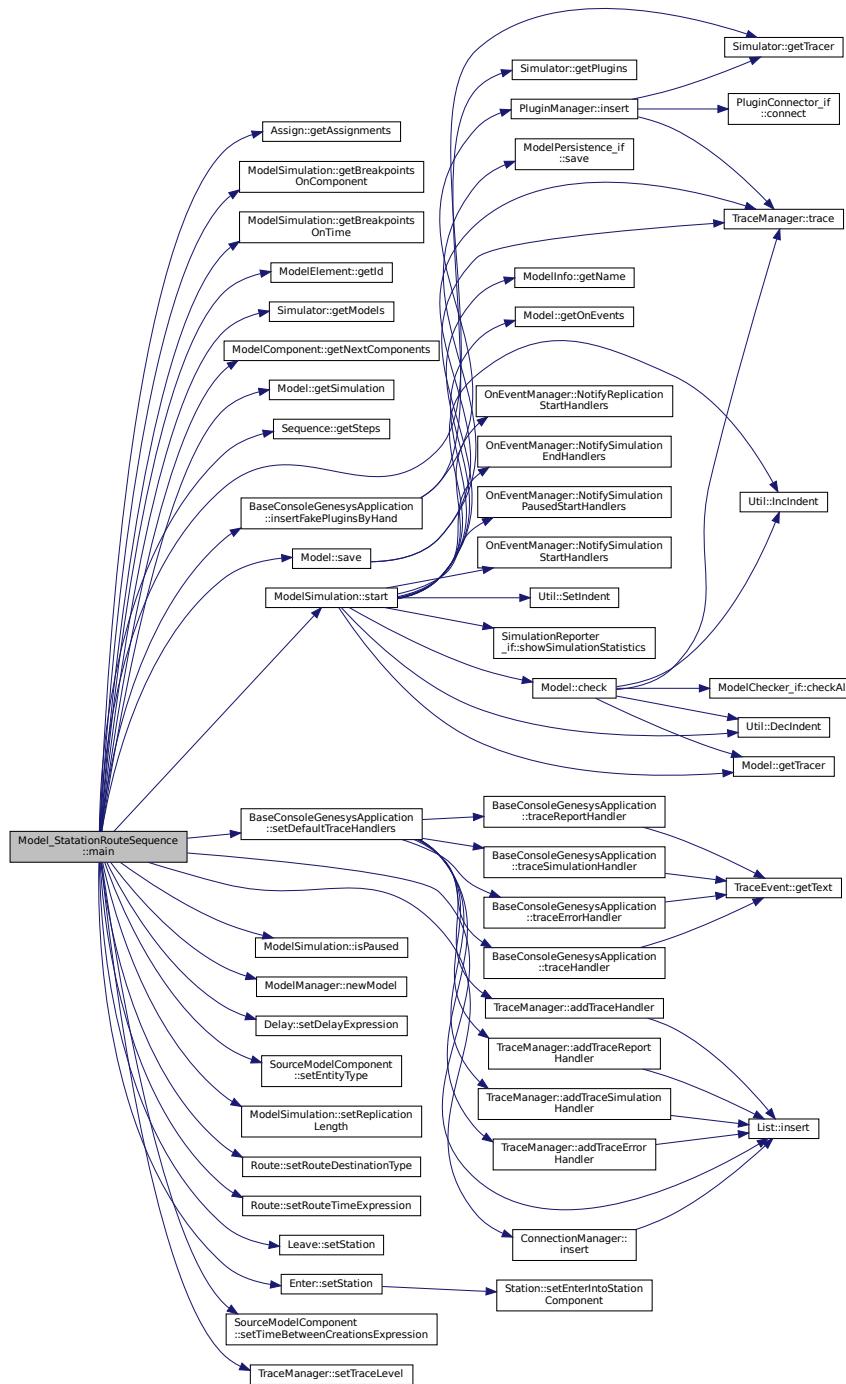
Definition at line 31 of file [Model_StationRouteSequence.cpp](#).

```
00031     {
00032         Simulator* genesys = new Simulator();
00033         this->insertFakePluginsByHand(genesys);
00034         this->setDefaultTraceHandlers(genesys->getTracer());
00035         genesys->getTracer()->setTraceLevel(Util::TraceLevel::modelSimulationEvent);
00036
00037         Model* m = genesys->getModels()->newModel();
00038         m->getSimulation()->setReplicationLength(60);
00039
00040         Create* c1 = new Create(m);
00041         c1->setEntityType(new EntityType(m));
00042         c1-> setTimeBetweenCreationsExpression("10");
00043         Station* s1 = new Station(m);
00044         Route* r0 = new Route(m);
00045         r0->setRouteDestinationType(Route::DestinationType::BySequence);
00046         // (Route::DestinationType::Station);
00047         //r0->setStation(s1);
00048         r0-> setRouteTimeExpression("0.1");
00049         Enter* e1 = new Enter(m);
00050         e1->setStation(s1);
00051         Delay* d1 = new Delay(m);
00052         Station* s2 = new Station(m);
00053         Leave* l1 = new Leave(m);
00054         l1->setStation(s1);
00055         Route* r1 = new Route(m);
00056         r1->setRouteDestinationType(Route::DestinationType::BySequence);
00057         // (Route::DestinationType::Station);
00058         //r1->setStation(s2);
00059         r1-> setRouteTimeExpression("0.2");
00060         Enter* e2 = new Enter(m);
00061         e2->setStation(s2);
00062         Delay* d2 = new Delay(m);
00063         d2-> setDelayExpression("2");
00064         Leave* l2 = new Leave(m);
00065         l2->setStation(s2);
00066         Station* s3 = new Station(m);
00067         Route* r2 = new Route(m);
00068         r2->setRouteDestinationType(Route::DestinationType::BySequence);
00069         // (Route::DestinationType::Station);
00070         //r2->setStation(s3);
00071         r2-> setRouteTimeExpression("0.3");
00072         Enter* e3 = new Enter(m);
00073         e3->setStation(s3);
00074         Dispose* dp1 = new Dispose(m);
00075
00076         Sequence* seq = new Sequence(m);
00077         seq->getSteps()->insert(new SequenceStep(s1));
00078         seq->getSteps()->insert(new SequenceStep(s2));
00079         seq->getSteps()->insert(new SequenceStep(s1));
```

```
00077     seq->getSteps()->insert(new SequenceStep(s1));
00078     seq->getSteps()->insert(new SequenceStep(s3));
00079
00080     Assign* a1 = new Assign(m);
00081     a1->getAssignments()->insert(new Assign::Assignment("Entity.Sequence",
00082                                     std::to_string(seq->getId())));
00083
00084     c1->getNextComponents()->insert(a1);
00085     a1->getNextComponents()->insert(r0);
00086     e1->getNextComponents()->insert(d1);
00087     d1->getNextComponents()->insert(l1);
00088     l1->getNextComponents()->insert(r1);
00089     e2->getNextComponents()->insert(d2);
00090     d2->getNextComponents()->insert(l2);
00091     l2->getNextComponents()->insert(r2);
00092     e3->getNextComponents()->insert(dp1);
00093
00094     ModelSimulation* sim = m->getSimulation();
00095     sim->getBreakpointsOnComponent()->insert(a1);
00096     sim->getBreakpointsOnComponent()->insert(l2);
00097     sim->getBreakpointsOnTime()->insert(40.0);
00098     sim->getBreakpointsOnTime()->insert(20.0);
00099
00100    m->save("./models/Model_StatationRouteSequence.txt");
00101
00102    do {
00103        sim->start();
00104    } while (sim->isPaused());
00105    return 0;
00106 }
```

References [Route::BySequence](#), [Assign::getAssignments\(\)](#), [ModelSimulation::getBreakpointsOnComponent\(\)](#), [ModelSimulation::getBreakpointsOnTime\(\)](#), [ModelElement::getId\(\)](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Model::getSimulation\(\)](#), [Sequence::getSteps\(\)](#), [Simulator::getTracer\(\)](#), [ConnectionManager::insert\(\)](#), [List< T >::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [ModelSimulation::isPaused\(\)](#), [Util::modelSimulationEvent](#), [ModelManager::newModel\(\)](#), [Model::save\(\)](#), [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#), [Delay::setDelayExpression\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [Route::setRouteDestinationType\(\)](#), [Route::setRouteTimeExpression\(\)](#), [Leave::setStation\(\)](#), [Enter::setStation\(\)](#), [SourceModelComponent::setTimeBetweenCreationsExpression\(\)](#), [TraceManager::setTraceLevel\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the call graph for this function:



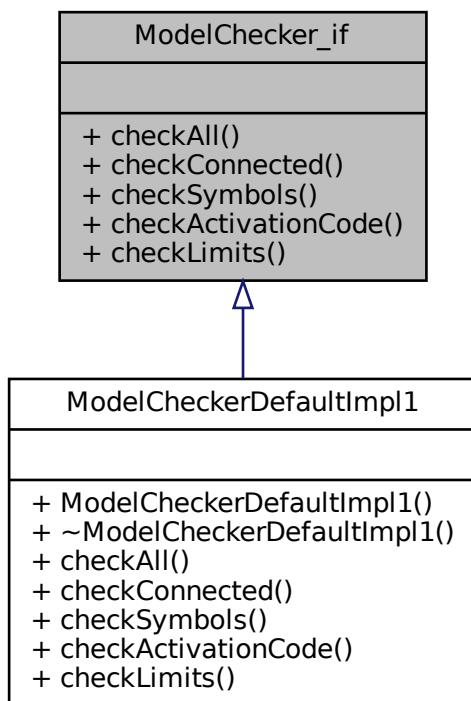
The documentation for this class was generated from the following files:

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

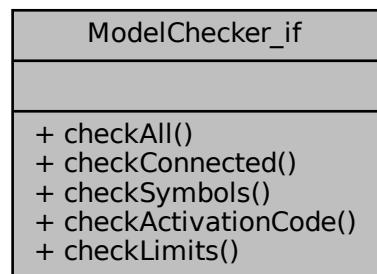
8.70 ModelChecker_if Class Reference

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

Inheritance diagram for ModelChecker_if:



Collaboration diagram for ModelChecker_if:



Public Member Functions

- virtual bool `checkAll ()=0`
- virtual bool `checkConnected ()=0`
- virtual bool `checkSymbols ()=0`
- virtual bool `checkActivationCode ()=0`
- virtual bool `checkLimits ()=0`

8.70.1 Detailed Description

The ModelChecker is responsible for verifying the model consistency, fixing inconsistencies whenever possible
Definition at line 26 of file [ModelChecker_if.h](#).

8.70.2 Member Function Documentation

8.70.2.1 `checkActivationCode()` virtual bool ModelChecker_if::checkActivationCode () [pure virtual]

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

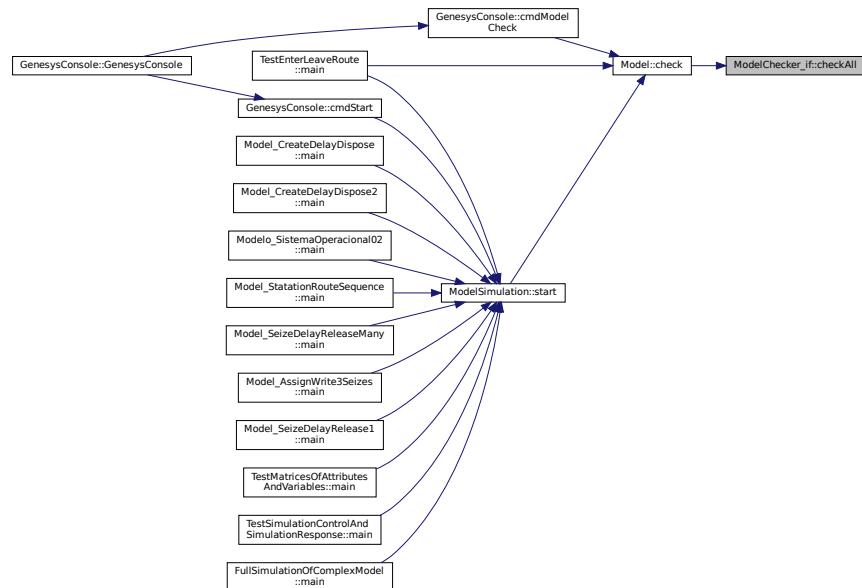
Implemented in [ModelCheckerDefaultImpl1](#).

8.70.2.2 `checkAll()` virtual bool ModelChecker_if::checkAll () [pure virtual]

Implemented in [ModelCheckerDefaultImpl1](#).

Referenced by [Model::check\(\)](#).

Here is the caller graph for this function:



8.70.2.3 `checkConnected()` `virtual bool ModelChecker_if::checkConnected () [pure virtual]`

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

Implemented in [ModelCheckerDefaultImpl1](#).

8.70.2.4 `checkLimits()` `virtual bool ModelChecker_if::checkLimits () [pure virtual]`

Checks if the installed version has acquired a valid activation code for commercial use

Implemented in [ModelCheckerDefaultImpl1](#).

8.70.2.5 `checkSymbols()` `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 [ModelCheckerDefaultImpl1](#).

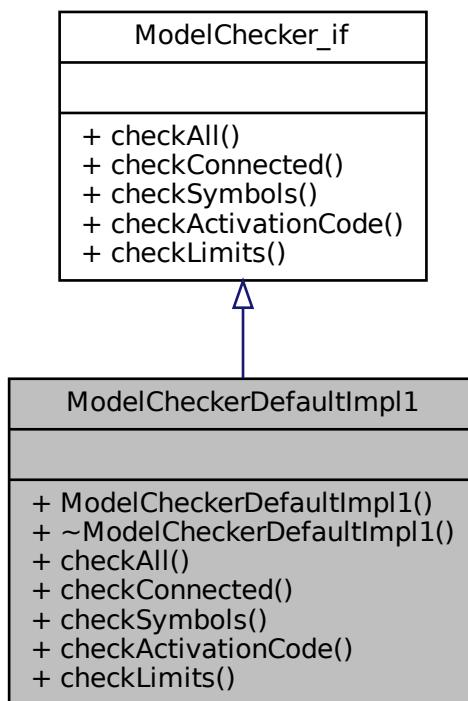
The documentation for this class was generated from the following file:

- [ModelChecker_if.h](#)

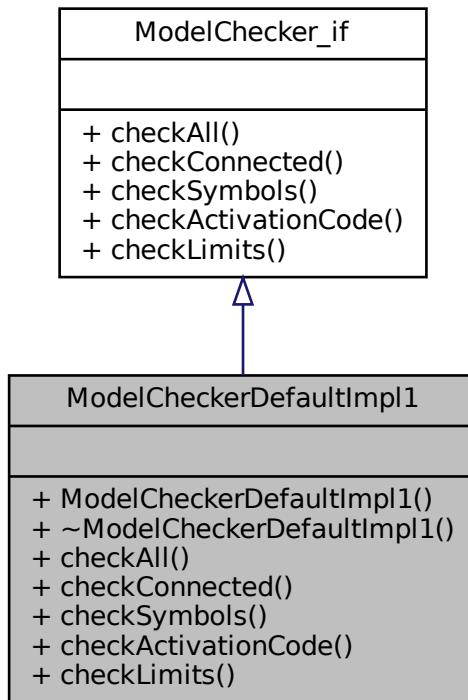
8.71 ModelCheckerDefaultImpl1 Class Reference

```
#include <ModelCheckerDefaultImpl1.h>
```

Inheritance diagram for ModelCheckerDefaultImpl1:



Collaboration diagram for ModelCheckerDefaultImpl1:



Public Member Functions

- `ModelCheckerDefaultImpl1 (Model *model)`
- virtual `~ModelCheckerDefaultImpl1 ()=default`
- virtual bool `checkAll ()`
- virtual bool `checkConnected ()`
- virtual bool `checkSymbols ()`
- virtual bool `checkActivationCode ()`
- virtual bool `checkLimits ()`

8.71.1 Detailed Description

Definition at line 23 of file [ModelCheckerDefaultImpl1.h](#).

8.71.2 Constructor & Destructor Documentation

8.71.2.1 ModelCheckerDefaultImpl1() ModelCheckerDefaultImpl1::ModelCheckerDefaultImpl1 (Model * model)

Definition at line 24 of file [ModelCheckerDefaultImpl1.cpp](#).

```
00024
00025     _model = model;
00026 }
```

8.71.2.2 ~ModelCheckerDefaultImpl1() virtual ModelCheckerDefaultImpl1::~ModelCheckerDefaultImpl1 () [virtual], [default]

8.71.3 Member Function Documentation

8.71.3.1 checkActivationCode() bool ModelCheckerDefaultImpl1::checkActivationCode () [virtual]

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

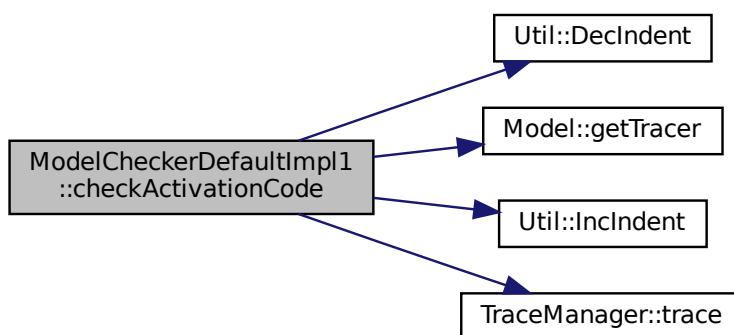
Implements [ModelChecker_if](#).

Definition at line 166 of file [ModelCheckerDefaultImpl1.cpp](#).

```
00166
00167     /* \todo: ++: not implemented yet */
00168     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking activation code");
00169     Util::IncIndent();
00170 {
00171
00172 }
00173     Util::DecIndent();
00174     return true;
00175 }
```

References [Util::DecIndent\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [Util::toolInternal](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.71.3.2 checkAll() `bool ModelCheckerDefaultImpl1::checkAll () [virtual]`

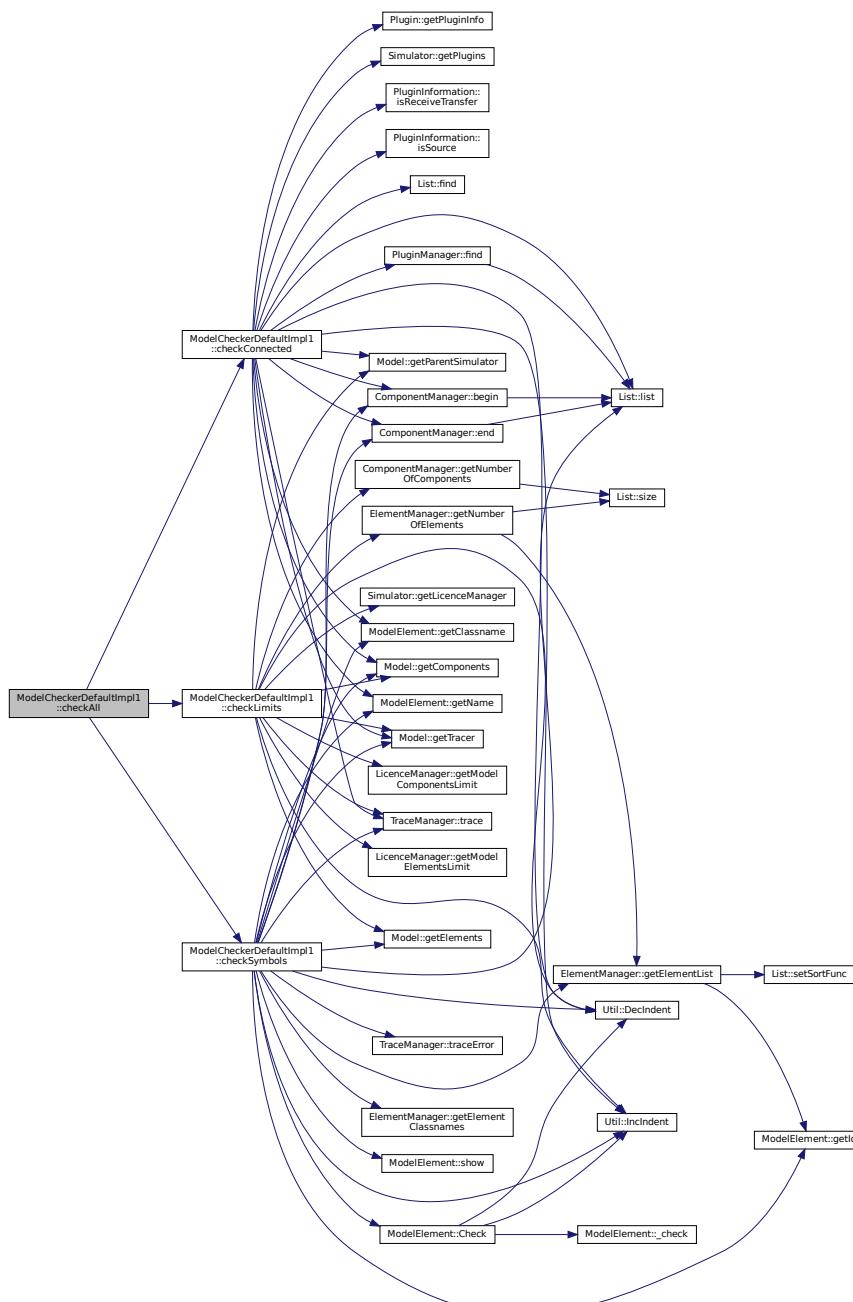
Implements [ModelChecker_if](#).

Definition at line 28 of file [ModelCheckerDefaultImpl1.cpp](#).

```
00028     bool res = true;
00029     res &= checkSymbols\(\);
00030     //res &= checkAndAddInternalLiterals\(\);
00031     //res &= checkActivationCode\(\);
00032     res &= checkLimits\(\);
00033     res &= checkConnected\(\);
00034
00035     return res;
00036 }
```

References [checkConnected\(\)](#), [checkLimits\(\)](#), and [checkSymbols\(\)](#).

Here is the call graph for this function:



8.71.3.3 checkConnected() bool ModelCheckerDefaultImpl1::checkConnected () [virtual]

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

Implements [ModelChecker_if](#).

Definition at line 76 of file [ModelCheckerDefaultImpl1.cpp](#).

```

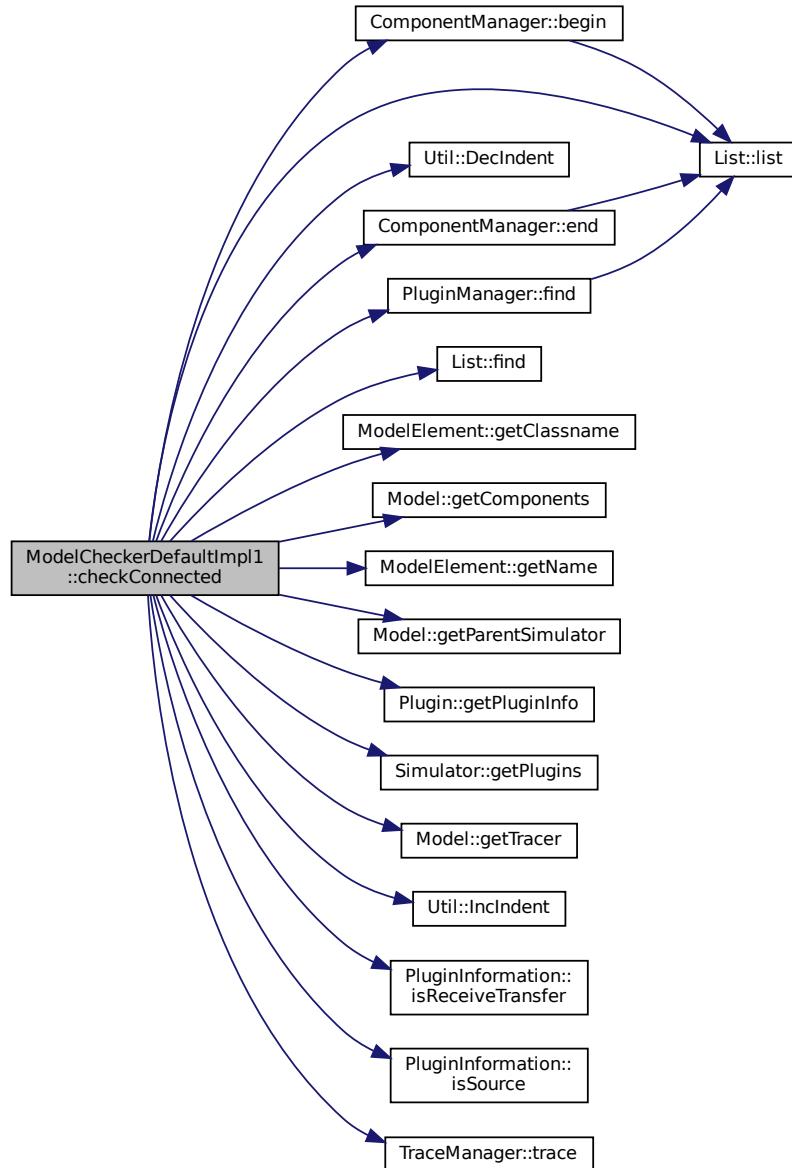
00076                                     {
00077     /* \todo: ++: not implemented yet */
00078     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking connected");
00079     bool resultAll = true;
00080     PluginManager* pluginManager = this->_model->getParentSimulator()->getPlugins();
00081     Plugin* plugin;
00082     Util::IncIndent();
00083     {
00084         List<ModelComponent*>* visited = new List<ModelComponent*>();
00085         List<ModelComponent*>* unconnected = new List<ModelComponent*>();
00086         ModelComponent* comp;
00087         for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00088             comp = (*it);
00089             plugin = pluginManager->find(comp->getClassname());
00090             assert(plugin != nullptr);
00091             if (plugin->getPluginInfo()->isSource() || plugin->getPluginInfo()->isReceiveTransfer()) {
00092                 // (dynamic_cast<SourceModelComponent*> (comp) != nullptr) {
00093                     // it is a source component OR it can receive entities from transfer
00094                     bool drenoFound = false;
00095                     _recursiveConnectedTo(pluginManager, comp, visited, unconnected, &drenoFound);
00096                 }
00097                 // check if any component remains unconnected
00098                 for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00099                     comp = (*it);
00100                     if (visited->find(comp) == visited->list()->end()) { //not found
00101                         resultAll = false;
00102                         _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Component \""
00103                                         + comp->getName() + "\" is unconnected.");
00104                     }
00105                 }
00106             }
00107             Util::DecIndent();
00108         return resultAll;
00109     }

```

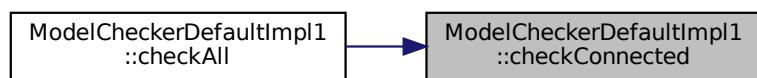
References [ComponentManager::begin\(\)](#), [Util::DecIndent\(\)](#), [ComponentManager::end\(\)](#), [Util::errorFatal](#), [PluginManager::find\(\)](#), [List< T >::find\(\)](#), [ModelElement::getClassname\(\)](#), [Model::getComponents\(\)](#), [ModelElement::getName\(\)](#), [Model::getParentSimulator\(\)](#), [Plugin::getPluginInfo\(\)](#), [Simulator::getPlugins\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [PluginInformation::isReceiveTransfer\(\)](#), [PluginInformation::isSource\(\)](#), [List< T >::list\(\)](#), [Util::toolInternal](#), and [TraceManager::trace\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.3.4 `checkLimits()` `bool ModelCheckerDefaultImpl1::checkLimits() [virtual]`

Checks if the installed version has acquired a valid activation code for commercial use

Implements [ModelChecker_if](#).

Definition at line 177 of file [ModelCheckerDefaultImpl1.cpp](#).

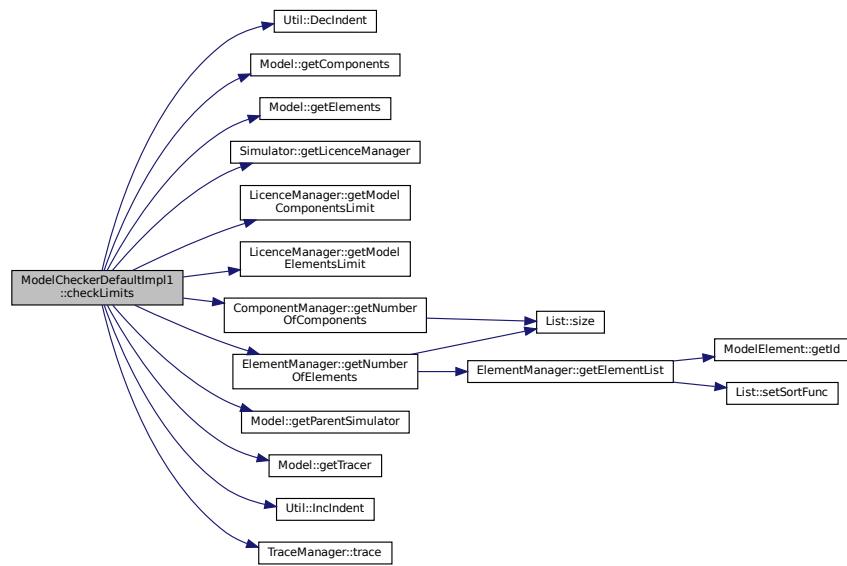
```

00177     {
00178         bool res = true;
00179         std::string text;
00180         unsigned int value, limit;
00181         LicenceManager *licence = _model->getParentSimulator()->getLicenceManager();
00182         _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking model limits");
00183         Util::IncIndent();
00184     {
00185         value = _model->getComponents()->getNumberOfComponents();
00186         limit = licence->getModelComponentsLimit();
00187         res &= value <= limit;
00188         _model->getTracer()->trace("Model has " + std::to_string(value) + "/" + std::to_string(limit)
+ " components");
00189         if (!res) {
00190             text = "Model has " + std::to_string(_model->getComponents()->getNumberOfComponents()) + "
components, exceeding the limit of " + std::to_string(licence->getModelComponentsLimit()) + "
components imposed by the current activation code";
00191             ///_model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00192         } else {
00193             value = _model->getElements()->getNumberOfElements();
00194             limit = licence->getModelElementsLimit();
00195             res &= value <= limit;
00196             _model->getTracer()->trace("Model has " + std::to_string(value) + "/" +
std::to_string(limit) + " elements");
00197             if (!res) {
00198                 text = "Model has " + std::to_string(_model->getElements()->getNumberOfElements()) + "
elements, exceeding the limit of " + std::to_string(licence->getModelElementsLimit()) + "
elements imposed by the current activation code";
00199                 ///_model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00200             }
00201         }
00202         if (!res) {
00203             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error: Checking has failed with
message '" + text + "'");
00204         }
00205     }
00206     Util::DecIndent();
00207     return res;
00208 }
```

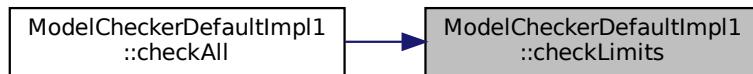
References [Util::DecIndent\(\)](#), [Util::errorFatal](#), [Model::getComponents\(\)](#), [Model::getElements\(\)](#), [Simulator::getLicenceManager\(\)](#), [LicenceManager::getModelComponentsLimit\(\)](#), [LicenceManager::getModelElementsLimit\(\)](#), [ComponentManager::getNumberOfComponents\(\)](#), [ElementManager::getNumberOfElements\(\)](#), [Model::getParentSimulator\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [Util::toolInternal](#), and [TraceManager::trace\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.3.5 checkSymbols() bool ModelCheckerDefaultImpl1::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](#).

Definition at line 111 of file [ModelCheckerDefaultImpl1.cpp](#).

```

00111
00112     bool res = true;
00113     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking symbols");
00114     Util::IncIndent();
00115     {
00116         // check components
00117         _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Components:");
00118         Util::IncIndent();
00119         {
00120             //List<ModelComponent*>* components = _model->getComponents();
00121             for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00122                 res &= (*it)->Check((*it));
00123             }
  
```

```

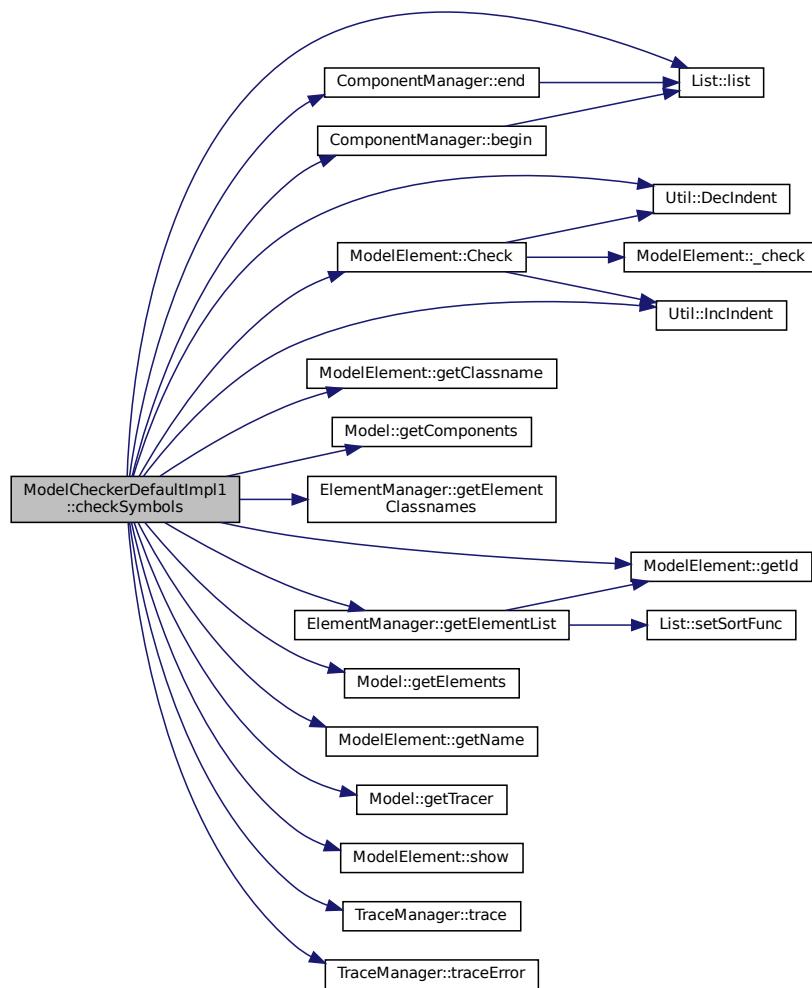
00124         }
00125         Util::DecIndent();
00126
00127         // check elements
00128         _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Elements:");
00129         Util::IncIndent();
00130     {
00131         std::string elementType;
00132         bool result;
00133         ModelElement* element;
00134         std::string* errorMessage = new std::string();
00135         std::list<std::string>* elementTypes = _model->getElements()->elementClassnames();
00136         for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end(); typeIt++) {
00137             elementType = (*typeIt);
00138             List<ModelElement*>* elements = _model->getElements()->elementList(elementType);
00139             for (std::list<ModelElement*>::iterator it = elements->list()->begin(); it != elements->list()->end(); it++) {
00140                 element = (*it);
00141                 // copied from modelComponent. It is not inside the ModelElement::Check because
00142                 // ModelElement has no access to Model to call Tracer
00143                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Checking " +
00144                 element->classname() + ": \" " + element->getName() + "\" (id " + std::to_string(element->getId()) +
00145                 ")"); //std::to_string(component->_id));
00146                 Util::IncIndent();
00147                 {
00148                     try {
00149                         result = element->Check((*it), errorMessage);
00150                         res &= result;
00151                         if (!result) {
00152                             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error:
00153                             Checking has failed with message '" + *errorMessage + "'");
00154                         }
00155                     } catch (const std::exception& e) {
00156                         _model->getTracer()->traceError(e, "Error verifying component " +
00157                         element->show());
00158                     }
00159                 Util::DecIndent();
00160             }
00161             Util::DecIndent();
00162
00163         return res;
00164     }

```

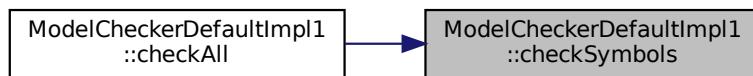
References ComponentManager::begin(), ModelElement::Check(), Util::DecIndent(), ComponentManager::end(), Util::errorFatal, ModelElement::getClassName(), Model::getComponents(), ElementManager::getElementClassnames(), ElementManager::getElementList(), Model::getElements(), ModelElement::getId(), ModelElement::getName(), Model::getTracer(), Util::IncIndent(), List< T >::list(), ModelElement::show(), Util::toolDetailed, Util::toolInternal, TraceManager::trace(), and TraceManager::traceError().

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

Here is the call graph for this function:



Here is the caller graph for this function:



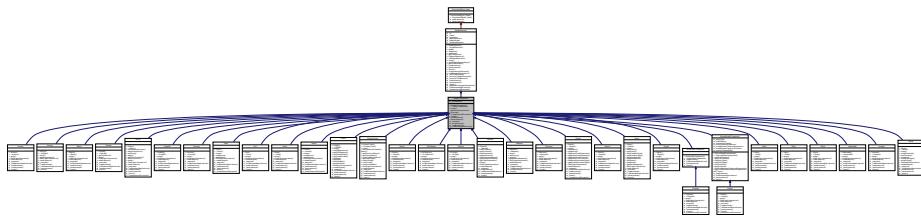
The documentation for this class was generated from the following files:

- [ModelCheckerDefaultImpl1.h](#)
- [ModelCheckerDefaultImpl1.cpp](#)

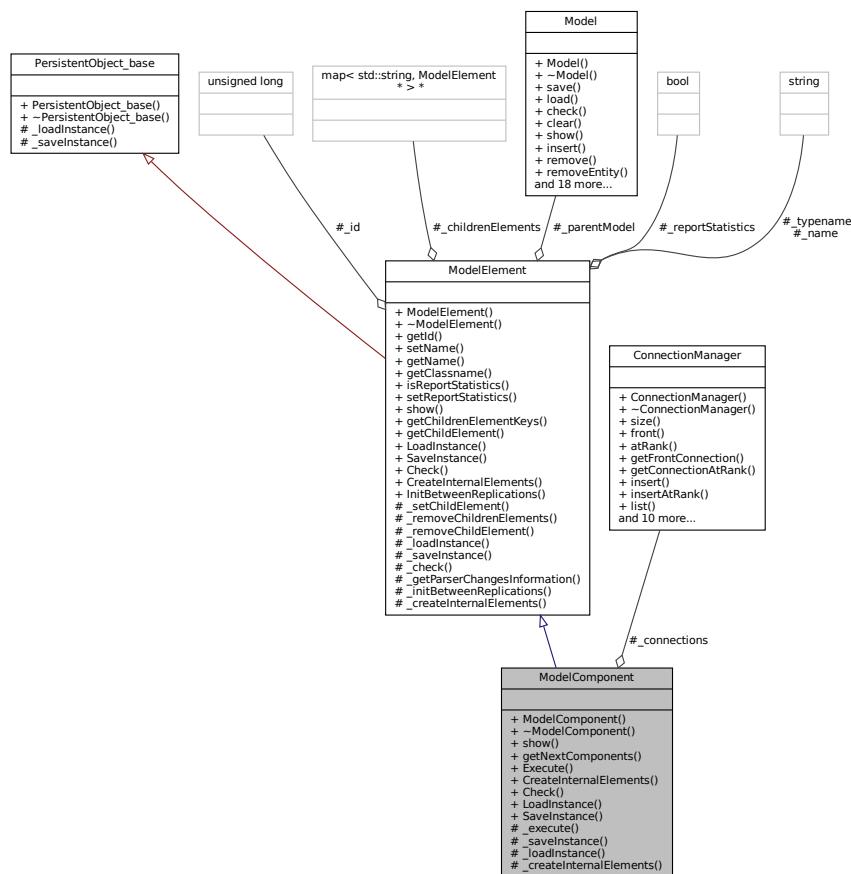
8.72 ModelComponent Class Reference

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

Inheritance diagram for ModelComponent:



Collaboration diagram for ModelComponent:



Public Member Functions

- [ModelComponent \(Model *model, std::string componentTypename, std::string name=""\)](#)
- virtual [~ModelComponent \(\)](#)
- virtual std::string [show \(\)](#)
- [ConnectionManager * 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](#)). In addition to the component, NextComponents specifies the inputNumber of the next component where the entity will be sent to. Usually the components have a single input, but they may have none (such as [Create](#)) or more than one (as [Match](#)).

Static Public Member Functions

- static void [Execute](#) ([Entity](#) *entity, [ModelComponent](#) *component, unsigned int inputNumber)

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 void [CreateInternalElements](#) ([ModelComponent](#) *component)
- static bool [Check](#) ([ModelComponent](#) *component)
- static [ModelComponent](#) * [LoadInstance](#) ([Model](#) *model, std::map< std::string, std::string > *fields)
- static std::map< std::string, std::string > * [SaveInstance](#) ([ModelComponent](#) *component)

Protected Member Functions

- virtual void [_execute](#) ([Entity](#) *entity)=0
- virtual std::map< std::string, std::string > * [_saveinstance](#) ()
- virtual bool [_loadinstance](#) (std::map< std::string, std::string > *fields)
- virtual void [_createinternalelements](#) ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Protected Attributes

- [ConnectionManager](#) * [_connections](#) = new [ConnectionManager](#)()

8.72.1 Detailed Description

A component of the model is a block that represents a specific behavior to be simulated. The behavior is triggered when an entity arrives at the component, which corresponds to the occurrence of an event. A simulation model corresponds to a set of interconnected components to form the process by which the entity is submitted.

Parameters

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

Definition at line 33 of file [ModelComponent.h](#).

8.72.2 Constructor & Destructor Documentation

8.72.2.1 ModelComponent()

```
ModelComponent::ModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "")
```

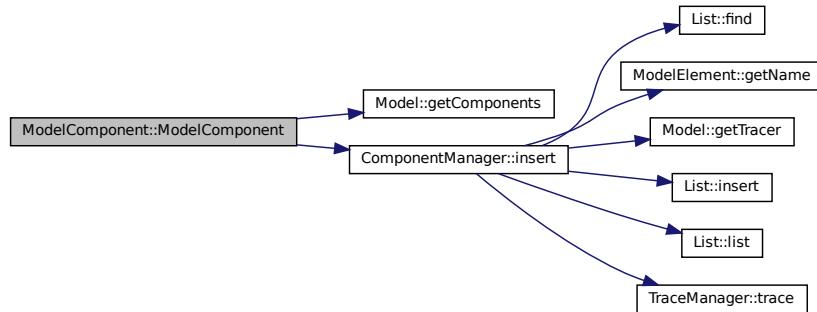
Definition at line 20 of file [ModelComponent.cpp](#).

```
00020     ModelElement(model, componentTypename, name, false) {
```

```
00021     model->getComponents()->insert(this);
00022 }
```

References [Model::getComponents\(\)](#), and [ComponentManager::insert\(\)](#).

Here is the call graph for this function:



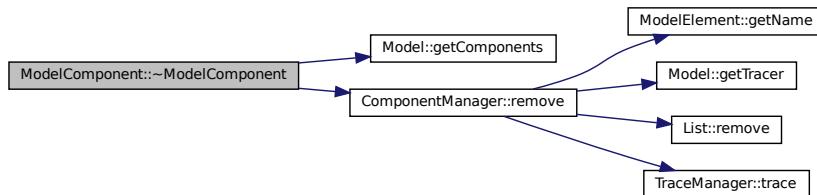
8.72.2.2 ~ModelComponent() ModelComponent::~ModelComponent () [virtual]

Definition at line 24 of file [ModelComponent.cpp](#).

```
00024 {
00025     _parentModel->getComponents()->remove(this);
00026 }
```

References [ModelElement::_parentModel](#), [Model::getComponents\(\)](#), and [ComponentManager::remove\(\)](#).

Here is the call graph for this function:



8.72.3 Member Function Documentation

8.72.3.1 `_createInternalElements()` void ModelComponent::_createInternalElements () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelElement](#).

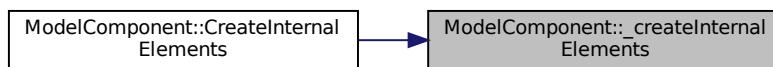
Reimplemented in [Enter](#), [Leave](#), [Route](#), [Create](#), [Delay](#), and [Dispose](#).

Definition at line 121 of file [ModelComponent.cpp](#).

```
00121
00122
00123 }
```

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

Here is the caller graph for this function:



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

Implemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Release](#), [Route](#), [Create](#), [PickStation](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), and [Submodel](#).

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

Here is the caller graph for this function:



8.72.3.3 `_loadInstance()` `bool ModelComponent::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Release](#), [Route](#), [Create](#), [PickStation](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [Start](#), [PickUp](#), [Signal](#), [SourceModelComponent](#), [Dispose](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 90 of file [ModelComponent.cpp](#).

```
00090
00091     bool res = ModelElement::_loadInstance(fields);
00092     if (res) {
00093         // Now it should load nextComponents. The problem is that the nextComponent may not be loaded
00094         // yet.
00095         // So, what can be done is to temporarily load the ID of the nextComponents, and to wait until
00096         // all the components have been loaded to update nextComponents based on the temporarilyIDs now being
00097         // loaded
00098         //unsigned short nextSize = std::stoi((*fields->find("nextSize")).second);
00099         //this->_tempLoadNextComponentsIDs = new List<Util::identification>();
00100         //for (unsigned short i = 0; i < nextSize; i++) {
00101         //    Util::identification nextId = std::stoi((*fields->find("nextId" +
00102             std::to_string(i))).second);
00103         //    this->_tempLoadNextComponentsIDs->insert(nextId);
00104         //}
00105     }
00106     return res;
00107 }
```

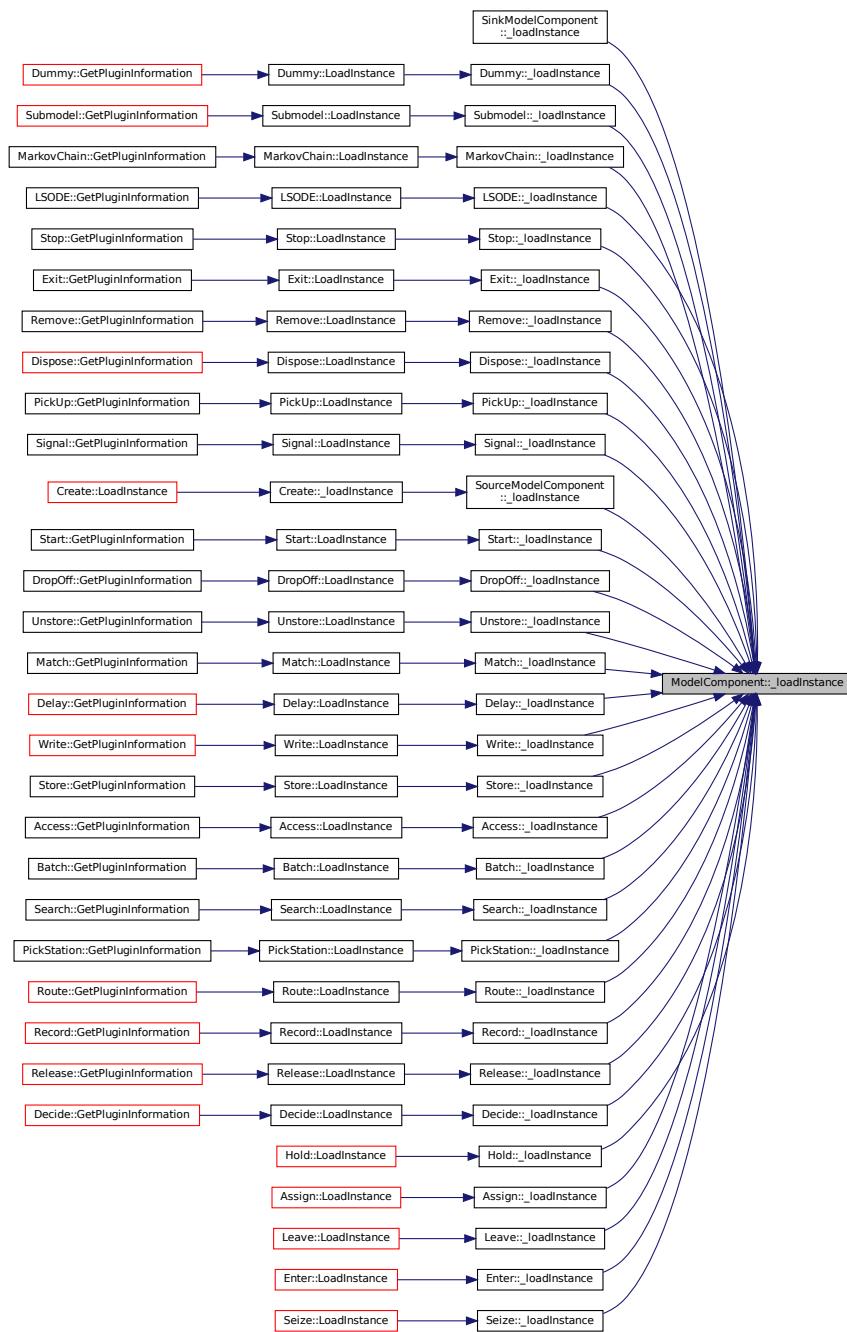
References [ModelElement::_loadInstance\(\)](#).

Referenced by [SinkModelComponent::_loadInstance\(\)](#), [Dummy::_loadInstance\(\)](#), [Submodel::_loadInstance\(\)](#), [MarkovChain::_loadInstance\(\)](#), [LSODE::_loadInstance\(\)](#), [Stop::_loadInstance\(\)](#), [Dispose::_loadInstance\(\)](#), [Exit::_loadInstance\(\)](#), [Remove::_loadInstance\(\)](#), [PickUp::_loadInstance\(\)](#), [Signal::_loadInstance\(\)](#), [SourceModelComponent::_loadInstance\(\)](#), [Start::_loadInstance\(\)](#), [DropOff::_loadInstance\(\)](#), [Unstore::_loadInstance\(\)](#), [Match::_loadInstance\(\)](#), [Delay::_loadInstance\(\)](#), [Write::_loadInstance\(\)](#), [Store::_loadInstance\(\)](#), [Access::_loadInstance\(\)](#), [Batch::_loadInstance\(\)](#), [Search::_loadInstance\(\)](#), [PickStation::_loadInstance\(\)](#), [Route::_loadInstance\(\)](#), [Record::_loadInstance\(\)](#), [Release::_loadInstance\(\)](#), [Decide::_loadInstance\(\)](#), [Hold::_loadInstance\(\)](#), [Assign::_loadInstance\(\)](#), [Leave::_loadInstance\(\)](#), [Enter::_loadInstance\(\)](#), and [Seize::_loadInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reimplemented from [ModelElement](#).

Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Release](#), [Route](#), [Create](#), [PickStation](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [SourceModelComponent](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 105 of file [ModelComponent.cpp](#).

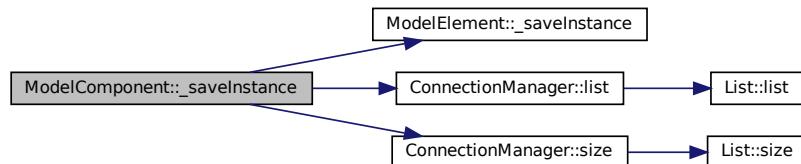
```

00105     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00106     if (true) { //(_connections->size() != 1) { // save nextSize only if it is != 1
00107         fields->emplace("nextSize", std::to_string(_connections->size()));
00108     }
00109     }
00110     unsigned short i = 0;
00111     for (std::list<Connection*>::iterator it = _connections->list()->begin(); it != _connections->list()->end(); it++) {
00112         fields->emplace("nextId" + std::to_string(i), std::to_string((*it)->first->_id));
00113         if (true) { //(*it)->second != 0) { // save nextInputNumber only if it is != 0
00114             fields->emplace("nextInputNumber" + std::to_string(i), std::to_string((*it)->second));
00115         }
00116         i++;
00117     }
00118     return fields;
00119 }
```

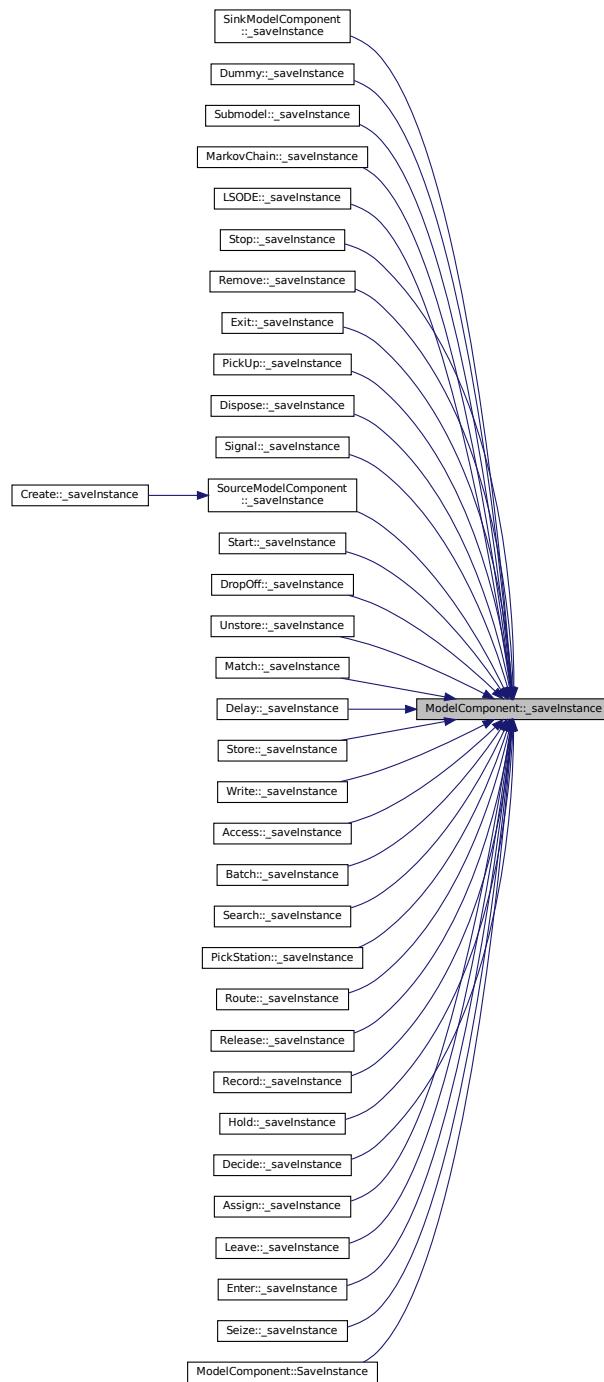
References [_connections](#), [ModelElement::_saveInstance\(\)](#), [ConnectionManager::list\(\)](#), and [ConnectionManager::size\(\)](#).

Referenced by [SinkModelComponent::_saveInstance\(\)](#), [Dummy::_saveInstance\(\)](#), [Submodel::_saveInstance\(\)](#), [MarkovChain::_saveInstance\(\)](#), [LSODE::_saveInstance\(\)](#), [Stop::_saveInstance\(\)](#), [Exit::_saveInstance\(\)](#), [Remove::_saveInstance\(\)](#), [PickUp::_saveInstance\(\)](#), [Dispose::_saveInstance\(\)](#), [Signal::_saveInstance\(\)](#), [SourceModelComponent::_saveInstance\(\)](#), [Start::_saveInstance\(\)](#), [DropOff::_saveInstance\(\)](#), [Unstore::_saveInstance\(\)](#), [Match::_saveInstance\(\)](#), [Delay::_saveInstance\(\)](#), [Write::_saveInstance\(\)](#), [Store::_saveInstance\(\)](#), [Access::_saveInstance\(\)](#), [Batch::_saveInstance\(\)](#), [Search::_saveInstance\(\)](#), [PickStation::_saveInstance\(\)](#), [Route::_saveInstance\(\)](#), [Record::_saveInstance\(\)](#), [Release::_saveInstance\(\)](#), [Hold::_saveInstance\(\)](#), [Decide::_saveInstance\(\)](#), [Assign::_saveInstance\(\)](#), [Leave::_saveInstance\(\)](#), [Enter::_saveInstance\(\)](#), [Seize::_saveInstance\(\)](#), and [SaveInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.72.3.5 Check() `bool ModelComponent::Check (`
 `ModelComponent * component) [static]`

Definition at line 63 of file [ModelComponent.cpp](#).

00063

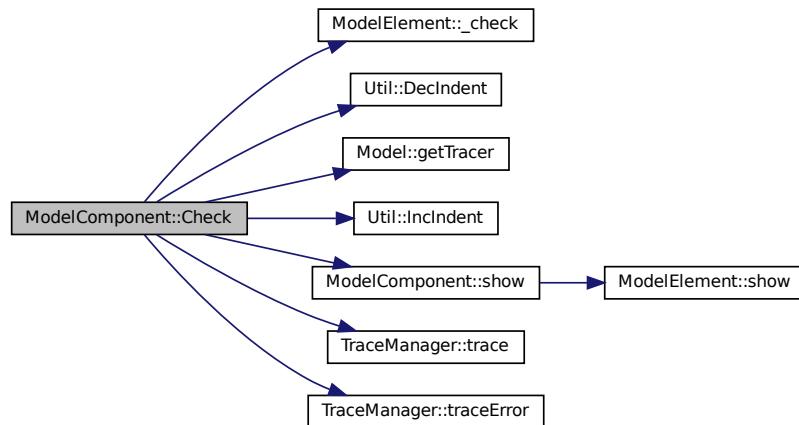
{

```

00064     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentDetailed, "Checking " +
00065     component->_typename + ":" + component->_name + "\n"); //std::to_string(component->_id));
00066     bool res = false;
00067     std::string* errorMessage = new std::string();
00068     Util::IncIndent();
00069     {
00070         try {
00071             res = component->_check(errorMessage);
00072             if (!res) {
00073                 component->_parentModel->getTracer()->trace(Util::TraceLevel::errorFatal, "Error:
00074                 Checking has failed with message '" + *errorMessage + "'");
00075             } catch (const std::exception& e) {
00076                 component->_parentModel->getTracer()->traceError(e, "Error verifying component " +
00077                 component->show());
00078             }
00079         Util::DecIndent();
00080     }
00080 }
```

References [ModelElement::_check\(\)](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_typename](#), [Util::componentDetailed](#), [Util::Declndent\(\)](#), [Util::errorFatal](#), [Model::getTracer\(\)](#), [Util::Inlndent\(\)](#), [show\(\)](#), [TraceManager::trace\(\)](#), and [TraceManager::traceError\(\)](#).

Here is the call graph for this function:



8.72.3.6 CreateInternalElements()

```

void ModelComponent::CreateInternalElements (
    ModelComponent * component ) [static]
```

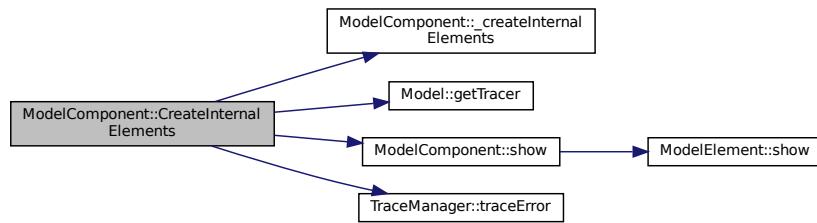
Definition at line 43 of file [ModelComponent.cpp](#).

```

00043
00044     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component
00045     \n" + component->_name + "\n"); //std::to_string(component->_id));
00046     try {
00047         component->_createInternalElements();
00048     } catch (const std::exception& e) {
00049         component->_parentModel->getTracer()->traceError(e, "Error creating elements of component " +
00050         component->show());
00050 }
```

References [_createInternalElements\(\)](#), [ModelElement::_parentModel](#), [Model::getTracer\(\)](#), [show\(\)](#), and [TraceManager::traceError\(\)](#).

Here is the call graph for this function:



8.72.3.7 Execute() void ModelComponent::Execute (Entity * entity,
ModelComponent * component,
unsigned int inputNumber) [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.

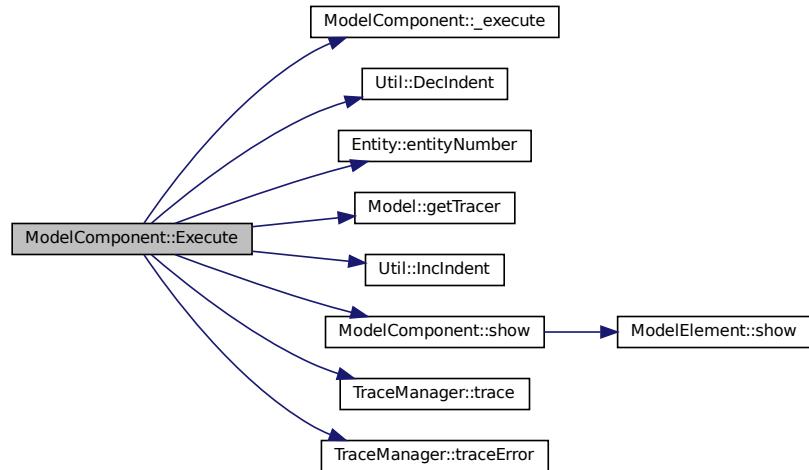
Definition at line 28 of file [ModelComponent.cpp](#).

```

00028
00029     std::string msg = "Entity " + std::to_string(entity->entityNumber()) + " has arrived at component
00030     \\" + component->_name + "\\";
00031     // \todo: How can I know the number of inputs?
00032     if (inputNumber > 0)
00033         msg += " by input " + std::to_string(inputNumber);
00034     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentArrival, msg);
00035     Util::IncIndent();
00036     try {
00037         component->_execute(entity);
00038     } catch (const std::exception& e) {
00039         component->_parentModel->getTracer()->traceError(e, "Error executing component " +
00040             component->show());
00041     }
00040     Util::DecIndent();
00041 }
```

References [_execute\(\)](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [Util::componentArrival](#), [Util::DecIndent\(\)](#), [Entity::entityNumber\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [show\(\)](#), [TraceManager::trace\(\)](#), and [TraceManager::traceError\(\)](#).

Here is the call graph for this function:



8.72.3.8 `getNextComponents()`

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`). In addition to the component, `NextComponents` specifies the `inputNumber` of the next component where the entity will be sent to. Ussually the components have a single input, but they may have none (such as `Create`) or more than one (as `Match`).

Definition at line 82 of file `ModelComponent.cpp`.

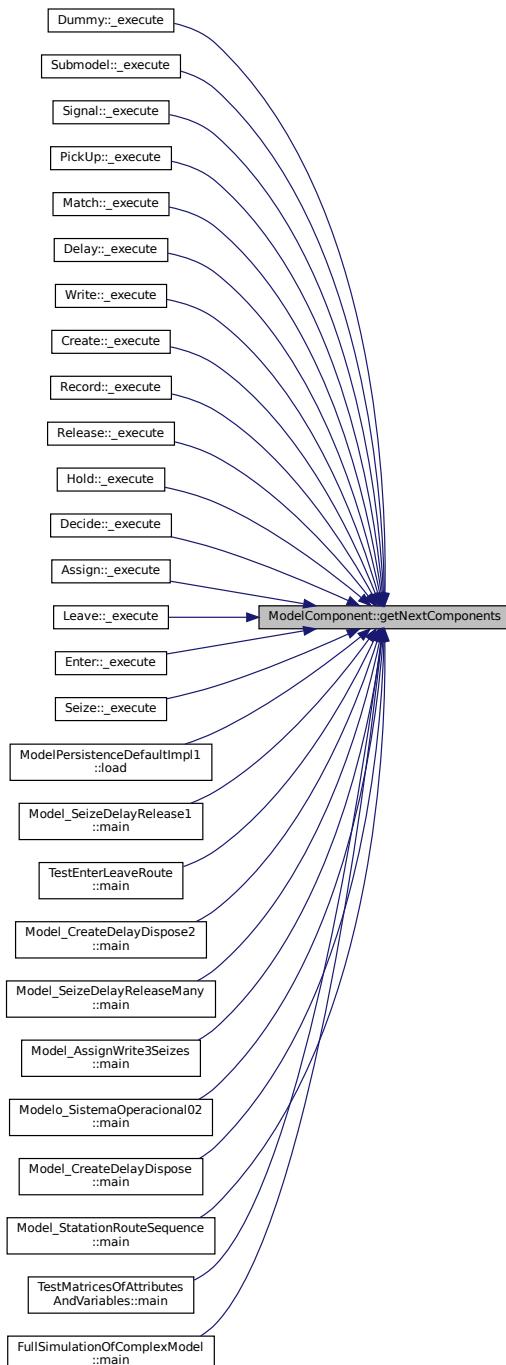
```

00082
00083     return _connections;
00084 }
```

References `_connections`.

Referenced by `Dummy::_execute()`, `Submodel::_execute()`, `PickUp::_execute()`, `Signal::_execute()`, `Match::_execute()`, `Delay::_execute()`, `Write::_execute()`, `Create::_execute()`, `Record::_execute()`, `Release::_execute()`, `Hold::_execute()`, `Decide::_execute()`, `Assign::_execute()`, `Leave::_execute()`, `Enter::_execute()`, `Seize::_execute()`, `ModelPersistenceDefaultImpl1::loadModel`, `Model_CreateDelayDispose2::main()`, `Model_CreateDelayDispose::main()`, `TestEnterLeaveRoute::main()`, `Model_StatationRouteSequence::main()`, `Model_SeizeDelayRelease1::main()`, `Model_AssignWrite3Seizes::main()`, `Model_SeizeDelayReleaseMany::main()`, `Modelo_SistemaOperacional02::main()`, `TestMatricesOfAttributesAndVariables::main()`, and `FullSimulationOfComplexModel::main()`.

Here is the caller graph for this function:



```

8.72.3.9 LoadInstance() static ModelComponent* ModelComponent::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
  
```

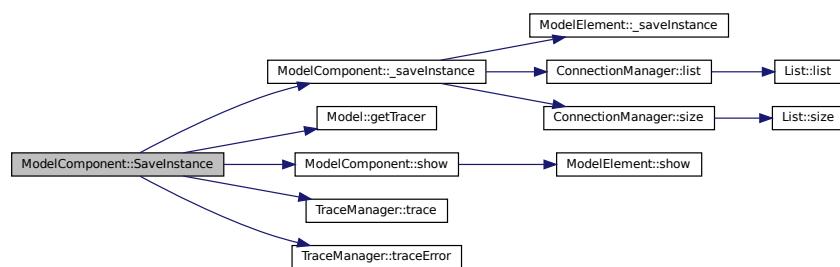
8.72.3.10 `SaveInstance()` `std::map< std::string, std::string > * ModelComponent::SaveInstance (ModelComponent * component) [static]`

Definition at line 52 of file [ModelComponent.cpp](#).

```
00052
00053     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentDetailed, "Writing
00054     component \" + component->_name + "\"; //std::to_string(component->_id));
00055     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00056     try {
00057         fields = component->_saveInstance();
00058     } catch (const std::exception& e) {
00059         component->_parentModel->getTracer()->traceError(e, "Error executing component " +
00060         component->show());
00061     }
00062     return fields;
00063 }
```

References [ModelElement::_name](#), [ModelElement::_parentModel](#), [_saveInstance\(\)](#), [Util::componentDetailed](#), [Model::getTracer\(\)](#), [show\(\)](#), [TraceManager::trace\(\)](#), and [TraceManager::traceError\(\)](#).

Here is the call graph for this function:



8.72.3.11 `show()` `std::string ModelComponent::show () [virtual]`

Reimplemented from [ModelElement](#).

Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Release](#), [PickStation](#), [Route](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Delay](#), [Match](#), [Unstore](#), [SourceModelComponent](#), [Write](#), [DropOff](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [Dummy](#), [Submodel](#), and [MarkovChain](#).

Definition at line 86 of file [ModelComponent.cpp](#).

```
00086
00087     return ModelElement::show(); // "{id=" + std::to_string(this->_id) + ",name=\"" + this->_name +
00088     "\"}"; // , nextComponents[] = (" + _nextComponents->show() + ")\"};
```

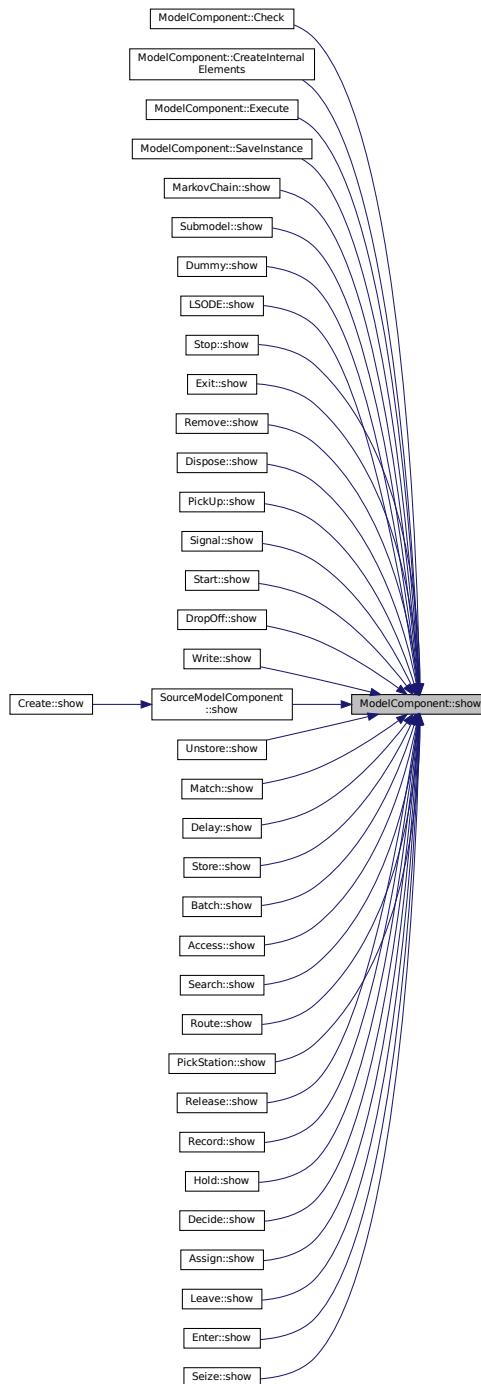
References [ModelElement::show\(\)](#).

Referenced by [Check\(\)](#), [CreateInternalElements\(\)](#), [Execute\(\)](#), [SaveInstance\(\)](#), [MarkovChain::show\(\)](#), [Submodel::show\(\)](#), [Dummy::show\(\)](#), [LSODE::show\(\)](#), [Stop::show\(\)](#), [Exit::show\(\)](#), [Remove::show\(\)](#), [Dispose::show\(\)](#), [PickUp::show\(\)](#), [Signal::show\(\)](#), [Start::show\(\)](#), [DropOff::show\(\)](#), [Write::show\(\)](#), [SourceModelComponent::show\(\)](#), [Unstore::show\(\)](#), [Match::show\(\)](#), [Delay::show\(\)](#), [Store::show\(\)](#), [Access::show\(\)](#), [Batch::show\(\)](#), [Search::show\(\)](#), [Route::show\(\)](#), [PickStation::show\(\)](#), [Release::show\(\)](#), [Record::show\(\)](#), [Hold::show\(\)](#), [Decide::show\(\)](#), [Assign::show\(\)](#), [Leave::show\(\)](#), [Enter::show\(\)](#), and [Seize::show\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.72.4 Member Data Documentation

8.72.4.1 `_connections` `ConnectionManager* ModelComponent::_connections = new ConnectionManager()`
`[protected]`

Definition at line 55 of file [ModelComponent.h](#).

Referenced by [_saveInstance\(\)](#), [Create::Create\(\)](#), [Dispose::Dispose\(\)](#), and [getNextComponents\(\)](#).

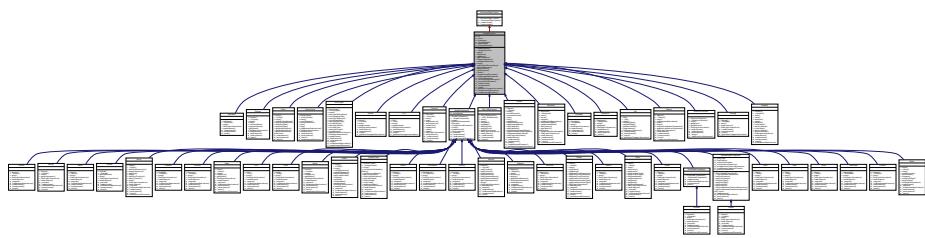
The documentation for this class was generated from the following files:

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

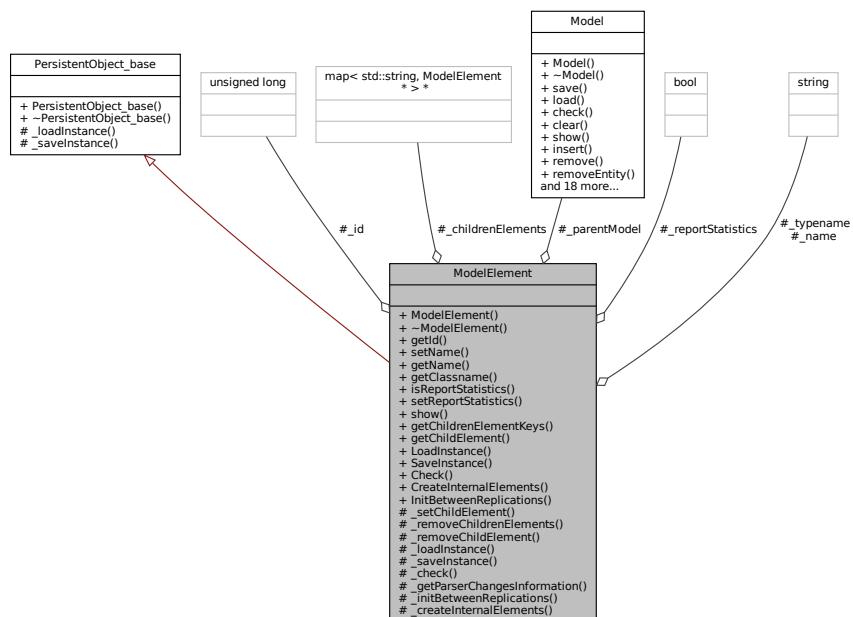
8.73 ModelElement Class Reference

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

Inheritance diagram for ModelElement:



Collaboration diagram for ModelElement:



Public Member Functions

- `ModelElement (Model *model, std::string elementTypename, std::string name="", bool insertIntoModel=true)`
- `virtual ~ModelElement ()`
- `Util::identification getId () const`
- `void setName (std::string _name)`
- `std::string getName () const`
- `std::string getClassname () const`
- `bool isReportStatistics () const`
- `void setReportStatistics (bool reportStatistics)`
- `virtual std::string show ()`
- `std::list< std::string > * getChildrenElementKeys () const`
- `ModelElement * getChildElement (std::string key) const`

Static Public Member Functions

- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields, bool insertIntoModel)`
- static `std::map< std::string, std::string > * SaveInstance (ModelElement *element)`
- static `bool Check (ModelElement *element, std::string *errorMessage)`
- static `void CreateInternalElements (ModelElement *element)`
- static `void InitBetweenReplications (ModelElement *element)`

Protected Member Functions

- `void _setChildElement (std::string key, ModelElement *child)`
- `void _removeChildrenElements ()`
- `void _removeChildElement (std::string key)`
- `virtual bool _loadInstance (std::map< std::string, std::string > *fields)`
- `virtual std::map< std::string, std::string > * _saveInstance ()`
- `virtual bool _check (std::string *errorMessage)`
- `virtual ParserChangesInformation * _getParserChangesInformation ()`
- `virtual void _initBetweenReplications ()`
- `virtual void _createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Protected Attributes

- `Util::identification _id`
- `std::string _name`
- `std::string _typename`
- `bool _reportStatistics`
- `Model * _parentModel`
- `std::map< std::string, ModelElement * > * _childrenElements = new std::map< std::string, ModelElement * >()`

8.73.1 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.

Definition at line 33 of file [ModelElement.h](#).

8.73.2 Constructor & Destructor Documentation

8.73.2.1 ModelElement()

```
ModelElement::ModelElement (
    Model * model,
    std::string elementTypename,
    std::string name = "",
    bool insertIntoModel = true )
```

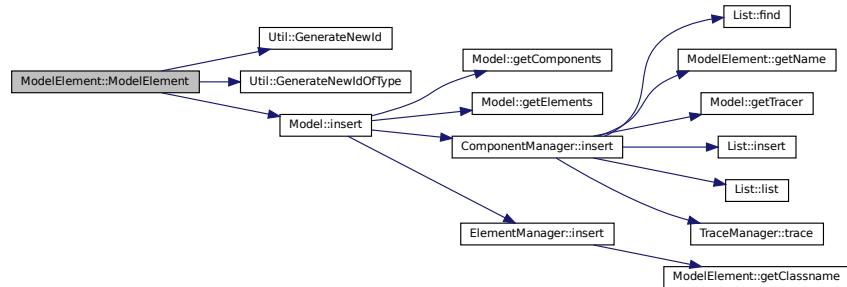
Definition at line 22 of file [ModelElement.cpp](#).

```
00022
00023     _id = Util::GenerateNewId(); //GenerateNewIdOfType(thistypename);
00024     _typename = thistypename;
00025     _parentModel = model;
00026     _reportStatistics = Traits<ModelElement>::reportStatistics; // \todo: shouould be a parameter
00027     before insertIntoModel
00028         if (name == "")
00029             _name = thistypename + "_" + std::to_string(Util::GenerateNewIdOfType(thistypename));
00030         else
00031             _name = name;
00032         if (insertIntoModel) {
00033             model->insert(this);
00034 }
```

References [_id](#), [_name](#), [_parentModel](#), [_reportStatistics](#), [_typename](#), [Util::GenerateNewId\(\)](#), [Util::GenerateNewIdOfType\(\)](#), and [Model::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



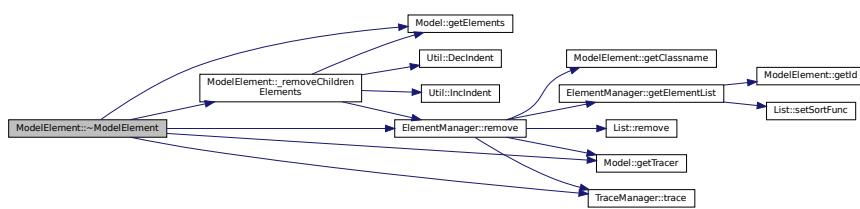
8.73.2.2 ~ModelElement() ModelElement::~ModelElement () [virtual]

Definition at line 42 of file [ModelElement.cpp](#).

```
00042     {
00043         _parentModel->getTracer()->trace(Util::TraceLevel::everythingMostDetailed, "Removing Element \\" + 
00044             this->_name + "\\ from the model");
00045         _removeChildrenElements();
00046         _parentModel->getElements()->remove(this);
00047     }
```

References [_name](#), [_parentModel](#), [_removeChildrenElements\(\)](#), [Util::everythingMostDetailed](#), [Model::getElements\(\)](#), [Model::getTracer\(\)](#), [ElementManager::remove\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.73.3 Member Function Documentation

8.73.3.1 _check() bool ModelElement::_check (std::string * errorMessage) [protected], [virtual]

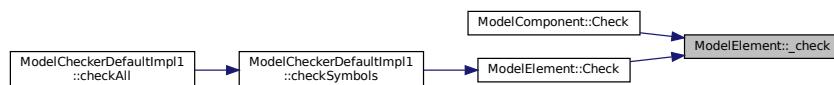
Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Sequence](#), [Decide](#), [Hold](#), [Schedule](#), [Record](#), [Release](#), [Route](#), [Station](#), [Create](#), [Failure](#), [File](#), [PickStation](#), [Attribute](#), [Set](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [EntityType](#), [OLD_ODElement](#), [SourceModelComponent](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Storage](#), [Stop](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [StatisticsCollector](#), [Counter](#), [Formula](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 100 of file [ModelElement.cpp](#).

```
00100     {
00101         return true; // if there is no override, return true
00102     }
```

Referenced by [ModelComponent::Check\(\)](#), and [Check\(\)](#).

Here is the caller graph for this function:



8.73.3.2 `_createInternalElements()` void ModelElement::`_createInternalElements` () [protected], [virtual]

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

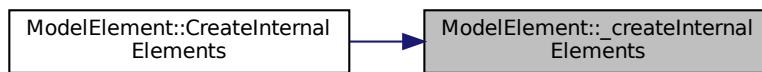
Reimplemented in [Resource](#), [Queue](#), [Enter](#), [Leave](#), [Route](#), [Station](#), [Create](#), [Delay](#), [EntityType](#), [Dispose](#), [ModelComponent](#), and [EntityGroup](#).

Definition at line 217 of file [ModelElement.cpp](#).

```
00217
00218
00219 }
```

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

Here is the caller graph for this function:



8.73.3.3 `_getParserChangesInformation()` ParserChangesInformation * ModelElement::`_getParserChangesInformation` () [protected], [virtual]

Reimplemented in [Queue](#), [Failure](#), [File](#), [Set](#), and [Storage](#).

Definition at line 104 of file [ModelElement.cpp](#).

```
00104
00105     return new ParserChangesInformation(); // if there is no override, return no changes
00106 }
```

8.73.3.4 `_initBetweenReplications()` void ModelElement::`_initBetweenReplications` () [protected], [virtual]

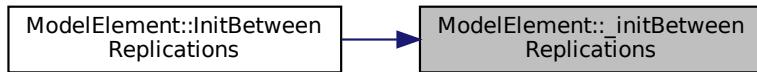
Reimplemented in [Seize](#), [Variable](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Release](#), [Create](#), [Route](#), [PickStation](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [EntityType](#), [DropOff](#), [SourceModelComponent](#), [Dispose](#), [Start](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 108 of file [ModelElement.cpp](#).

```
00108
00109
00110 }
```

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

Here is the caller graph for this function:



8.73.3.5 `_loadInstance()` `bool ModelElement::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]`

Implements [PersistentObject_base](#).

Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Sequence](#), [Decide](#), [Hold](#), [Schedule](#), [Record](#), [Release](#), [Route](#), [Station](#), [Create](#), [Failure](#), [File](#), [PickStation](#), [Attribute](#), [Set](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [ModelComponent](#), [EntityType](#), [OLD_ODElement](#), [Start](#), [PickUp](#), [Signal](#), [SourceModelComponent](#), [Dispose](#), [Exit](#), [Remove](#), [Stop](#), [Storage](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [StatisticsCollector](#), [Counter](#), [Formula](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

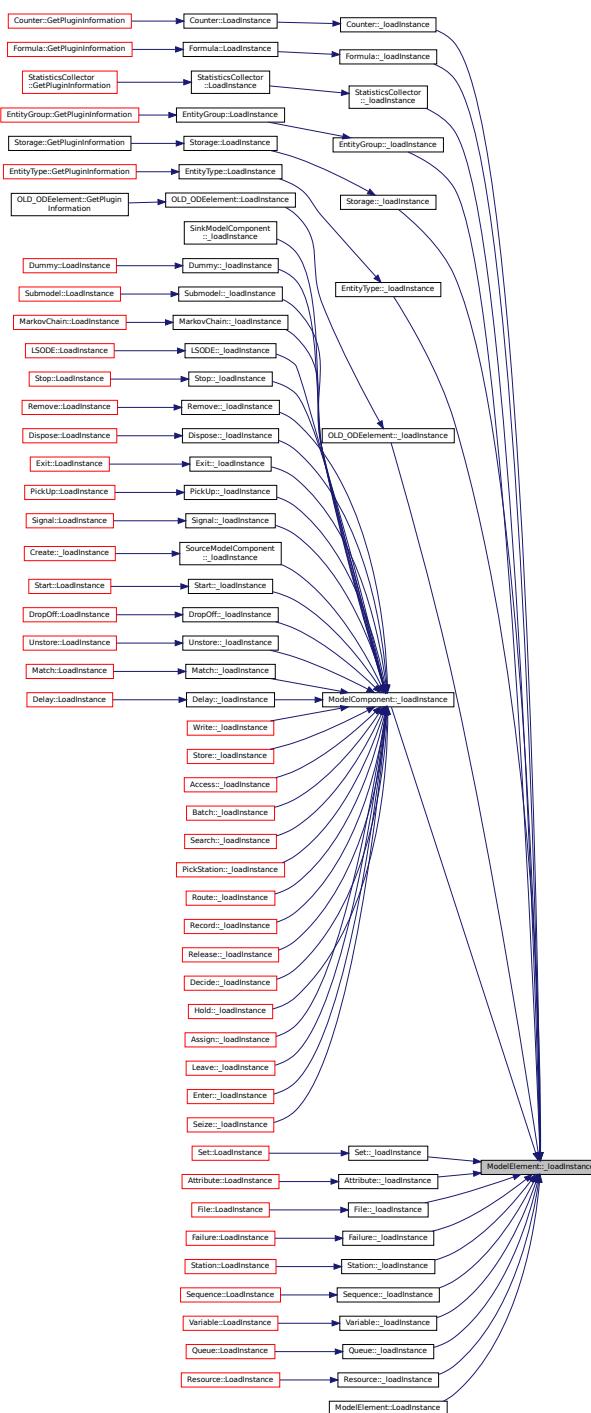
Definition at line 78 of file [ModelElement.cpp](#).

```
00078     bool res = true;
00079     std::map<std::string, std::string>::iterator it;
00080     it = fields->find("id");
00081     if (it != fields->end()) ? this->_id = std::stoi((*it).second) : res = false;
00082     it = fields->find("name");
00083     if (it != fields->end()) ? this->_name = (*it).second : this->_name = "";
00084     //if (it != fields->end()) ? this->_name = (*it).second : res = false;
00085     it = fields->find("reportStatistics");
00086     if (it != fields->end()) ? this->_reportStatistics = std::stoi((*it).second) : this->_reportStatistics
00087     = Traits<ModelElement>::reportStatistics;
00088     return res;
00089 }
```

References [_id](#), [_name](#), and [_reportStatistics](#).

Referenced by [Counter::_loadInstance\(\)](#), [Formula::_loadInstance\(\)](#), [StatisticsCollector::_loadInstance\(\)](#), [EntityGroup::_loadInstance\(\)](#), [Storage::_loadInstance\(\)](#), [EntityType::_loadInstance\(\)](#), [OLD_ODElement::_loadInstance\(\)](#), [ModelComponent::_loadInstance\(\)](#), [Set::_loadInstance\(\)](#), [Attribute::_loadInstance\(\)](#), [File::_loadInstance\(\)](#), [Failure::_loadInstance\(\)](#), [Station::_loadInstance\(\)](#), [Sequence::_loadInstance\(\)](#), [Variable::_loadInstance\(\)](#), [Queue::_loadInstance\(\)](#), [Resource::_loadInstance\(\)](#), and [LoadInstance\(\)](#).

Here is the caller graph for this function:



8.73.3.6 `_removeChildElement()` void ModelElement::`_removeChildElement` (std::string key) [protected]

Definition at line 61 of file [ModelElement.cpp](#).

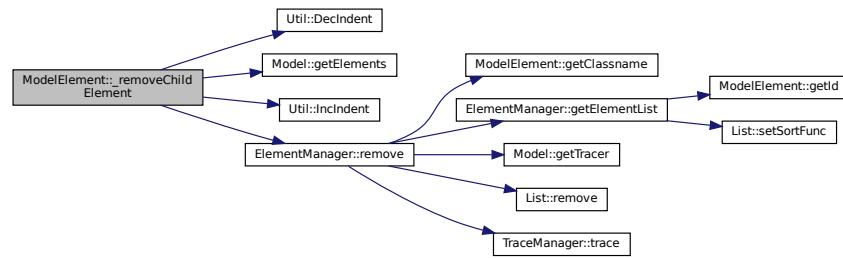
```

00062     Util::IncIndent();
00063     {
00064         //for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it != _childrenElements->end(); it++) {
00065             std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin();
00066             while (it != _childrenElements->end()) {
00067                 if ((*it).first == key) {
00068                     this->_parentModel->getElements()->remove((*it).second);
00069                     (*it).second->~ModelElement();
00070                     _childrenElements->erase(it);
00071                     it = _childrenElements->begin();
00072                 }
00073             }
00074         }
00075         Util::DecIndent();
00076     }

```

References `_childrenElements`, `_parentModel`, `Util::DecIndent()`, `Model::getElements()`, `Util::IncIndent()`, and `ElementManager::remove()`.

Here is the call graph for this function:



8.73.3.7 `_removeChildrenElements()` void ModelElement::_removeChildrenElements () [protected]

Definition at line 48 of file `ModelElement.cpp`.

```

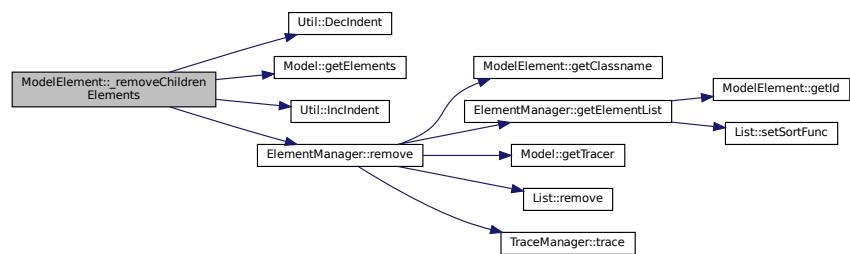
00048     Util::IncIndent();
00049     {
00050         for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it != _childrenElements->end(); it++) {
00051             this->_parentModel->getElements()->remove((*it).second);
00052             (*it).second->~ModelElement();
00053         }
00054         _childrenElements->clear();
00055     }
00056     Util::DecIndent();
00057
00058 }

```

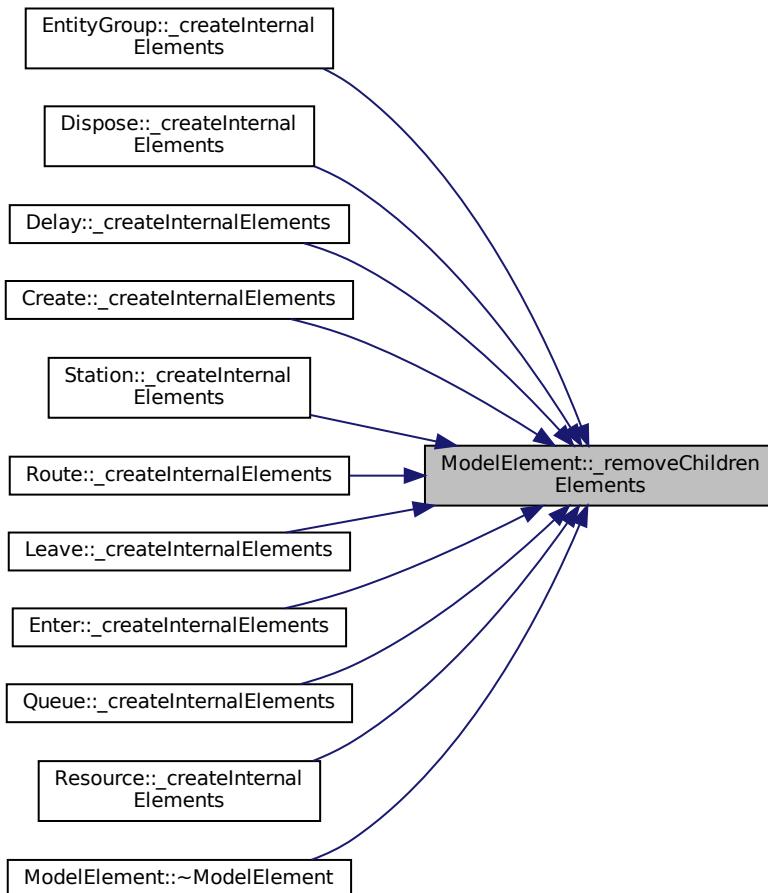
References `_childrenElements`, `_parentModel`, `Util::DecIndent()`, `Model::getElements()`, `Util::IncIndent()`, and `ElementManager::remove()`.

Referenced by `EntityGroup::_createInternalElements()`, `Dispose::_createInternalElements()`, `Delay::_createInternalElements()`, `Create::_createInternalElements()`, `Station::_createInternalElements()`, `Route::_createInternalElements()`, `Leave::_createInternalElements()`, `Enter::_createInternalElements()`, `Queue::_createInternalElements()`, `Resource::_createInternalElements()`, and `~ModelElement()`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

Implements [PersistentObject_base](#).

Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Sequence](#), [Decide](#), [Hold](#), [Schedule](#), [Record](#), [Release](#), [Route](#), [Station](#), [Create](#), [Failure](#), [File](#), [PickStation](#), [Attribute](#), [Set](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [EntityType](#), [OLD_ODElement](#), [SourceModelComponent](#), [Start](#), [Dispose](#), [ModelComponent](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [Storage](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [StatisticsCollector](#), [Counter](#), [Formula](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

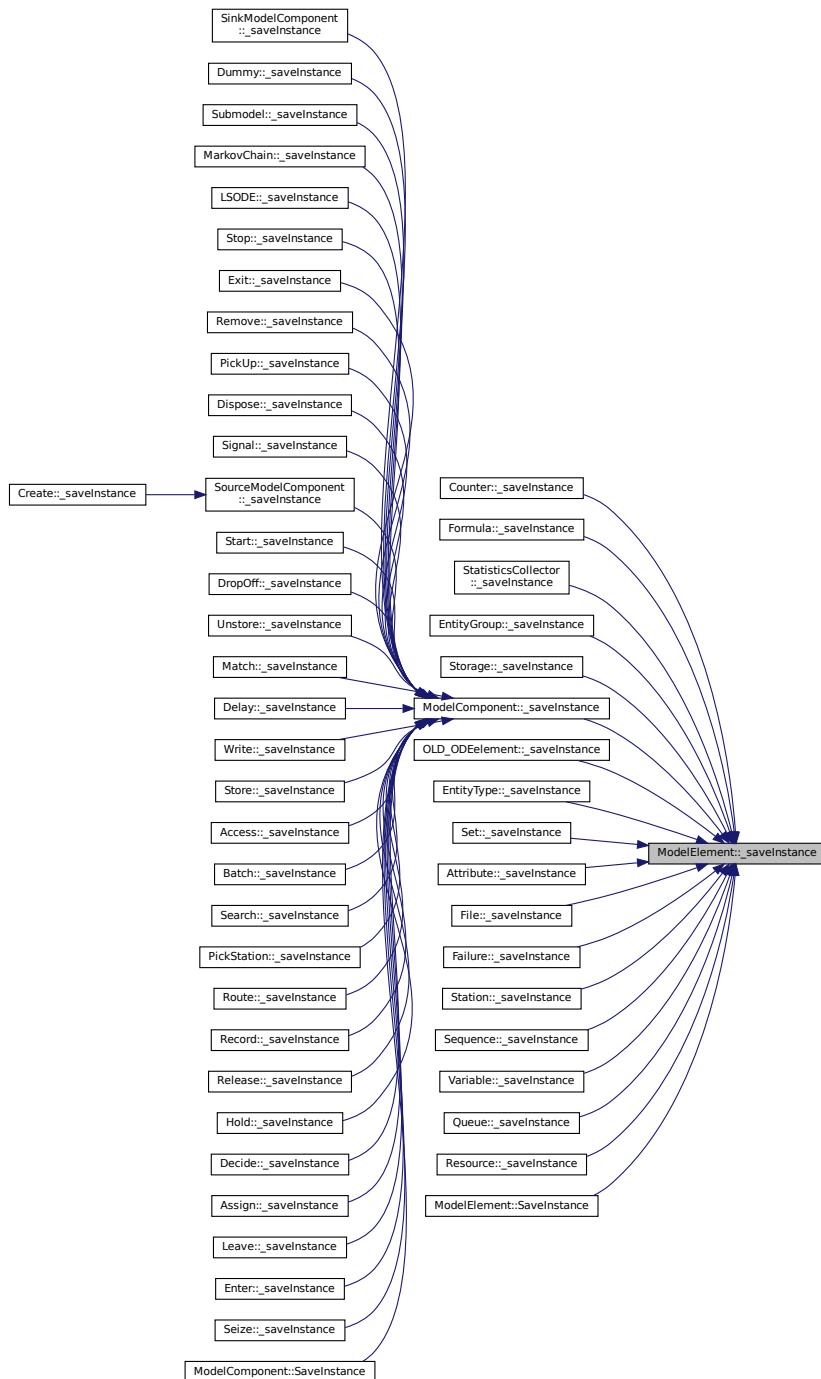
Definition at line 91 of file [ModelElement.cpp](#).

```
00091
00092     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00093     fields->emplace("typename", this->typename);
00094     fields->emplace("id", std::to_string(this->id));
00095     fields->emplace("name", "\"" + this->name + "\"");
00096     if (this->reportStatistics != Traits<ModelElement>::reportStatistics)
00097         fields->emplace("reportStatistics", std::to_string(this->reportStatistics));
00098     return fields;
00099 }
```

References [_id](#), [_name](#), [_reportStatistics](#), and [_typename](#).

Referenced by [Counter::_saveInstance\(\)](#), [Formula::_saveInstance\(\)](#), [StatisticsCollector::_saveInstance\(\)](#), [EntityGroup::_saveInstance\(\)](#), [Storage::_saveInstance\(\)](#), [ModelComponent::_saveInstance\(\)](#), [EntityType::_saveInstance\(\)](#), [OLD_ODElement::_saveInstance\(\)](#), [Set::_saveInstance\(\)](#), [Attribute::_saveInstance\(\)](#), [File::_saveInstance\(\)](#), [Failure::_saveInstance\(\)](#), [Station::_saveInstance\(\)](#), [Sequence::_saveInstance\(\)](#), [Variable::_saveInstance\(\)](#), [Queue::_saveInstance\(\)](#), [Resource::_saveInstance\(\)](#), and [SaveInstance\(\)](#).

Here is the caller graph for this function:



8.73.3.9 `_setChildElement()` void ModelElement::_setChildElement (std::string key, ModelElement * child) [protected]

8.73.3.10 Check() `bool ModelElement::Check (`

```
    ModelElement * element,
    std::string * errorMessage ) [static]
```

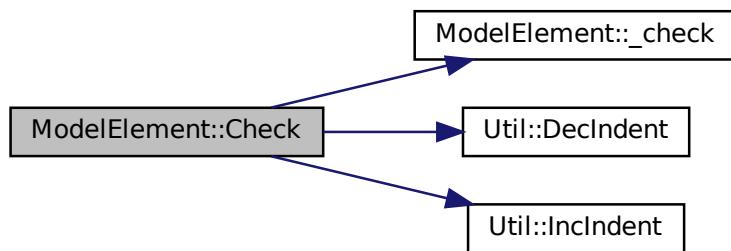
Definition at line 189 of file `ModelElement.cpp`.

```
00189
00190     //   element->_model->getTraceManager()->trace(Util::TraceLevel::mostDetailed, "Checking " +
00191     element->_typename + ":" + element->_name); //std::to_string(element->_id));
00192     bool res = false;
00193     Util::IncIndent();
00194     {
00195         try {
00196             res = element->_check(errorMessage);
00197             if (!res) {
00198                 //   element->_model->getTraceManager()->trace(Util::TraceLevel::errors,
00199                 "Error: Checking has failed with message '" + *errorMessage + "'");
00200             } catch (const std::exception& e) {
00201                 //   element->_model->getTraceManager()->traceError(e, "Error verifying element "
00202                 + element->show());
00203             }
00204         }
00205     }
00206 }
```

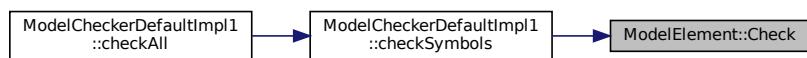
References `_check()`, `Util::DecIndent()`, and `Util::IncIndent()`.

Referenced by `ModelCheckerDefaultImpl1::checkSymbols()`.

Here is the call graph for this function:



Here is the caller graph for this function:



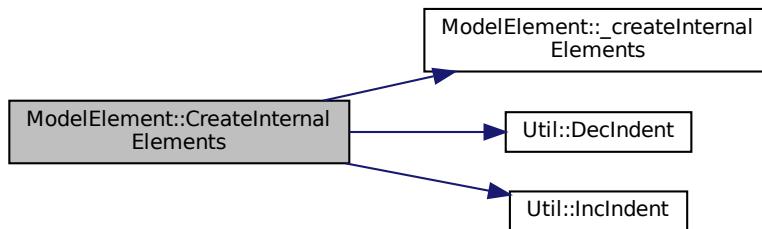
8.73.3.11 CreateInternalElements() void ModelElement::CreateInternalElements (ModelElement * element) [static]

Definition at line 207 of file [ModelElement.cpp](#).

```
00207                                         {
00208     try {
00209         Util::IncIndent ();
00210         element->_createInternalElements ();
00211         Util::DecIndent ();
00212     } catch (const std::exception& e) {
00213         //element->...->_model->getTraceManager()->traceError(e, "Error creating elements of element "
00214         + element->show ());
00215 }
```

References [_createInternalElements\(\)](#), [Util::DecIndent\(\)](#), and [Util::IncIndent\(\)](#).

Here is the call graph for this function:



8.73.3.12 getChildElement() ModelElement* ModelElement::getChildElement (std::string key) const

8.73.3.13 getChildrenElementKeys() std::list< std::string > * ModelElement::getChildrenElementKeys () const

Definition at line 132 of file [ModelElement.cpp](#).

```
00132                                         {
00133     std::list<std::string>* result = new std::list<std::string>();
00134     return result;
00135 }
```

8.73.3.14 `getClassName()` `std::string ModelElement::getClassName() const`

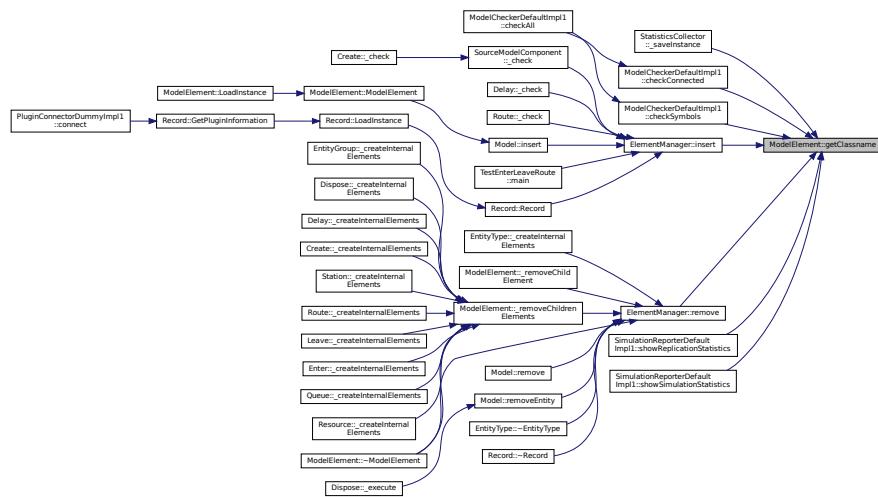
Definition at line 149 of file `ModelElement.cpp`.

```
00149
00150     return _typename;
00151 }
```

References `_typename`.

Referenced by `StatisticsCollector::_saveInstance()`, `ModelCheckerDefaultImpl1::checkConnected()`, `ModelCheckerDefaultImpl1::checkSymbols()`, `ElementManager::insert()`, `ElementManager::remove()`, `SimulationReporterDefaultImpl1::showReplicationStatistics()`, and `SimulationReporterDefaultImpl1::showSimulationStatistics()`.

Here is the caller graph for this function:



8.73.3.15 `getId()` `Util::identification ModelElement::getId() const`

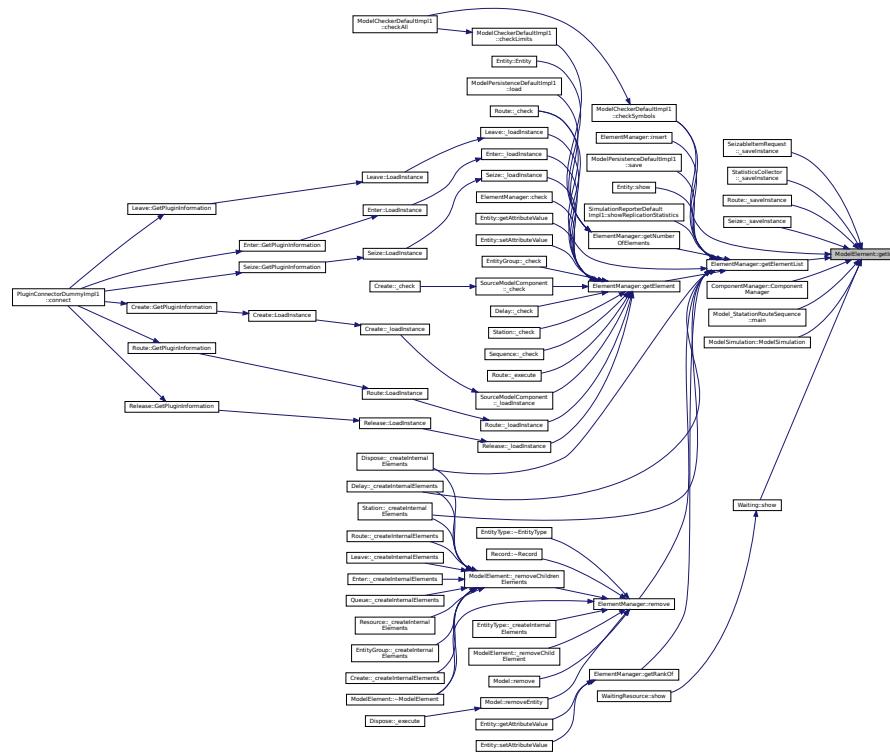
Definition at line 137 of file `ModelElement.cpp`.

```
00137
00138     return _id;
00139 }
```

References `_id`.

Referenced by `SeizableItemRequest::_saveInstance()`, `StatisticsCollector::_saveInstance()`, `Route::_saveInstance()`, `Seize::_saveInstance()`, `ModelCheckerDefaultImpl1::checkSymbols()`, `ComponentManager::ComponentManager()`, `ElementManager::getElementList()`, `Model_StatationRouteSequence::main()`, `ModelSimulation::ModelSimulation()`, and `Waiting::show()`.

Here is the caller graph for this function:



8.73.3.16 getName() std::string ModelElement::getName() const

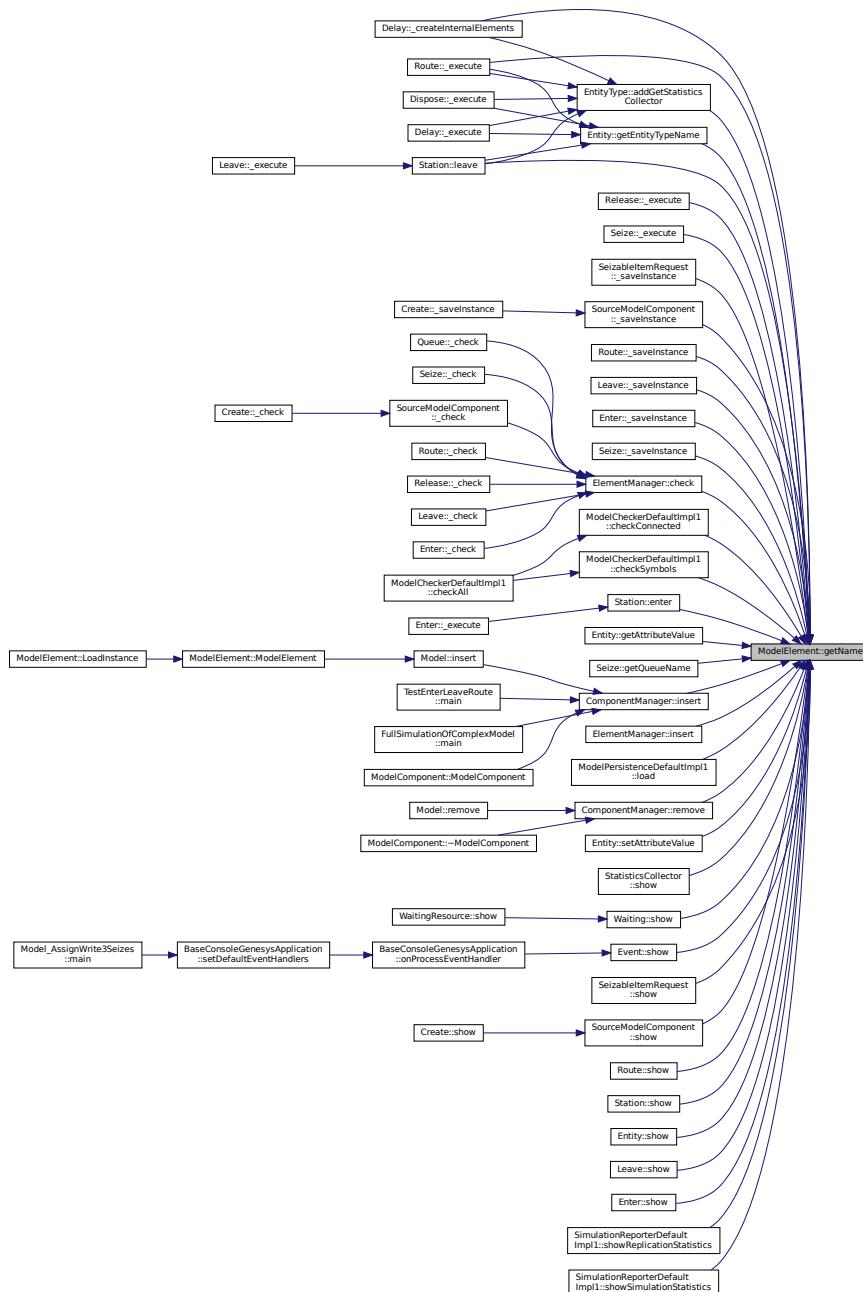
Definition at line 145 of file [ModelElement.cpp](#).

```
00145
00146     return _name;
00147 }
```

References [_name](#).

Referenced by [Delay::createInternalElements\(\)](#), [Route::_execute\(\)](#), [Release::_execute\(\)](#), [Seize::_execute\(\)](#), [SeizableItemRequest::_saveInstance\(\)](#), [SourceModelComponent::_saveInstance\(\)](#), [Route::_saveInstance\(\)](#), [Leave::_saveInstance\(\)](#), [Enter::_saveInstance\(\)](#), [Seize::_saveInstance\(\)](#), [EntityType::addGetStatisticsCollector\(\)](#), [ElementManager::check\(\)](#), [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [Station::enter\(\)](#), [Entity::getAttributeValue\(\)](#), [Entity::getEntityType\(\)](#), [Seize::getQueueName\(\)](#), [ComponentManager::insert\(\)](#), [ElementManager::insert\(\)](#), [Station::leave\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [ComponentManager::remove\(\)](#), [Entity::setAttributeValue\(\)](#), [StatisticsCollector::show\(\)](#), [Waiting::show\(\)](#), [Event::show\(\)](#), [SeizableItemRequest::show\(\)](#), [SourceModelComponent::show\(\)](#), [Route::show\(\)](#), [Station::show\(\)](#), [Entity::show\(\)](#), [Leave::show\(\)](#), [Enter::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.73.3.17 InitBetweenReplications() void ModelElement::InitBetweenReplications (ModelElement * element) [static]

Definition at line 153 of file [ModelElement.cpp](#).

```

00153
00154     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component
00155     \'" + component->_name + "\"); //std::to_string(component->_id));
00156     try {
00157         element->_initBetweenReplications();
00158     } catch (const std::exception& e) {

```

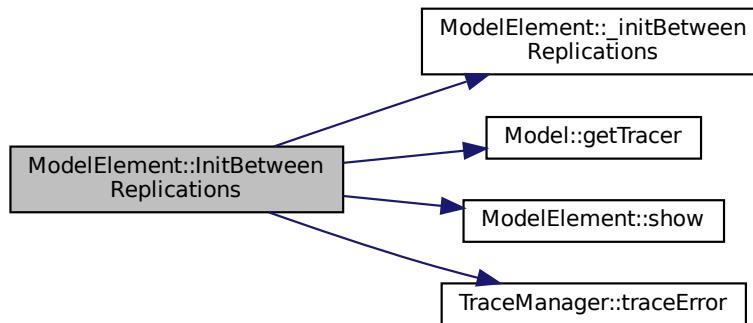
```

00158     element->_parentModel->getTracer()->traceError(e, "Error initing component " +
00159         );
00160 }

```

References [_initBetweenReplications\(\)](#), [_parentModel](#), [Model::getTracer\(\)](#), [show\(\)](#), and [TraceManager::traceError\(\)](#).

Here is the call graph for this function:



8.73.3.18 isReportStatistics() bool ModelElement::isReportStatistics () const

Definition at line 225 of file [ModelElement.cpp](#).

```

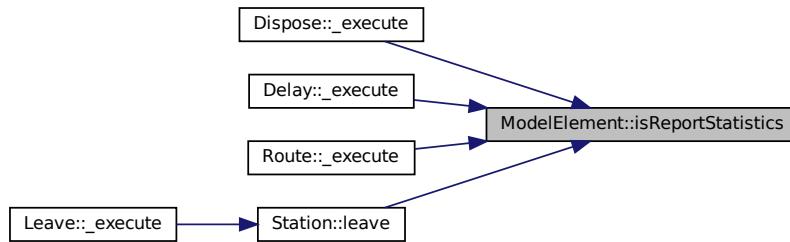
00225
00226     return _reportStatistics;
00227 }

```

References [_reportStatistics](#).

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Route::_execute\(\)](#), and [Station::leave\(\)](#).

Here is the caller graph for this function:



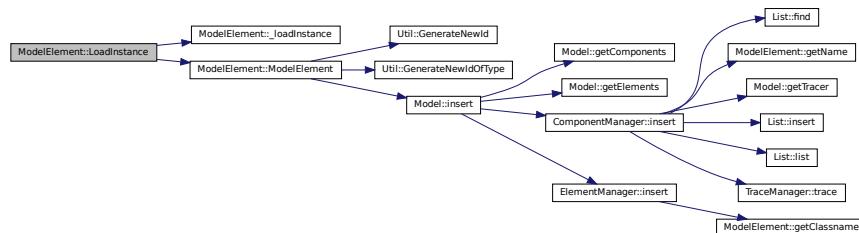
```
8.73.3.19 LoadInstance() ModelElement * ModelElement::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields,
    bool insertIntoModel ) [static]
```

Definition at line 162 of file [ModelElement.cpp](#).

```
00162
00163     {
00164         std::string name = "";
00165         if (insertIntoModel) {
00166             // extracts the name from the fields even before "_laodInstance" and even before construct a
00167             // new ModelElement in such way when constructing the ModelElement, it's done with the correct name and
00168             // that correct name is show in trace
00169             std::map<std::string, std::string>::iterator it = fields->find("name");
00170             if (it != fields->end())
00171                 name = (*it).second;
00172         }
00173         ModelElement* newElement = new ModelElement(model, "ModelElement", name, insertIntoModel);
00174         try {
00175             newElement->_loadInstance(fields);
00176         } catch (const std::exception& e) {
00177         }
00178         return newElement;
00179     }
```

References [_loadInstance\(\)](#), and [ModelElement\(\)](#).

Here is the call graph for this function:



```
8.73.3.20 SaveInstance() std::map< std::string, std::string > * ModelElement::SaveInstance (
    ModelElement * element ) [static]
```

Definition at line 179 of file [ModelElement.cpp](#).

```
00179
00180     std::map<std::string, std::string>* fields; // = new std::list<std::string>();
00181     try {
00182         fields = element->_saveInstance();
00183     } catch (const std::exception& e) {
00184         //element->_model->getTrace()->traceError(e, "Error saving anElement " + element->show());
00185     }
00186     return fields;
00187 }
```

References [_saveInstance\(\)](#).

Here is the call graph for this function:



8.73.3.21 setName() void ModelElement::setName (std::string _name)

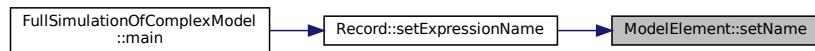
Definition at line 141 of file ModelElement.cpp.

```
00141
00142     this->_name = _name;
00143 }
```

References [_name](#).

Referenced by [Record::setExpressionName\(\)](#).

Here is the caller graph for this function:



8.73.3.22 setReportStatistics() void ModelElement::setReportStatistics (bool reportStatistics)

Definition at line 221 of file ModelElement.cpp.

```
00221
00222     this->_reportStatistics = reportStatistics;
00223 }
```

References [_reportStatistics](#).

8.73.3.23 show() std::string ModelElement::show () [virtual]

Reimplemented in [Seize](#), [Enter](#), [Queue](#), [Leave](#), [Resource](#), [Assign](#), [Variable](#), [Sequence](#), [Decide](#), [Entity](#), [Hold](#), [Schedule](#), [Failure](#), [Record](#), [Create](#), [File](#), [Release](#), [Station](#), [Attribute](#), [PickStation](#), [Route](#), [Set](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Delay](#), [Match](#), [Unstore](#), [SourceModelComponent](#), [Write](#), [DropOff](#), [Storage](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [OLD_ODElement](#), [ModelComponent](#), [EntityType](#), [StatisticsCollector](#), [Counter](#), [EntityGroup](#), [LSODE](#), [Dummy](#), [Formula](#), [Submodel](#), and [MarkovChain](#).

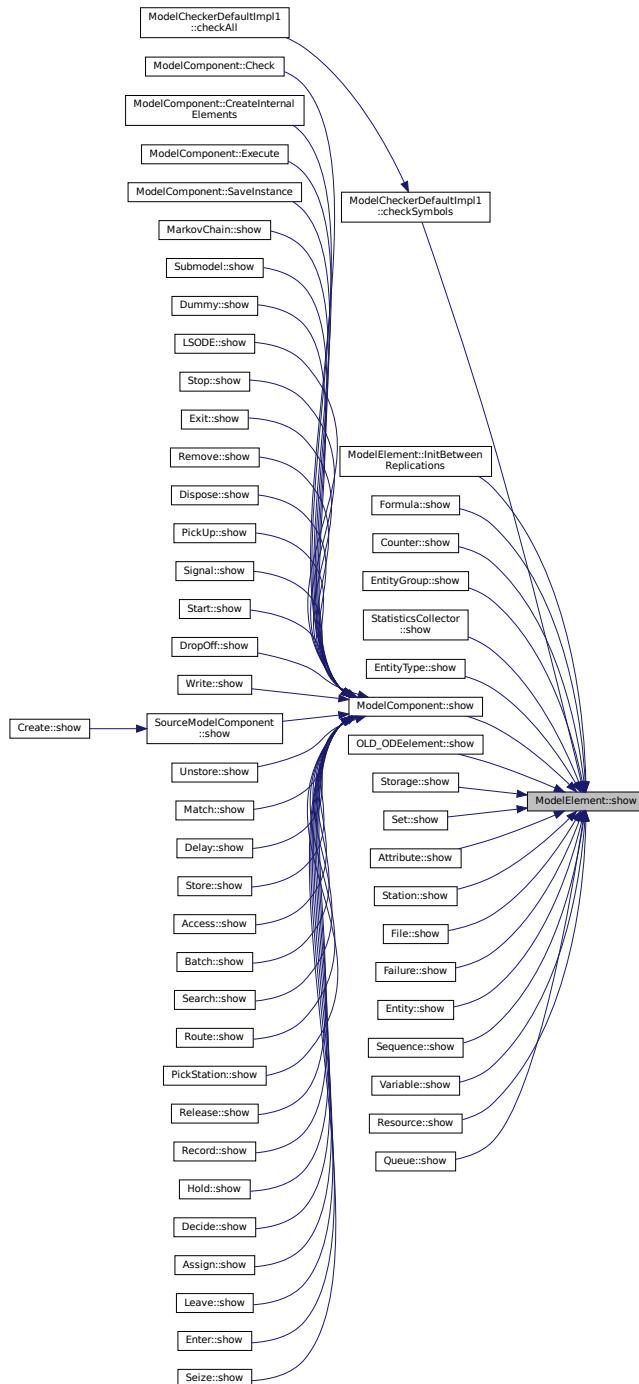
Definition at line 120 of file ModelElement.cpp.

```
00120
00121     std::string children = "";
00122     if (_childrenElements->size() > 0) {
00123         children = ", children=[";
00124         for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it != _childrenElements->end(); it++) {
00125             children += (*it).second->getName() + ",";
00126         }
00127         children = children.substr(0, children.length() - 1) + "]";
00128     }
00129     return "id=" + std::to_string(_id) + ",name=\"\" + _name + \"\" + children;
00130 }
```

References [_childrenElements](#), [_id](#), and [_name](#).

Referenced by [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [InitBetweenReplications\(\)](#), [Formula::show\(\)](#), [Counter::show\(\)](#), [EntityGroup::show\(\)](#), [StatisticsCollector::show\(\)](#), [EntityType::show\(\)](#), [ModelComponent::show\(\)](#), [OLD_ODElement::show\(\)](#), [Storage::show\(\)](#), [Set::show\(\)](#), [Attribute::show\(\)](#), [Station::show\(\)](#), [File::show\(\)](#), [Failure::show\(\)](#), [Entity::show\(\)](#), [Sequence::show\(\)](#), [Variable::show\(\)](#), [Resource::show\(\)](#), and [Queue::show\(\)](#).

Here is the caller graph for this function:



8.73.4 Member Data Documentation

8.73.4.1 `_childrenElements` `std::map<std::string, ModelElement*>* ModelElement::_children← Elements = new std::map<std::string, ModelElement*>() [protected]`

Definition at line 78 of file [ModelElement.h](#).

Referenced by [EntityGroup::createInternalElements\(\)](#), [Dispose::createInternalElements\(\)](#), [Delay::createInternalElements\(\)](#), [Create::createInternalElements\(\)](#), [Route::createInternalElements\(\)](#), [Station::createInternalElements\(\)](#), [Leave::createInternalElements\(\)](#), [Enter::createInternalElements\(\)](#), [Queue::createInternalElements\(\)](#), [Resource::createInternalElements\(\)](#), [_removeChildElement\(\)](#), [_removeChildrenElements\(\)](#), [EntityType::addGetStatisticsCollector\(\)](#), and [show\(\)](#).

8.73.4.2 `_id` `Util::identification ModelElement::_id [protected]`

Definition at line 72 of file [ModelElement.h](#).

Referenced by [_loadInstance\(\)](#), [_saveInstance\(\)](#), [Station::enter\(\)](#), [getId\(\)](#), [ModelElement\(\)](#), and [show\(\)](#).

8.73.4.3 `_name` `std::string ModelElement::_name [protected]`

Definition at line 73 of file [ModelElement.h](#).

Referenced by [StatisticsCollector::addSimulationResponses\(\)](#), [Dispose::createInternalElements\(\)](#), [Delay::createInternalElements\(\)](#), [Create::createInternalElements\(\)](#), [Route::createInternalElements\(\)](#), [Station::createInternalElements\(\)](#), [Leave::createInternalElements\(\)](#), [Enter::createInternalElements\(\)](#), [Queue::createInternalElements\(\)](#), [Resource::createInternalElements\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [ModelComponent::Check\(\)](#), [Counter::Counter\(\)](#), [Create::Create\(\)](#), [Delay::Delay\(\)](#), [ModelComponent::Execute\(\)](#), [getName\(\)](#), [ModelElement\(\)](#), [Resource::Resource\(\)](#), [ModelComponent::SaveInstance\(\)](#), [setName\(\)](#), [show\(\)](#), [Variable::Variable\(\)](#), and [~ModelElement\(\)](#).

8.73.4.4 `_parentModel` `Model* ModelElement::_parentModel [protected]`

Definition at line 76 of file [ModelElement.h](#).

Referenced by [StatisticsCollector::addSimulationResponses\(\)](#), [Formula::check\(\)](#), [EntityGroup::check\(\)](#), [SourceModelComponent::check\(\)](#), [OLD_ODElement::check\(\)](#), [Delay::check\(\)](#), [Write::check\(\)](#), [Route::check\(\)](#), [Station::check\(\)](#), [Release::check\(\)](#), [Record::check\(\)](#), [Decide::check\(\)](#), [Sequence::check\(\)](#), [Assign::check\(\)](#), [Leave::check\(\)](#), [Enter::check\(\)](#), [Queue::check\(\)](#), [Seize::check\(\)](#), [EntityGroup::createInternalElements\(\)](#), [Dispose::createInternalElements\(\)](#), [EntityType::createInternalElements\(\)](#), [Delay::createInternalElements\(\)](#), [Create::createInternalElements\(\)](#), [Route::createInternalElements\(\)](#), [Station::createInternalElements\(\)](#), [Leave::createInternalElements\(\)](#), [Enter::createInternalElements\(\)](#), [Queue::createInternalElements\(\)](#), [Resource::createInternalElements\(\)](#), [Dummy::execute\(\)](#), [Submodel::execute\(\)](#), [MarkovChain::execute\(\)](#), [LSODE::execute\(\)](#), [Stop::execute\(\)](#), [Remove::execute\(\)](#), [Exit::execute\(\)](#), [Signal::execute\(\)](#), [PickUp::execute\(\)](#), [Dispose::execute\(\)](#), [Start::execute\(\)](#), [DropOff::execute\(\)](#), [Unstore::execute\(\)](#), [Match::execute\(\)](#), [Delay::execute\(\)](#), [Store::execute\(\)](#), [Write::execute\(\)](#), [Batch::execute\(\)](#), [Access::execute\(\)](#), [Search::execute\(\)](#), [PickStation::execute\(\)](#), [Create::execute\(\)](#), [Route::execute\(\)](#), [Record::execute\(\)](#), [Release::execute\(\)](#), [Hold::execute\(\)](#), [Decide::execute\(\)](#), [Assign::execute\(\)](#), [Leave::execute\(\)](#), [Enter::execute\(\)](#), [Seize::execute\(\)](#), [Write::initBetweenReplications\(\)](#), [SourceModelComponent::loadInstance\(\)](#), [Route::loadInstance\(\)](#), [Release::loadInstance\(\)](#), [Leave::loadInstance\(\)](#), [Enter::loadInstance\(\)](#), [Seize::loadInstance\(\)](#), [_removeChildElement\(\)](#), [_removeChildrenElements\(\)](#), [EntityType::addGetStatisticsCollector\(\)](#), [ModelComponent::Check\(\)](#), [Counter::Counter\(\)](#), [ModelComponent::CreateInternalElements\(\)](#), [Delay::delay\(\)](#), [Station::enter\(\)](#), [Entity::Entity\(\)](#), [ModelComponent::Execute\(\)](#), [Entity::getAttributeValue\(\)](#), [InitBetweenReplications\(\)](#), [Station::leave\(\)](#), [ModelElement\(\)](#), [Record::Record\(\)](#), [Queue::removeElement\(\)](#), [ModelComponent::SaveInstance\(\)](#), [Entity::setAttributeValue\(\)](#), [Entity::show\(\)](#), [Formula::value\(\)](#), [EntityType::~EntityType\(\)](#), [ModelComponent::~ModelComponent\(\)](#), [~ModelElement\(\)](#), and [Record::~Record\(\)](#).

8.73.4.5 `_reportStatistics` `bool ModelElement::_reportStatistics [protected]`

Definition at line 75 of file [ModelElement.h](#).

Referenced by [EntityGroup::_createInternalElements\(\)](#), [Dispose::_createInternalElements\(\)](#), [EntityType::_createInternalElements\(\)](#), [Delay::_createInternalElements\(\)](#), [Create::_createInternalElements\(\)](#), [Route::_createInternalElements\(\)](#), [Station::_createInternalElements\(\)](#), [Leave::_createInternalElements\(\)](#), [Enter::_createInternalElements\(\)](#), [Queue::_createInternalElements\(\)](#), [Resource::_createInternalElements\(\)](#), [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Leave::_execute\(\)](#), [Enter::_execute\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [Station::enter\(\)](#), [Resource::initBetweenReplications\(\)](#), [Queue::insertElement\(\)](#), [isReportStatistics\(\)](#), [Station::leave\(\)](#), [ModelElement\(\)](#), [Resource::release\(\)](#), [Queue::removeElement\(\)](#), [Resource::seize\(\)](#), and [setReportStatistics\(\)](#).

8.73.4.6 `_typename` `std::string ModelElement::_typename [protected]`

Definition at line 74 of file [ModelElement.h](#).

Referenced by [_saveInstance\(\)](#), [ModelComponent::Check\(\)](#), [getClassName\(\)](#), and [ModelElement\(\)](#).

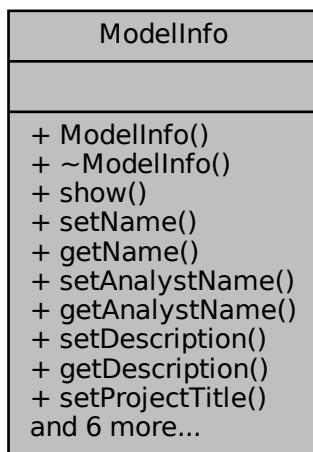
The documentation for this class was generated from the following files:

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

8.74 ModellInfo Class Reference

```
#include <ModellInfo.h>
```

Collaboration diagram for ModellInfo:



Public Member Functions

- `ModelInfo()`
- `virtual ~ModelInfo() = default`
- `std::string show()`
- `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 getProjectTitle() const`
- `void setVersion(std::string _version)`
- `std::string getVersion() const`
- `void loadInstance(std::map< std::string, std::string > *fields)`
- `std::map< std::string, std::string > * saveInstance()`
- `bool hasChanged() const`

8.74.1 Detailed Description

`ModelInfo` stores basic model project information.

Definition at line 25 of file `ModelInfo.h`.

8.74.2 Constructor & Destructor Documentation

8.74.2.1 `ModelInfo()` `ModelInfo::ModelInfo()`

Definition at line 19 of file `ModelInfo.cpp`.

```
00019      {
00020      _name = "Model " + std::to_string(Util::GenerateNewIdOfType<Model>());
00021 }
```

8.74.2.2 `~ModelInfo()` `virtual ModelInfo::~ModelInfo() [virtual], [default]`

8.74.3 Member Function Documentation

8.74.3.1 `getAnalystName()` `std::string ModelInfo::getAnalystName() const`

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

```
00044      {
00045      return _analystName;
00046 }
```

8.74.3.2 `getDescription()` `std::string ModelInfo::getDescription () const`

Definition at line 53 of file `ModelInfo.cpp`.

```
00053     {  
00054         return _description;  
00055     }
```

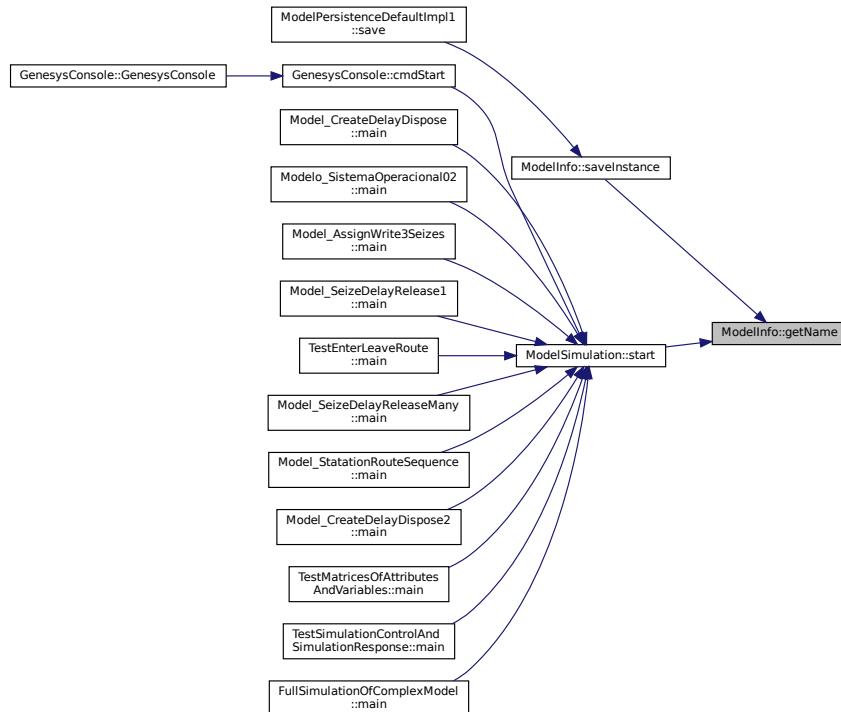
8.74.3.3 `getName()` `std::string ModelInfo::getName () const`

Definition at line 35 of file `ModelInfo.cpp`.

```
00035     {  
00036         return _name;  
00037     }
```

Referenced by `saveInstance()`, and `ModelSimulation::start()`.

Here is the caller graph for this function:



8.74.3.4 `getProjectTitle()` `std::string ModelInfo::getProjectTitle () const`

Definition at line 62 of file `ModelInfo.cpp`.

```
00062     {  
00063         return _projectTitle;  
00064     }
```

8.74.3.5 getVersion() std::string ModelInfo::getVersion () const

Definition at line 71 of file [ModelInfo.cpp](#).

```
00071     {  
00072         return _version;  
00073     }
```

8.74.3.6 hasChanged() bool ModelInfo::hasChanged () const

Definition at line 98 of file [ModelInfo.cpp](#).

```
00098     {  
00099         return _hasChanged;  
00100     }
```

Referenced by [Model::hasChanged\(\)](#).

Here is the caller graph for this function:

**8.74.3.7 loadInstance()** void ModelInfo::loadInstance (std::map< std::string, std::string * > * fields)

Definition at line 75 of file [ModelInfo.cpp](#).

```
00075     {  
00076         this->_analystName = loadField(fields, "analystName", "");  
00077         this->_description = loadField(fields, "description", "");  
00078         this->_name = loadField(fields, "name", "");  
00079         this->_projectTitle = loadField(fields, "projectTitle", "");  
00080         this->_version = loadField(fields, "version", "1.0");  
00081         _hasChanged = false;  
00082     }
```

References [loadField\(\)](#).

Here is the call graph for this function:



8.74.3.8 **saveInstance()** std::map< std::string, std::string > * ModelInfo::saveInstance ()

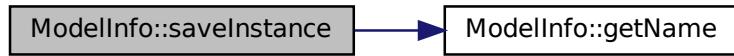
Definition at line 86 of file [ModelInfo.cpp](#).

```
00086     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00087     fields->emplace("typename", "ModelInfo");
00088     if (_analystName != "") fields->emplace("analystName", "\"" + _analystName + "\"");
00089     if (_description != "") fields->emplace("description", "\"" + _description + "\"");
00090     fields->emplace("name", "\"" + getName\(\) + "\"");
00091     if (_projectTitle != "") fields->emplace("projectTitle", "\"" + _projectTitle + "\"");
00092     if (_version != "1.0") fields->emplace("version", "\"" + _version + "\"");
00093     \_hasChanged = false;
00094     return fields;
00095 }
```

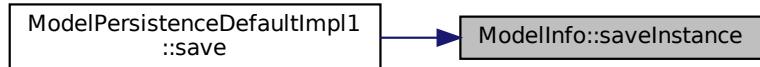
References [getName\(\)](#).

Referenced by [ModelPersistenceDefaultImpl1::save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



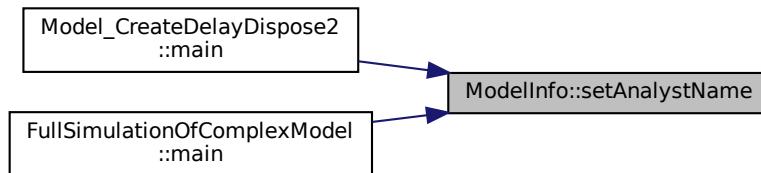
8.74.3.9 **setAnalystName()** void ModelInfo::setAnalystName (std::string _analystName)

Definition at line 39 of file [ModelInfo.cpp](#).

```
00039
00040     this->_analystName = _analystName;
00041     \_hasChanged = true;
00042 }
```

Referenced by [Model_CreateDelayDispose2::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.74.3.10 setDescription() void ModelInfo::setDescription (std::string _description)

Definition at line 48 of file [ModelInfo.cpp](#).

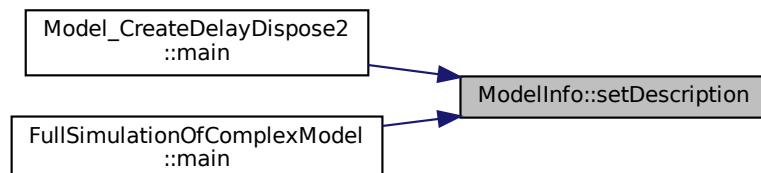
```

00048
00049     this->_description = _description;
00050     _hasChanged = true;
00051 }

```

Referenced by [Model_CreateDelayDispose2::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.74.3.11 setName() void ModelInfo::setName (std::string _name)

Definition at line 30 of file [ModelInfo.cpp](#).

```

00030
00031     this->_name = _name;
00032     _hasChanged = true;
00033 }

```

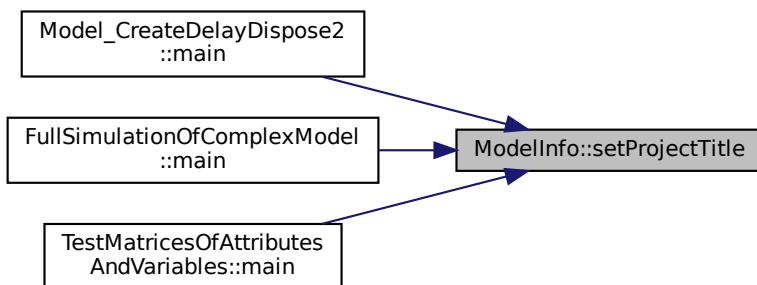
8.74.3.12 setProjectTitle() void ModelInfo::setProjectTitle (std::string _projectTitle)

Definition at line 57 of file [ModelInfo.cpp](#).

```
00057
00058     this->_projectTitle = _projectTitle;
00059     _hasChanged = true;
00060 }
```

Referenced by [Model_CreateDelayDispose2::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.74.3.13 setVersion() void ModelInfo::setVersion (std::string _version)

Definition at line 66 of file [ModelInfo.cpp](#).

```
00066
00067     this->_version = _version;
00068     _hasChanged = true;
00069 }
```

8.74.3.14 show() std::string ModelInfo::show ()

Definition at line 23 of file [ModelInfo.cpp](#).

```
00023
00024     return "analystName=\"" + this->_analystName + "\" +
00025         ",description=\"" + this->_description + "\" +
00026         ",name=\"" + this->_name + "\" +
00027         ",version=\"" + this->_version;
00028 }
```

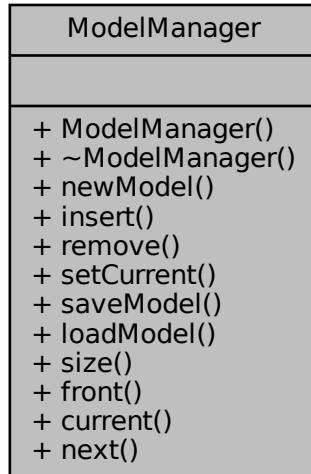
The documentation for this class was generated from the following files:

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

8.75 ModelManager Class Reference

```
#include <ModelManager.h>
```

Collaboration diagram for ModelManager:



Public Member Functions

- `ModelManager (Simulator *simulator)`
- virtual `~ModelManager ()`=default
- `Model * newModel ()`
- `void insert (Model *model)`
- `void remove (Model *model)`
- `void setCurrent (Model *model)`
- `bool saveModel (std::string filename)`
- `bool loadModel (std::string filename)`
- `unsigned int size ()`
- `Model * front ()`
- `Model * current ()`
- `Model * next ()`

8.75.1 Detailed Description

Definition at line 22 of file [ModelManager.h](#).

8.75.2 Constructor & Destructor Documentation

8.75.2.1 ModelManager() `ModelManager::ModelManager (Simulator * simulator)`

Definition at line 20 of file [ModelManager.cpp](#).

```
00020
00021     _simulator = simulator;
00022     _currentModel = nullptr;
00023 }
```

8.75.2.2 ~ModelManager() `virtual ModelManager::~ModelManager () [virtual], [default]`

8.75.3 Member Function Documentation

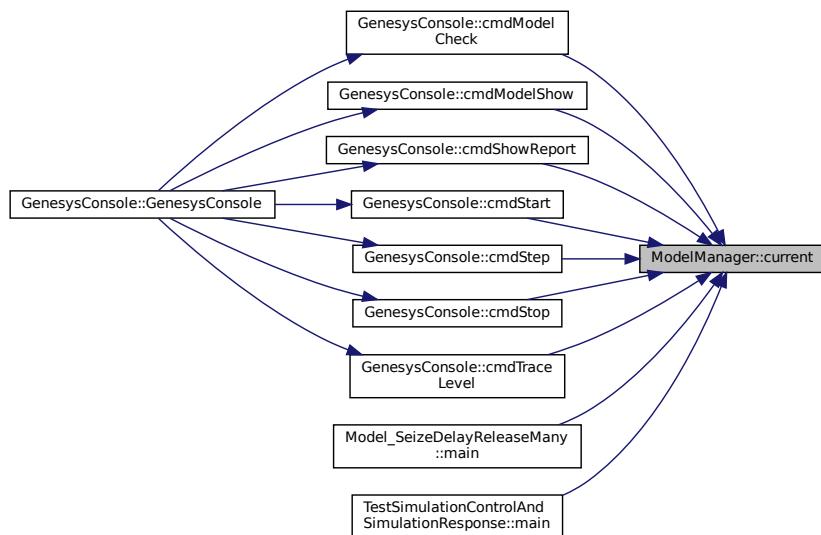
8.75.3.1 current() `Model * ModelManager::current ()`

Definition at line 72 of file [ModelManager.cpp](#).

```
00072
00073     return _currentModel;
00074 }
```

Referenced by [GenesysConsole::cmdModelCheck\(\)](#), [GenesysConsole::cmdModelShow\(\)](#), [GenesysConsole::cmdShowReport\(\)](#), [GenesysConsole::cmdStart\(\)](#), [GenesysConsole::cmdStep\(\)](#), [GenesysConsole::cmdStop\(\)](#), [GenesysConsole::cmdTraceLevel\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), and [TestSimulationControlAndSimulationResponse::main\(\)](#).

Here is the caller graph for this function:



8.75.3.2 front() `Model * ModelManager::front ()`

Definition at line 76 of file [ModelManager.cpp](#).

```
00076     {
00077         return _models->front ();
00078     }
```

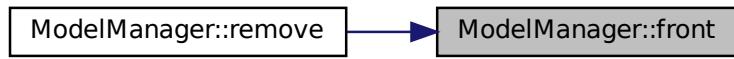
References [List< T >::front\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:

**8.75.3.3 insert()** `void ModelManager::insert (Model * model)`

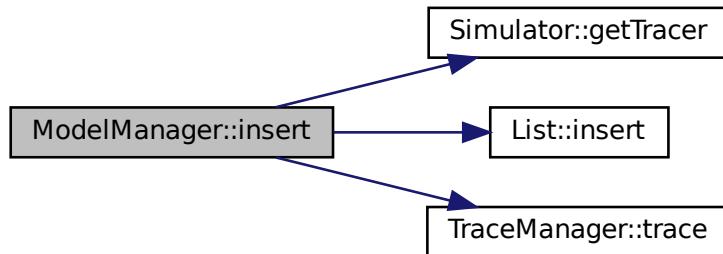
Definition at line 30 of file [ModelManager.cpp](#).

```
00030     {
00031         _models->insert (model);
00032         this->_currentModel = model;
00033         _simulator->getTracer()->trace (Util::TraceLevel::simulatorResult, "Model successfully inserted");
00034     }
```

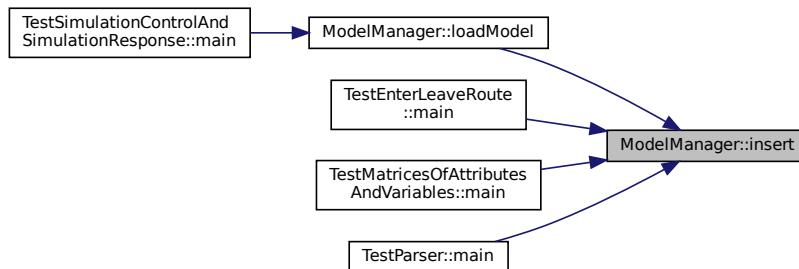
References [Simulator::getTracer\(\)](#), [List< T >::insert\(\)](#), [Util::simulatorResult](#), and [TraceManager::trace\(\)](#).

Referenced by [loadModel\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), and [TestParser::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.75.3.4 `loadModel()` `bool ModelManager::loadModel (std::string filename)`

Definition at line 55 of file `ModelManager.cpp`.

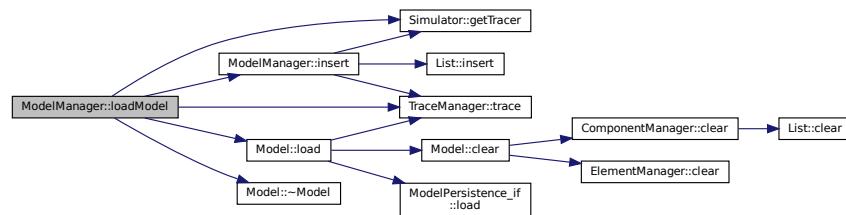
```

00055
00056     Model* model = new Model(_simulator);
00057     bool res = model->load(filename);
00058     if (res) {
00059         this->insert(model);
00060         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model successfully
loaded");
00061     } else {
00062         model->~Model();
00063         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model could not be loaded");
00064     }
00065     return res;
00066 }
```

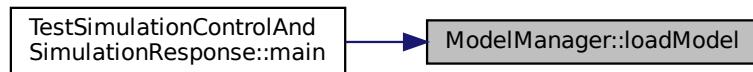
References `Simulator::getTracer()`, `insert()`, `Model::load()`, `Util::simulatorResult`, `TraceManager::trace()`, and `Model::~Model()`.

Referenced by `TestSimulationControlAndSimulationResponse::main()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.75.3.5 newModel() `Model * ModelManager::newModel()`

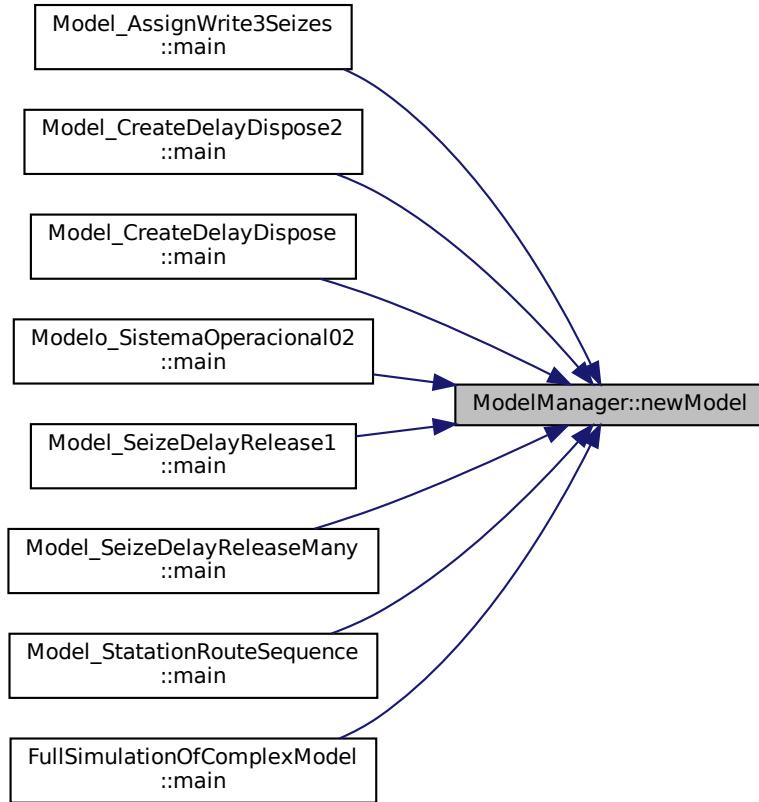
Definition at line 25 of file `ModelManager.cpp`.

```

00025     {
00026         _currentModel = new Model(_simulator);
00027         return _currentModel;
00028     }
  
```

Referenced by `Model_AssignWrite3Seizes::main()`, `Model_CreateDelayDispose2::main()`, `Model_StatationRouteSequence::main()`, `Model_SeizeDelayReleaseMany::main()`, `Model_SeizeDelayRelease1::main()`, `Modelo_SistemaOperacional02::main()`, `Model_CreateDelayDispose::main()`, and `FullSimulationOfComplexModel::main()`.

Here is the caller graph for this function:



8.75.3.6 `next()` `Model * ModelManager::next ()`

Definition at line 80 of file [ModelManager.cpp](#).

```

00080
00081     return _models->next\(\);
00082 }
```

References [List< T >::next\(\)](#).

Here is the call graph for this function:



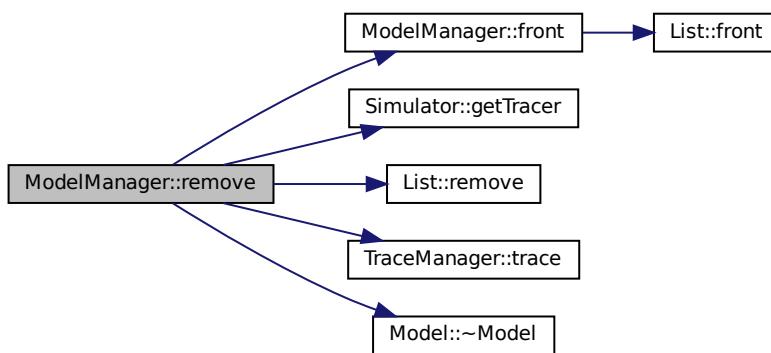
8.75.3.7 remove() void ModelManager::remove (
 Model * model)

Definition at line 36 of file ModelManager.cpp.

```
00036     {
00037         _models->remove(model);
00038         if (_currentModel == model) {
00039             _currentModel = this->front();
00040         }
00041         model->~Model();
00042         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model successfully removed");
00043     }
```

References [front\(\)](#), [Simulator::getTracer\(\)](#), [List< T >::remove\(\)](#), [Util::simulatorResult](#), [TraceManager::trace\(\)](#), and [Model::~Model\(\)](#).

Here is the call graph for this function:



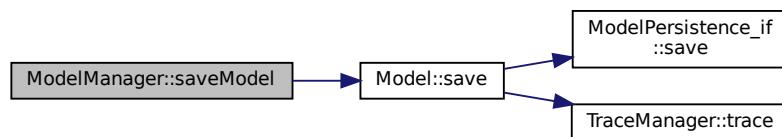
8.75.3.8 saveModel() bool ModelManager::saveModel (std::string filename)

Definition at line 49 of file ModelManager.cpp.

```
00049     {
00050         if (_currentModel != nullptr)
00051             return _currentModel->save(filename);
00052         return false;
00053     }
```

References [Model::save\(\)](#).

Here is the call graph for this function:



8.75.3.9 setCurrent() void ModelManager::setCurrent (Model * model)

Definition at line 68 of file [ModelManager.cpp](#).

```
00068 {  
00069     this->_currentModel = model;  
00070 }
```

8.75.3.10 size() unsigned int ModelManager::size ()

Definition at line 45 of file [ModelManager.cpp](#).

```
00045 {  
00046     return _models->size();  
00047 }
```

References [List< T >::size\(\)](#).

Here is the call graph for this function:



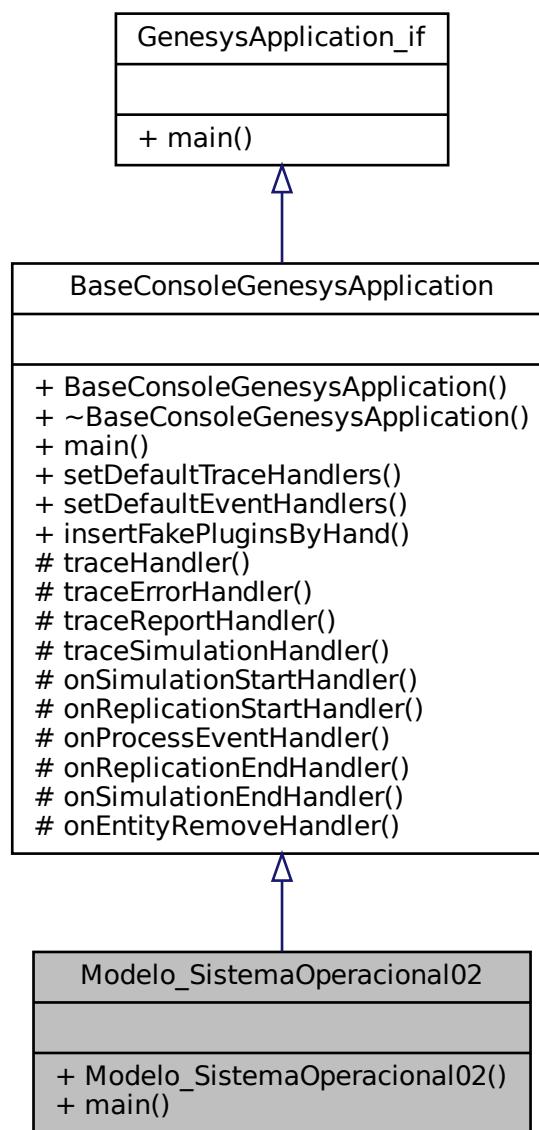
The documentation for this class was generated from the following files:

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

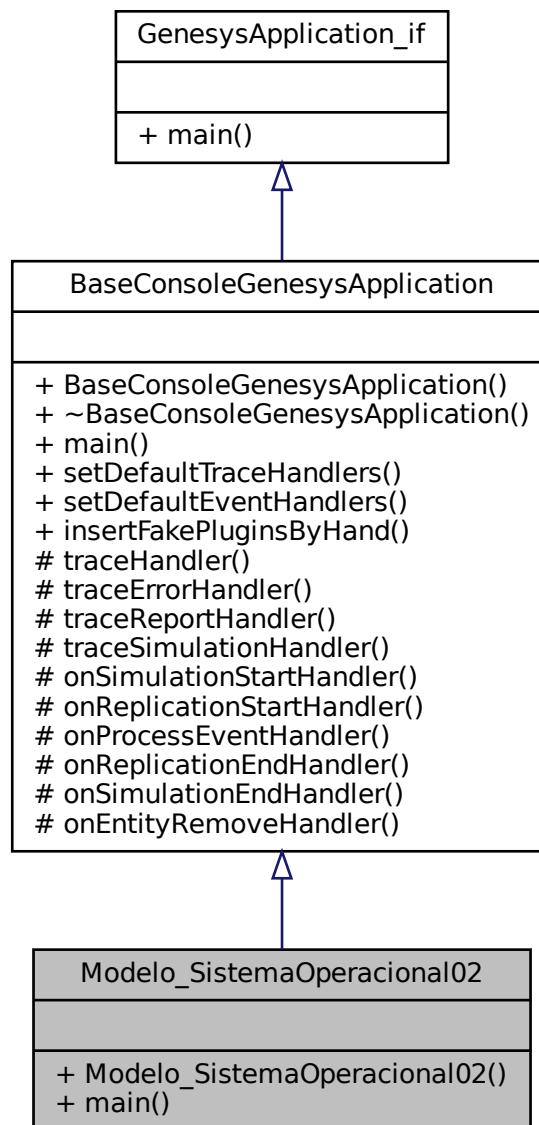
8.76 Modelo_SistemaOperacional02 Class Reference

```
#include <Modelo_SistemaOperacional02.h>
```

Inheritance diagram for Modelo_SistemaOperacional02:



Collaboration diagram for Modelo_SistemaOperacional02:



Public Member Functions

- `Modelo_SistemaOperacional02 ()`
- virtual int `main (int argc, char **argv)`

Additional Inherited Members

8.76.1 Detailed Description

Definition at line 19 of file [Modelo_SistemaOperacional02.h](#).

8.76.2 Constructor & Destructor Documentation

8.76.2.1 Modelo_SistemaOperacional02() [Modelo_SistemaOperacional02::Modelo_SistemaOperacional02](#)

```
( )
```

Definition at line 33 of file [Modelo_SistemaOperacional02.cpp](#).

```
00033 {  
00034 }
```

8.76.3 Member Function Documentation

8.76.3.1 main() [int Modelo_SistemaOperacional02::main](#) (

```
    int argc,  
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 36 of file [Modelo_SistemaOperacional02.cpp](#).

```
00036 {  
00037     Simulator* genesys = new Simulator();  
00038     this->setDefaultTraceHandlers(genesys->getTracer());  
00039     this->insertFakePluginsByHand(genesys);  
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);  
00041  
00042     Model* m = genesys->getModels()->newModel();  
00043     m->load("./models/SistemaOperacional02.txt");  
00044     m->getSimulation()->start();  
00045     return;  
00046     EntityType* et = new EntityType(m, "processo");  
00047     Create* cl = new Create(m, "Processo é criado no computador");  
00048     cl->setEntityType(et);  
00049     cl-> setTimeBetweenCreationsExpression("expo(10)");  
00050     cl-> setTimeUnit(Util::TimeUnit::milisecond);  
00051     Assign* al = new Assign(m, "Define tempo de execução e memória ocupada");  
00052     al->getAssignments()->insert(new Assign::Assignment("memoriaOcupada", "TRUNC(UNIF(10,50))"));  
00053     al->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "NORM(3,1) *  
memoriaOcupada/10"));  
00054     Attribute* att1 = new Attribute(m, "memoriaOcupada");  
00055     Attribute* att2 = new Attribute(m, "tempoExecucao");  
00056     cl->getNextComponents()->insert(al);  
00057     Resource* mem = new Resource(m, "memoria");  
00058     mem->setCapacity(64);  
00059     Queue* q1 = new Queue(m, "Fila_Alocacao_Memoria");  
00060     Seize* sz1 = new Seize(m, "Processo aloca memória");  
00061     al->getNextComponents()->insert(sz1);  
00062     sz1->setQueue(q1);  
00063     sz1->getSeizeRequests()->insert(new SeizableItemRequest(mem, "memoriaOcupada"));  
00064     Station* st1 = new Station(m, "Estação de Execução");  
00065     Route* rl = new Route(m, "Processo é enviado para execução na CPU");  
00066     rl->setStation(st1);  
00067     sz1->getNextComponents()->insert(rl);  
00068  
00069     Enter* el = new Enter(m, "Processo chega para ser executado");  
00070     el->setStation(st1);  
00071     Resource* cpu = new Resource(m, "CPU");  
00072     Queue* q2 = new Queue(m, "Fila_CPU");  
00073     Seize* sz2 = new Seize(m, "Processo aloca CPU");  
00074     sz2->setQueue(q2);  
00075     sz2->getSeizeRequests()->insert(new SeizableItemRequest(cpu, "1"));  
00076     el->getNextComponents()->insert(sz2);  
00077     Decide* dec1 = new Decide(m, "Define tempo de execução da fatia de tempo atual");  
00078     dec1->getConditions()->insert("tempoExecucao >= 1");  
00079     sz2->getNextComponents()->insert(dec1);  
00080     Assign* a2 = new Assign(m, "Define execução por um quantum de tempo");  
00081     a2->getAssignments()->insert(new Assign::Assignment("fatiaTempo", "1"));  
00082     a2->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "tempoExecucao-fatiaTempo"));  
00083     Attribute* att3 = new Attribute(m, "fatiaTempo");
```

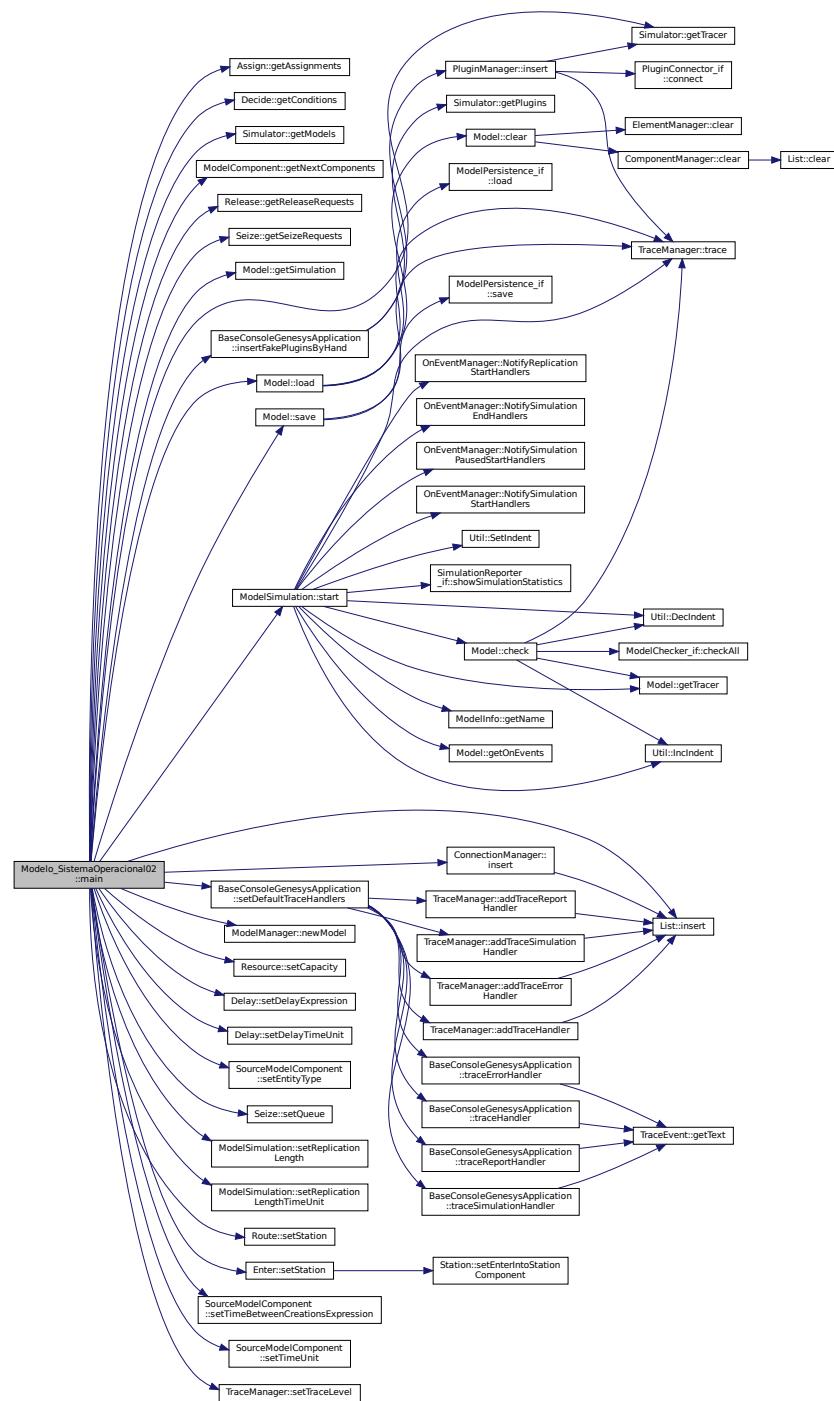
```

00084     decl->getNextComponents()->insert(a2);
00085     Assign* a3 = new Assign(m, "Executa até o final");
00086     a3->getAssignments()->insert(new Assign::Assignment("fatiaTempo", "tempoExecucao"));
00087     a3->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "tempoExecucao-fatiaTempo"));
00088     decl->getNextComponents()->insert(a3);
00089     Delay* dl1 = new Delay(m, "Processo executa na CPU");
00090     dl1->setDelayExpression("fatiaTempo");
00091     dl1->setDelayTimeUnit(Util::TimeUnit::milisecond);
00092     a2->getNextComponents()->insert(dl1);
00093     a3->getNextComponents()->insert(dl1);
00094     Release* r11 = new Release(m, "Processo libera CPU");
00095     r11->getReleaseRequests()->insert(new SeizableItemRequest(cpu, "1"));
00096     dl1->getNextComponents()->insert(r11);
00097     Decide* dc2 = new Decide(m, "Se processo ainda precisa executar então vai para estação de
execução");
00098     dc2->getConditions()->insert("tempoExecucao > 0");
00099     r11->getNextComponents()->insert(dc2);
00100     Route* r2 = new Route(m, "Processo é enviado de volta para execução");
00101     r2->setStation(st1);
00102     dc2->getNextComponents()->insert(r2);
00103     Station* st2 = new Station(m, "Estacao de liberação de memória");
00104     Route* r3 = new Route(m, "Processo é enviado para liberar memória");
00105     r3->setStation(st2);
00106     dc2->getNextComponents()->insert(r3);
00107
00108     Enter* e2 = new Enter(m, "Processo chega para liberar memória");
00109     e2->setStation(st2);
00110     Release* r12 = new Release(m, "Processo libera memória");
00111     r12->getReleaseRequests()->insert(new SeizableItemRequest(mem, "memoriaOcupada"));
00112     e2->getNextComponents()->insert(r12);
00113     Dispose* disp1 = new Dispose(m, "Processo é encerrado");
00114     r12->getNextComponents()->insert(disp1);
00115
00116     ModelSimulation* sim = m->getSimulation();
00117     sim->setReplicationLength(1);
00118     sim->setReplicationLengthTimeUnit(Util::TimeUnit::second);
00119     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00120     m->save("./models/SistemaOperacional02.txt");
00121     sim->start();
00122 }

```

References [Util::everythingMostDetailed](#), [Assign::getAssignments\(\)](#), [Decide::getConditions\(\)](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Release::getReleaseRequests\(\)](#), [Seize::getSeizeRequests\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [ConnectionManager::insert\(\)](#), [List< T >::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [Model::load\(\)](#), [Util::milisecond](#), [ModelManager::newModel\(\)](#), [Model::save\(\)](#), [Util::second](#), [Resource::setCapacity\(\)](#), [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#), [Delay::setDelayExpression\(\)](#), [Delay::setDelayTimeUnit\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [Seize::setQueue\(\)](#), [ModelSimulation::setReplicationLength\(\)](#), [ModelSimulation::setReplicationLengthTimeUnit\(\)](#), [Route::setStation\(\)](#), [Enter::setStation\(\)](#), [SourceModelComponent::setTimeBetweenCreationsExpression\(\)](#), [SourceModelComponent::setTimeUnit\(\)](#), [TraceManager::setTraceLevel\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the call graph for this function:



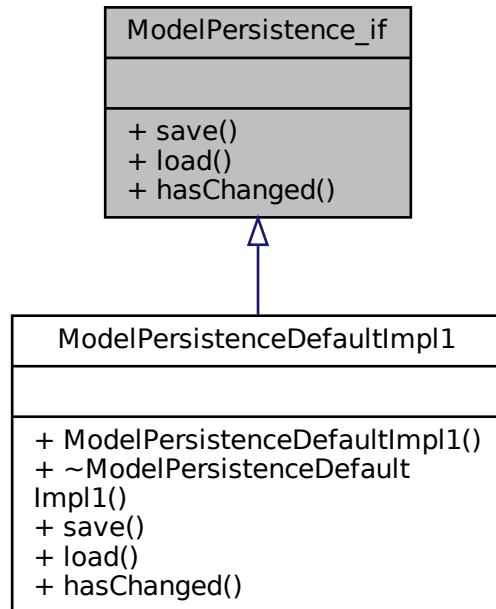
The documentation for this class was generated from the following files:

- `Modelo_SistemaOperacional02.h`
 - `Modelo_SistemaOperacional02.cpp`

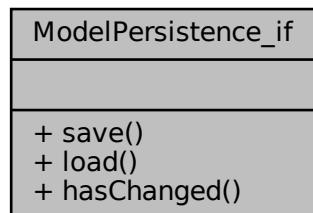
8.77 ModelPersistence_if Class Reference

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

Inheritance diagram for ModelPersistence_if:



Collaboration diagram for ModelPersistence_if:



Public Member Functions

- virtual bool `save` (std::string filename)=0
- virtual bool `load` (std::string filename)=0
- virtual bool `hasChanged` ()=0

8.77.1 Detailed Description

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

Definition at line 22 of file [ModelPersistence_if.h](#).

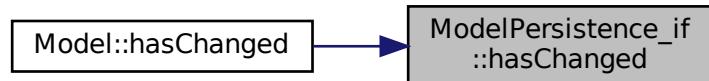
8.77.2 Member Function Documentation

8.77.2.1 hasChanged() virtual bool ModelPersistence_if::hasChanged () [pure virtual]

Implemented in [ModelPersistenceDefaultImpl1](#).

Referenced by [Model::hasChanged\(\)](#).

Here is the caller graph for this function:

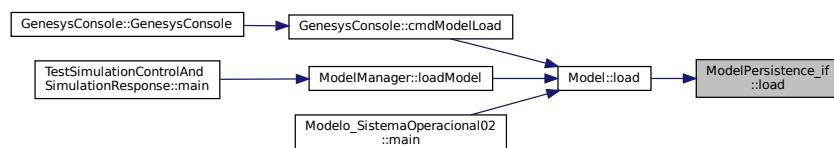


8.77.2.2 load() virtual bool ModelPersistence_if::load (std::string filename) [pure virtual]

Implemented in [ModelPersistenceDefaultImpl1](#).

Referenced by [Model::load\(\)](#).

Here is the caller graph for this function:

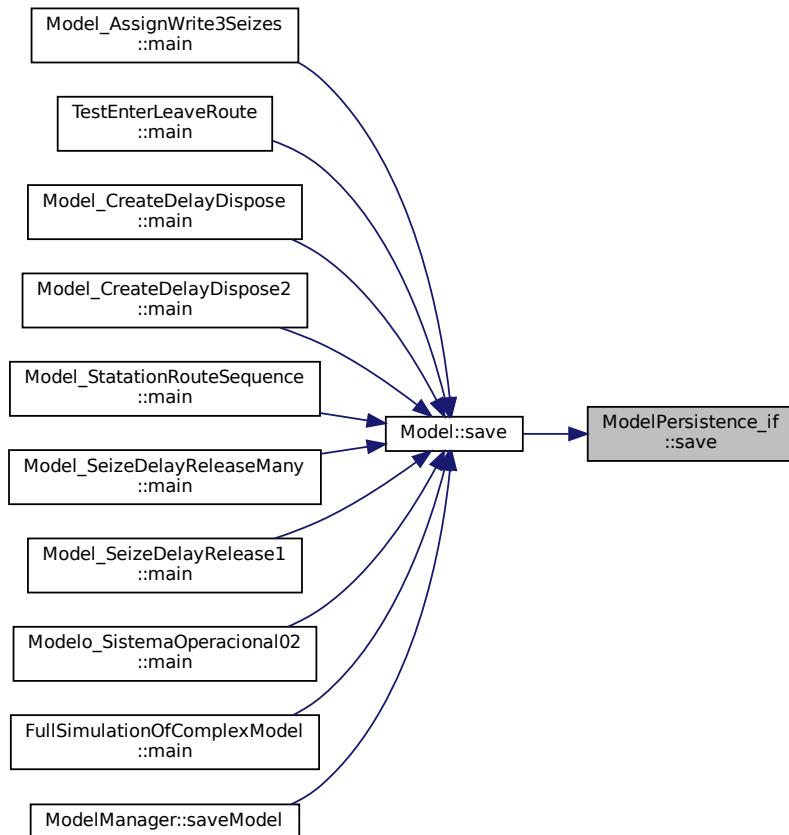


```
8.77.2.3 save() virtual bool ModelPersistence_if::save (
    std::string filename ) [pure virtual]
```

Implemented in [ModelPersistenceDefaultImpl1](#).

Referenced by [Model::save\(\)](#).

Here is the caller graph for this function:



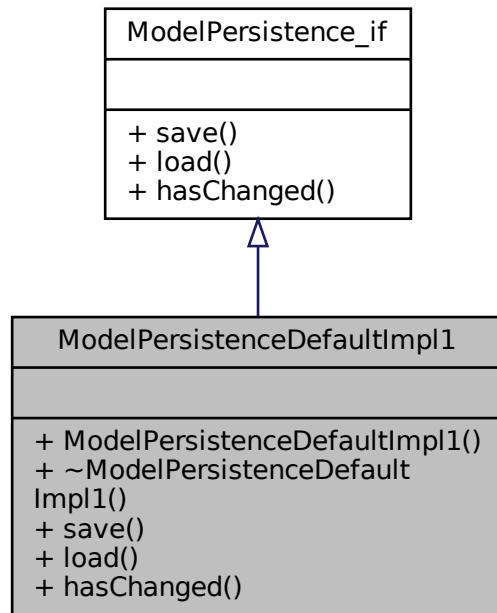
The documentation for this class was generated from the following file:

- [ModelPersistence_if.h](#)

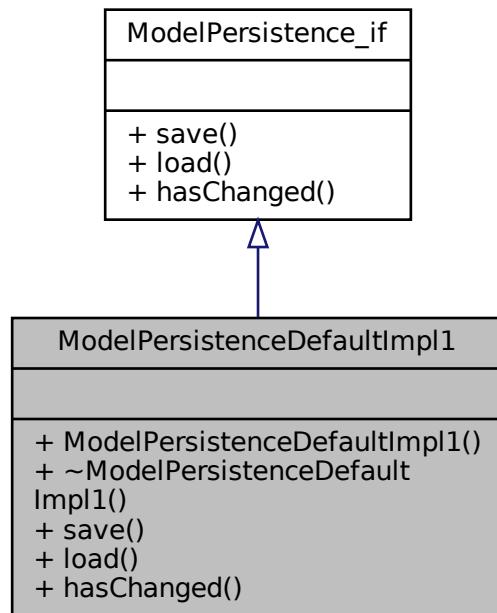
8.78 ModelPersistenceDefaultImpl1 Class Reference

```
#include <ModelPersistenceDefaultImpl1.h>
```

Inheritance diagram for ModelPersistenceDefaultImpl1:



Collaboration diagram for ModelPersistenceDefaultImpl1:



Public Member Functions

- `ModelPersistenceDefaultImpl1 (Model *model)`
- `virtual ~ModelPersistenceDefaultImpl1 ()=default`
- `virtual bool save (std::string filename)`
- `virtual bool load (std::string filename)`
- `virtual bool hasChanged ()`

8.78.1 Detailed Description

Definition at line 22 of file [ModelPersistenceDefaultImpl1.h](#).

8.78.2 Constructor & Destructor Documentation

8.78.2.1 ModelPersistenceDefaultImpl1() `ModelPersistenceDefaultImpl1::ModelPersistenceDefaultImpl1 (`
`Model * model)`

Definition at line 26 of file [ModelPersistenceDefaultImpl1.cpp](#).

```
00026
00027     _model = model;
00028 }
```

8.78.2.2 ~ModelPersistenceDefaultImpl1() `virtual ModelPersistenceDefaultImpl1::~ModelPersistenceDefaultImpl1 () [virtual], [default]`

8.78.3 Member Function Documentation

8.78.3.1 hasChanged() `bool ModelPersistenceDefaultImpl1::hasChanged () [virtual]`

Implements [ModelPersistence_if](#).

Definition at line 351 of file [ModelPersistenceDefaultImpl1.cpp](#).

```
00351
00352     return _hasChanged;
00353 }
```

8.78.3.2 load() bool ModelPersistenceDefaultImpl1::load (std::string filename) [virtual]

Implements [ModelPersistence_if](#).

Definition at line 219 of file [ModelPersistenceDefaultImpl1.cpp](#).

```

00219
00220     //std::list<Plugin*> plugins = this->_model->getParent()->getPlugins();
00221     //plugins->front()->
00222     //return false;
00223     bool res = true;
00224     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Loading file \\" + filename + "\\");
00225     Util::IncIndent();
00226     _componentFields->clear();
00227     {
00228         std::ifstream modelFile;
00229         std::string inputLine;
00230         try {
00231             modelFile.open(filename);
00232             while (getline(modelFile, inputLine) && res) {
00233                 //trim(&inputLine);
00234                 if (inputLine.substr(0, 1) != "#" && !inputLine.empty()) {
00235                     //Util::IncIndent();
00236                     res &= _loadFields(inputLine);
00237                     //Util::DecIndent();
00238                 }
00239             }
00240             modelFile.close();
00241         } catch (...) {
00242             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error loading file \\" + filename + "\\");
00243         }
00244     }
00245     // check if something was loaded
00246     res &= (_model->getComponents()->getNumberOfComponents() > 0) &
00247     (_model->getElements()->getNumberOfElements() > 0);
00248     if (res) {
00249         //
00250         // CONNECT LOADED COMPONENTS (must wait for all components to be loaded so they can be
00251         // connected)
00252         ComponentManager* cm = _model->getComponents();
00253         _model->getTracer()->trace(Util::TraceLevel::simulatorDetailed, "Connecting loaded
00254         components");
00255         Util::IncIndent();
00256         {
00257             for (std::list<std::map<std::string, std::string>>::iterator it =
00258                 _componentFields->begin(); it != _componentFields->end(); it++) {
00259                 std::map<std::string, std::string>* fields = (*it);
00260                 // find the component
00261                 ModelComponent* thisComponent = nullptr;
00262                 Util::identification thisId = std::stoi((*fields->find("id")).second);
00263                 for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp != cm->end();
00264                     itcomp++) {
00265                     if ((*itcomp)->getId() == thisId) {
00266                         thisComponent = (*itcomp);
00267                         break; // end inner for loop
00268                     }
00269                 }
00270                 assert(thisComponent != nullptr);
00271
00272                 // find the next components connected with this one
00273                 unsigned short nextSize = 1;
00274                 if ((*fields->find("nextSize")) != *fields->end()) { // found nextSize
00275                     nextSize = std::stoi((*fields->find("nextSize")).second);
00276                 }
00277                 for (unsigned short i = 0; i < nextSize; i++) {
00278                     Util::identification nextId = std::stoi((*fields->find("nextId" +
00279                         std::to_string(i))).second);
00280                     unsigned short nextInputNumber = 0; // default value if it is not found bellow
00281                     if (fields->find("nextInputNumber" + std::to_string(i)) != fields->end())
00282                         nextInputNumber = std::stoi((*fields->find("nextInputNumber" +
00283                             std::to_string(i))).second);
00284                     ModelComponent* nextComponent = nullptr;
00285                     for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp !=
00286                         cm->end(); itcomp++) // connect the components
00287                     if ((*itcomp)->getId() == nextId) { // connect the components
00288                         nextComponent = (*itcomp);
00289                         thisComponent->getNextComponents()->insert(nextComponent,
00290                             nextInputNumber);
00291                         _model->getTracer()->trace(Util::TraceLevel::toolDetailed,
00292                             thisComponent->getName() + "<" + std::to_string(i) + ">" + " -> " + nextComponent->getName() + "<" +
00293                             std::to_string(nextInputNumber) + ">");
00294                         break;
00295                     }
00296                 }
00297             }
00298         }
00299     }

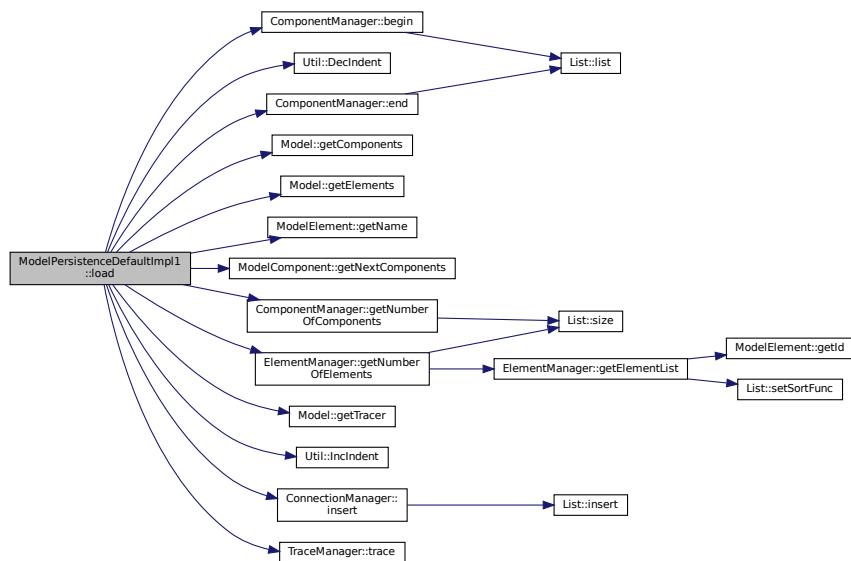
```

```

00286             }
00287             assert(nextComponent != nullptr);
00288         }
00289     }
00290 }
00291 Util::DecIndent();
00292 _model->getTracer()->trace(Util::TraceLevel::simulatorInternal, "File successfully loaded with
00293 " + std::to_string(_model->getComponents()->getNumberOfComponents()) + " components and " +
00294 std::to_string(_model->getElements()->getNumberOfElements()) + " elements");
00295 }
00296 Util::DecIndent();
00297 if (res) {
00298     _hasChanged = false;
00299 }
00300 return res;
00300 }
```

References ComponentManager::begin(), Util::DecIndent(), ComponentManager::end(), Util::errorFatal, Model::getComponents(), Model::getElements(), ModelElement::getName(), ModelComponent::getNextComponents(), ComponentManager::getNumberOfComponents(), ElementManager::getNumberOfElements(), Model::getTracer(), Util::IncIndent(), ConnectionManager::insert(), Util::simulatorDetailed, Util::simulatorInternal, Util::toolDetailed, Util::toolInternal, and TraceManager::trace().

Here is the call graph for this function:



8.78.3.3 save() bool ModelPersistenceDefaultImpl1::save (std::string filename) [virtual]

Implements ModelPersistence_if.

Definition at line 39 of file ModelPersistenceDefaultImpl1.cpp.

```

00039                                         {
00040     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Saving file \"\" + filename + "\\"");
00041     Util::IncIndent();
00042     std::list<std::string> *simulatorInfosToSave, *simulationInfosToSave, *modelInfosToSave,
00043     *modelElementsToSave, *modelComponentsToSave;
00044     {
00045         //bool res = true;
00046         std::map<std::string, std::string>* fields;
00047         fields = _getSimulatorInfoFieldsToSave();
00048         simulatorInfosToSave = _adjustFieldsToSave(fields);
```

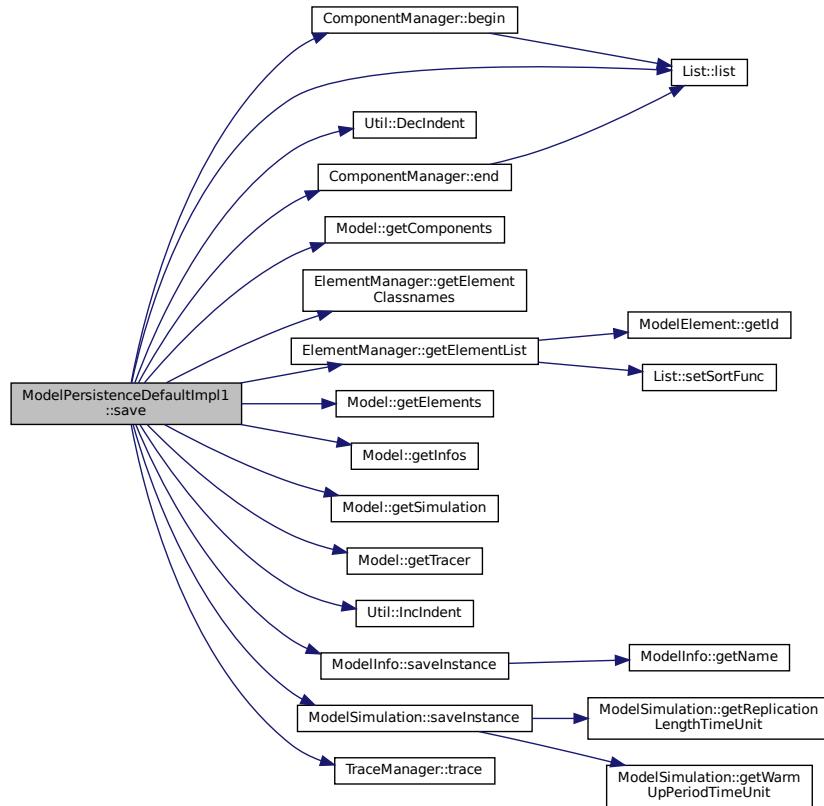
```

00048     // save model own infos
00049     fields = _model->getInfos()->saveInstance();
00050     modelInfosToSave = _adjustFieldsToSave(fields);
00051     // save model own infos
00052     // \todo save modelSimulation fields (breakpoints)
00053     fields = _model->getSimulation()->saveInstance();
00054     simulationInfosToSave = _adjustFieldsToSave(fields);
00055     // save infras
00056     modelElementsToSave = new std::list<std::string>();
00057     std::list<std::string>* elementTypenames = _model->getElements()->elementClassnames();
00058     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00059     for (std::list<std::string>::iterator itTypenames = elementTypenames->begin(); itTypenames != elementTypenames->end(); itTypenames++) {
00060         if ((*itTypenames) != Util::TypeOf<StatisticsCollector>() && (*itTypenames) != UtilTypeOfCounter) { // STATISTICS COLLECTR and COUNTERS do NOT need to be saved
00061             List<ModelElement*>* infras = _model->getElements()->elementList((*itTypenames));
00062             _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing elements of type "
00063             \"" + (*itTypenames) + "\":");
00064             Util::IncIndent();
00065             for (std::list<ModelElement*>::iterator it = infras->list()->begin(); it != infras->list()->end(); it++) {
00066                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing " +
00067                 (*itTypenames) + " \\" + (*it)->getName() + "\\"");
00068                 fields = (*it)->SaveInstance((*it));
00069                 Util::IncIndent();
00070                 modelElementsToSave->merge(*_adjustFieldsToSave(fields));
00071                 Util::DecIndent();
00072             }
00073             Util::DecIndent();
00074         }
00075     }
00076     // save components
00077     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing components\":");
00078     //List<ModelComponent*>* components = this->_model->getComponents();
00079     modelComponentsToSave = new std::list<std::string>();
00080     Util::IncIndent();
00081     {
00082         for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00083             fields = (*it)->SaveInstance((*it));
00084             Util::IncIndent();
00085             modelComponentsToSave->merge(*_adjustFieldsToSave(fields));
00086             Util::DecIndent();
00087         }
00088     }
00089     Util::DecIndent();
00090     // SAVE FILE
00091     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Saving file");
00092     Util::IncIndent();
00093     {
00094         // open file
00095         std::ofstream savefile;
00096         savefile.open(filename, std::ofstream::out);
00097         savefile « "# Genesys simulation model " « std::endl;
00098         time_t now = time(0);
00099         char* dt = ctime(&now);
00100         savefile « "# Last saved on " « dt;
00101         savefile « "# simulator infos" « std::endl;
00102         _saveContent(simulatorInfosToSave, &savefile);
00103         savefile « "# model infos" « std::endl;
00104         _saveContent(modelInfosToSave, &savefile);
00105         savefile « "# simulation infos / experimental design" « std::endl;
00106         _saveContent(simulationInfosToSave, &savefile);
00107         savefile « "#" « std::endl;
00108         savefile « "# model elements" « std::endl;
00109         _saveContent(modelElementsToSave, &savefile);
00110         savefile « "#" « std::endl;
00111         savefile « "# model components" « std::endl;
00112         _saveContent(modelComponentsToSave, &savefile);
00113         savefile.close();
00114     }
00115     Util::DecIndent();
00116 }
00117 Util::DecIndent();
00118 this->_hasChanged = false;
00119 return true; // \todo: check if save really saved successfully
00120 }

```

References ComponentManager::begin(), Util::DecIndent(), ComponentManager::end(), Model::getComponents(), ElementManager::getElementClassnames(), ElementManager::getElementList(), Model::getElements(), Model::getInfos(), Model::getSimulation(), Model::getTracer(), Util::IncIndent(), List< T >::list(), ModelInfo::saveInstance(), ModelSimulation::saveInstance(), Util::toolDetailed, Util::toolInternal, and TraceManager::trace().

Here is the call graph for this function:



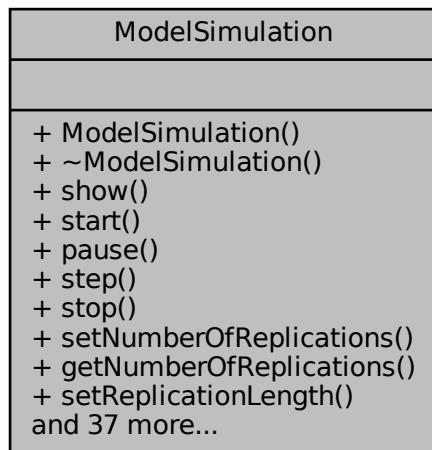
The documentation for this class was generated from the following files:

- [ModelPersistenceDefaultImpl1.h](#)
- [ModelPersistenceDefaultImpl1.cpp](#)

8.79 ModelSimulation Class Reference

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

Collaboration diagram for ModelSimulation:



Public Member Functions

- `ModelSimulation (Model *model)`
- virtual `~ModelSimulation ()=default`
- `std::string show ()`
- `void start ()`

Starts a sequential execution of a simulation, ie, a set of replications of this model.
- `void pause ()`
- `void step ()`

Executes the processing of a single event, the next one in the future events list.
- `void stop ()`
- `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`
- `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`
- void `setReporter (SimulationReporter_if *_simulationReporter)`
- `SimulationReporter_if * getReporter () const`
- double `getSimulatedTime () const`
- bool `isRunning () const`
- bool `isPaused () const`
- unsigned int `getCurrentReplicationNumber () const`
- `ModelComponent * getCurrentComponent () const`
- `Entity * getCurrentEntity () const`
- unsigned int `getCurrentInputNumber () const`
- void `setShowReportsAfterReplication (bool showReportsAfterReplication)`
- bool `isShowReportsAfterReplication () const`
- void `setShowReportsAfterSimulation (bool showReportsAfterSimulation)`
- bool `isShowReportsAfterSimulation () const`
- `List< double > * getBreakpointsOnTime () const`
- `List< Entity * > * getBreakpointsOnEntity () const`
- `List< ModelComponent * > * getBreakpointsOnComponent () const`
- void `loadInstance (std::map< std::string, std::string > *fields)`
- `std::map< std::string, std::string > * saveInstance ()`

8.79.1 Detailed Description

The `ModelSimulation` controls the simulation of a model, allowing to start, pause, resume or stop a simulation, composed by a set of replications.

Definition at line 30 of file `ModelSimulation.h`.

8.79.2 Constructor & Destructor Documentation

8.79.2.1 `ModelSimulation()`

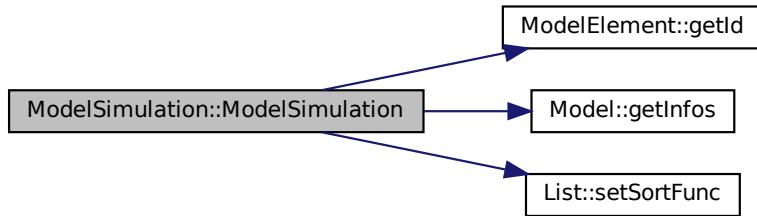
```
ModelSimulation::ModelSimulation (
    Model * model )
```

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

```
00029                                         {
00030     _model = model;
00031     _info = model->getInfos();
00032     _statsCountersSimulation->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00033         return a->getId() < b->getId();
00034     });
00035     _simulationReporter = new Traits<SimulationReporter_if>::Implementation(this, model,
00036     this->_statsCountersSimulation);
00036 }
```

References `ModelElement::getId()`, `Model::getInfos()`, and `List< T >::setSortFunc()`.

Here is the call graph for this function:



8.79.2.2 ~ModelSimulation() `virtual ModelSimulation::~ModelSimulation () [virtual], [default]`

8.79.3 Member Function Documentation

8.79.3.1 getBreakpointsOnComponent() `List< ModelComponent * > * ModelSimulation::getBreakpointsOnComponent () const`

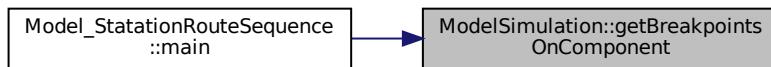
Definition at line 543 of file [ModelSimulation.cpp](#).

```

00543
00544     return _breakpointsOnComponent;
00545 }
```

Referenced by [Model_StatationRouteSequence::main\(\)](#).

Here is the caller graph for this function:



8.79.3.2 getBreakpointsOnEntity() `List< Entity * > * ModelSimulation::getBreakpointsOnEntity () const`

Definition at line 539 of file [ModelSimulation.cpp](#).

```

00539
00540     return _breakpointsOnEntity;
00541 }
```

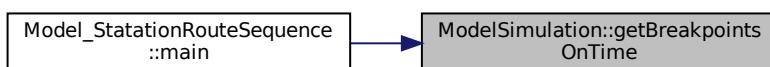
8.79.3.3 getBreakpointsOnTime() `List< double > * ModelSimulation::getBreakpointsOnTime () const`

Definition at line 535 of file [ModelSimulation.cpp](#).

```
00535
00536     return _breakpointsOnTime;
00537 }
```

Referenced by [Model_StationRouteSequence::main\(\)](#).

Here is the caller graph for this function:

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

Definition at line 499 of file [ModelSimulation.cpp](#).

```
00499
00500     return _currentComponent;
00501 }
```

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

Definition at line 503 of file [ModelSimulation.cpp](#).

```
00503
00504     return _currentEntity;
00505 }
```

8.79.3.6 getCurrentInputNumber() `unsigned int ModelSimulation::getCurrentInputNumber () const`

Definition at line 515 of file [ModelSimulation.cpp](#).

```
00515
00516     return _currentInputNumber;
00517 }
```

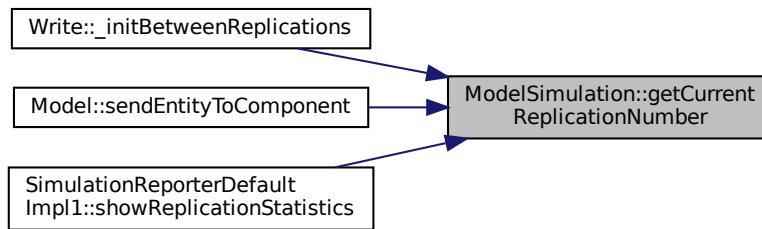
8.79.3.7 getCurrentReplicationNumber() `unsigned int ModelSimulation::getCurrentReplicationNumber () const`

Definition at line 495 of file [ModelSimulation.cpp](#).

```
00495
00496     return _currentReplicationNumber;
00497 }
```

Referenced by [Write::_initBetweenReplications\(\)](#), [Model::sendEntityToComponent\(\)](#), and [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#).

Here is the caller graph for this function:



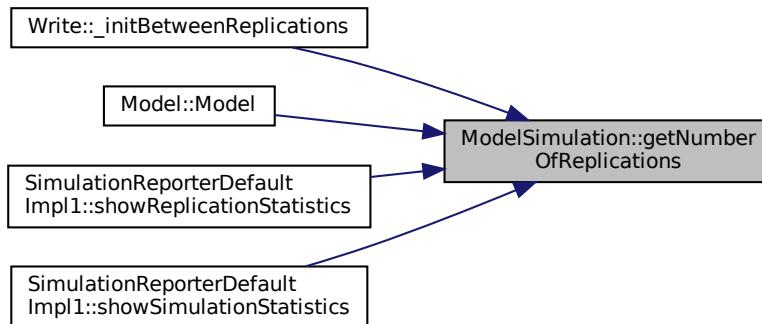
8.79.3.8 getNumberOfReplications() `unsigned int ModelSimulation::getNumberOfReplications () const`

Definition at line 556 of file [ModelSimulation.cpp](#).

```
00556
00557     return _numberOfReplications;
00558 }
```

Referenced by [Write::_initBetweenReplications\(\)](#), [Model::Model\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



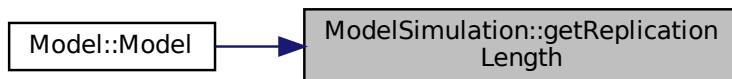
8.79.3.9 `getReplicationLength()` `double ModelSimulation::getReplicationLength () const`

Definition at line 565 of file [ModelSimulation.cpp](#).

```
00565
00566     return _replicationLength;
00567 }
```

Referenced by [Model::Model\(\)](#).

Here is the caller graph for this function:



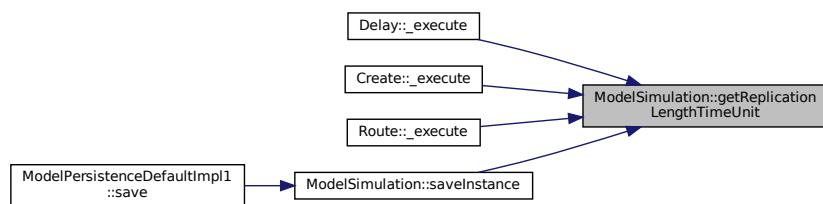
8.79.3.10 `getReplicationLengthTimeUnit()` `Util::TimeUnit ModelSimulation::getReplicationLengthTimeUnit () const`

Definition at line 574 of file [ModelSimulation.cpp](#).

```
00574
00575     return _replicationLengthTimeUnit;
00576 }
```

Referenced by [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), and [saveInstance\(\)](#).

Here is the caller graph for this function:



8.79.3.11 getReporter() `SimulationReporter_if * ModelSimulation::getReporter () const`Definition at line 511 of file [ModelSimulation.cpp](#).

```
00511
00512     return _simulationReporter;
00513 }
```

Referenced by [GenesysConsole::cmdShowReport\(\)](#).

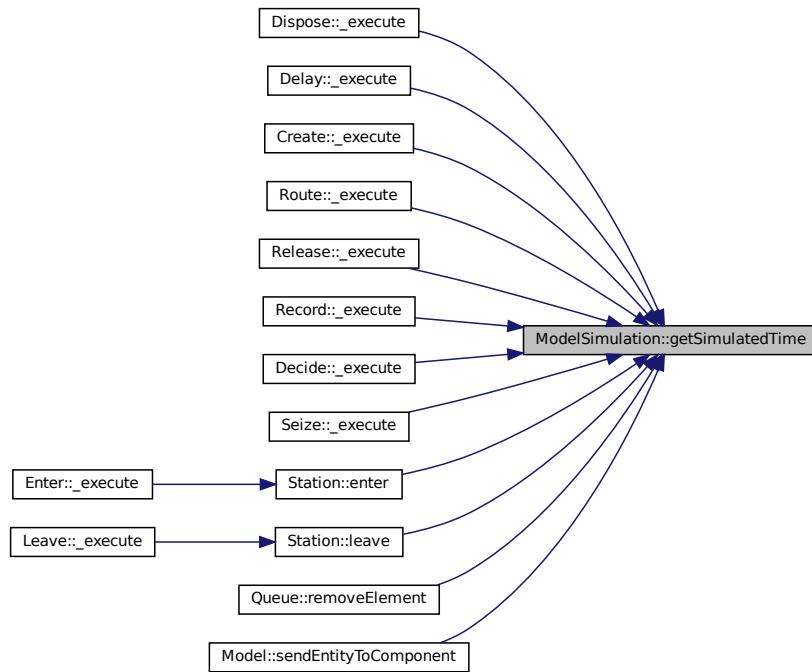
Here is the caller graph for this function:

**8.79.3.12 getSimulatedTime()** `double ModelSimulation::getSimulatedTime () const`Definition at line 487 of file [ModelSimulation.cpp](#).

```
00487
00488     return _simulatedTime;
00489 }
```

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Release::_execute\(\)](#), [Record::_execute\(\)](#), [Decide::_execute\(\)](#), [Seize::_execute\(\)](#), [Station::enter\(\)](#), [Station::leave\(\)](#), [Queue::removeElement\(\)](#), and [Model::sendEntityToComponent\(\)](#).

Here is the caller graph for this function:



8.79.3.13 getTerminatingCondition() `std::string ModelSimulation::getTerminatingCondition () const`

Definition at line 601 of file [ModelSimulation.cpp](#).

```
00601
00602     return _terminatingCondition;
00603 }
```

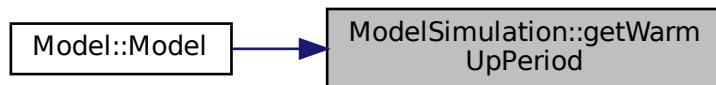
8.79.3.14 getWarmUpPeriod() `double ModelSimulation::getWarmUpPeriod () const`

Definition at line 583 of file [ModelSimulation.cpp](#).

```
00583
00584     return _warmUpPeriod;
00585 }
```

Referenced by [Model::Model\(\)](#).

Here is the caller graph for this function:



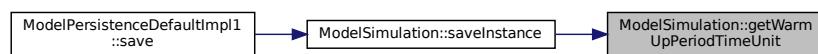
8.79.3.15 getWarmUpPeriodTimeUnit() `Util::TimeUnit ModelSimulation::getWarmUpPeriodTimeUnit () const`

Definition at line 592 of file [ModelSimulation.cpp](#).

```
00592
00593     return _warmUpPeriodTimeUnit;
00594 }
```

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

Here is the caller graph for this function:



8.79.3.16 isInitializeStatistics() bool ModelSimulation::isInitializeStatistics () const

Definition at line 459 of file [ModelSimulation.cpp](#).

```
00459
00460     return _initializeStatisticsBetweenReplications;
00461 }
```

8.79.3.17 isInitializeSystem() bool ModelSimulation::isInitializeSystem () const

Definition at line 467 of file [ModelSimulation.cpp](#).

```
00467
00468     return _initializeSystem;
00469 }
```

8.79.3.18 isPaused() bool ModelSimulation::isPaused () const

Definition at line 547 of file [ModelSimulation.cpp](#).

```
00547
00548     return _isPaused;
00549 }
```

Referenced by [Model_StatationRouteSequence::main\(\)](#).

Here is the caller graph for this function:

**8.79.3.19 isPauseOnEvent()** bool ModelSimulation::isPauseOnEvent () const

Definition at line 451 of file [ModelSimulation.cpp](#).

```
00451
00452     return _pauseOnEvent;
00453 }
```

8.79.3.20 isPauseOnReplication() bool ModelSimulation::isPauseOnReplication () const

Definition at line 483 of file [ModelSimulation.cpp](#).

```
00483
00484     return _pauseOnReplication;
00485 }
```

8.79.3.21 isRunning() bool ModelSimulation::isRunning () const

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

Definition at line 491 of file [ModelSimulation.cpp](#).

```
00491     {
00492         return _isRunning;
00493     }
```

8.79.3.22 isShowReportsAfterReplication() bool ModelSimulation::isShowReportsAfterReplication () const

Definition at line 523 of file [ModelSimulation.cpp](#).

```
00523     {
00524         return _showReportsAfterReplication;
00525     }
```

8.79.3.23 isShowReportsAfterSimulation() bool ModelSimulation::isShowReportsAfterSimulation () const

Definition at line 531 of file [ModelSimulation.cpp](#).

```
00531     {
00532         return _showReportsAfterSimulation;
00533     }
```

8.79.3.24 isStepByStep() bool ModelSimulation::isStepByStep () const

Definition at line 475 of file [ModelSimulation.cpp](#).

```
00475     {
00476         return _stepByStep;
00477     }
```

8.79.3.25 loadInstance() void ModelSimulation::loadInstance (std::map< std::string, std::string * > * fields)

Definition at line 605 of file [ModelSimulation.cpp](#).

```
00605     {
00606         this->_numberOfReplications = std::stoi(loadField(fields, "numberOfReplications", "1"));
00607         this->_replicationLength = std::stod(loadField(fields, "replicationLength", "3600.0"));
00608         this->_replicationLengthTimeUnit = static_cast<Util::.TimeUnit>
00609             (std::stoi((*fields->find("replicationLengthTimeUnit")).second));
00610         this->_terminatingCondition = loadField(fields, "terminatingCondition", "");
00611         this->_warmUpPeriod = std::stod(loadField(fields, "warmUpTime", "0.0"));
00612         this->_warmUpPeriodTimeUnit = static_cast<Util::.TimeUnit>
00613             (std::stoi((*fields->find("warmUpTimeTimeUnit"))).second));
00614         _hasChanged = false;
00615     }
```

References [loadField\(\)](#).

Here is the call graph for this function:



8.79.3.26 pause() void ModelSimulation::pause ()Definition at line 425 of file [ModelSimulation.cpp](#).

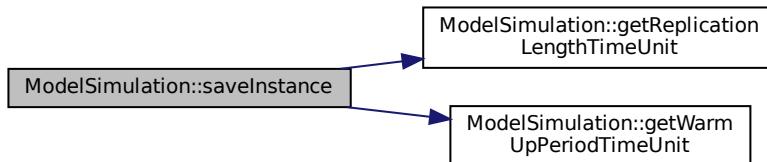
```
00425     {
00426 }
```

8.79.3.27 saveInstance() std::map< std::string, std::string > * ModelSimulation::saveInstance ()Definition at line 617 of file [ModelSimulation.cpp](#).

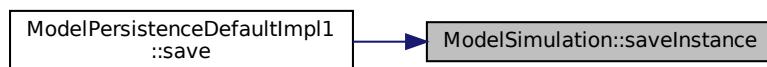
```
00617     {
00618     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00619     fields->emplace("typename", "ModelSimulation");
00620     if (_numberOfReplications != 1) fields->emplace("numberOfReplications",
00621         std::to_string(_numberOfReplications));
00622     if (this->_replicationLength != 3600) fields->emplace("replicationLength",
00623         std::to_string(_replicationLength));
00624     fields->emplace("replicationLengthTimeUnit", std::to_string(static_cast<int>
00625         (getReplicationLengthTimeUnit())));
00626     if (this->_terminatingCondition != "") fields->emplace("terminatingCondition", "\""
00627         + _terminatingCondition + "\"");
00628     if (this->_warmUpPeriod) fields->emplace("warmUpTime", std::to_string(_warmUpPeriod));
00629     fields->emplace("warmUpTimeTimeUnit", std::to_string(static_cast<int>
00630         (getWarmUpPeriodTimeUnit())));
00631     _hasChanged = false;
00632     return fields;
00633 }
```

References [getReplicationLengthTimeUnit\(\)](#), and [getWarmUpPeriodTimeUnit\(\)](#).Referenced by [ModelPersistenceDefaultImpl1::save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.79.3.28 setInitializeStatistics() void ModelSimulation::setInitializeStatistics (bool _initializeStatistics)

Definition at line 455 of file [ModelSimulation.cpp](#).

```
00455     {
00456         this->_initializeStatisticsBetweenReplications = _initializeStatistics;
00457     }
```

8.79.3.29 setInitializeSystem() void ModelSimulation::setInitializeSystem (bool _initializeSystem)

Definition at line 463 of file [ModelSimulation.cpp](#).

```
00463     {
00464         this->_initializeSystem = _initializeSystem;
00465     }
```

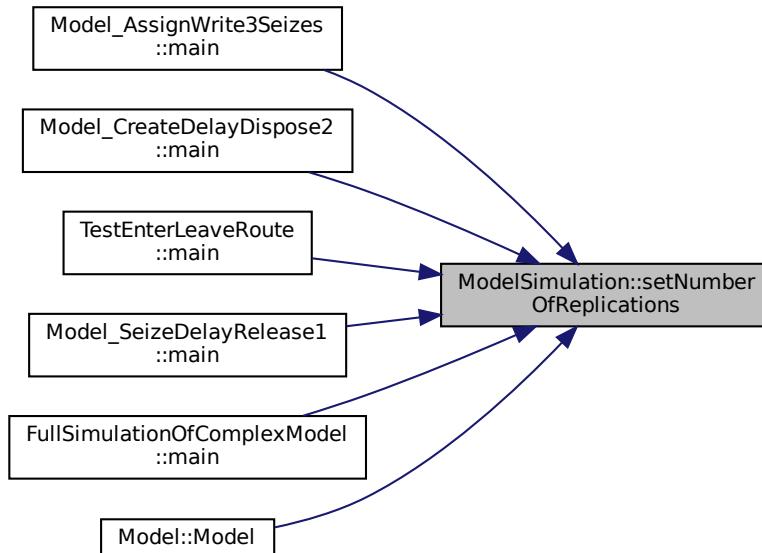
8.79.3.30 setNumberOfReplications() void ModelSimulation::setNumberOfReplications (unsigned int _numberOfReplications)

Definition at line 551 of file [ModelSimulation.cpp](#).

```
00551     {
00552         this->_numberOfReplications = _numberOfReplications;
00553         _hasChanged = true;
00554     }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), and [Model::Model\(\)](#).

Here is the caller graph for this function:



8.79.3.31 setPauseOnEvent() void ModelSimulation::setPauseOnEvent (bool _pauseOnEvent)

Definition at line 447 of file ModelSimulation.cpp.

```
00447
00448     this->_pauseOnEvent = _pauseOnEvent;
00449 }
```

8.79.3.32 setPauseOnReplication() void ModelSimulation::setPauseOnReplication (bool _pauseBetweenReplications)

Definition at line 479 of file ModelSimulation.cpp.

```
00479
00480     this->_pauseOnReplication = _pauseOnReplication;
00481 }
```

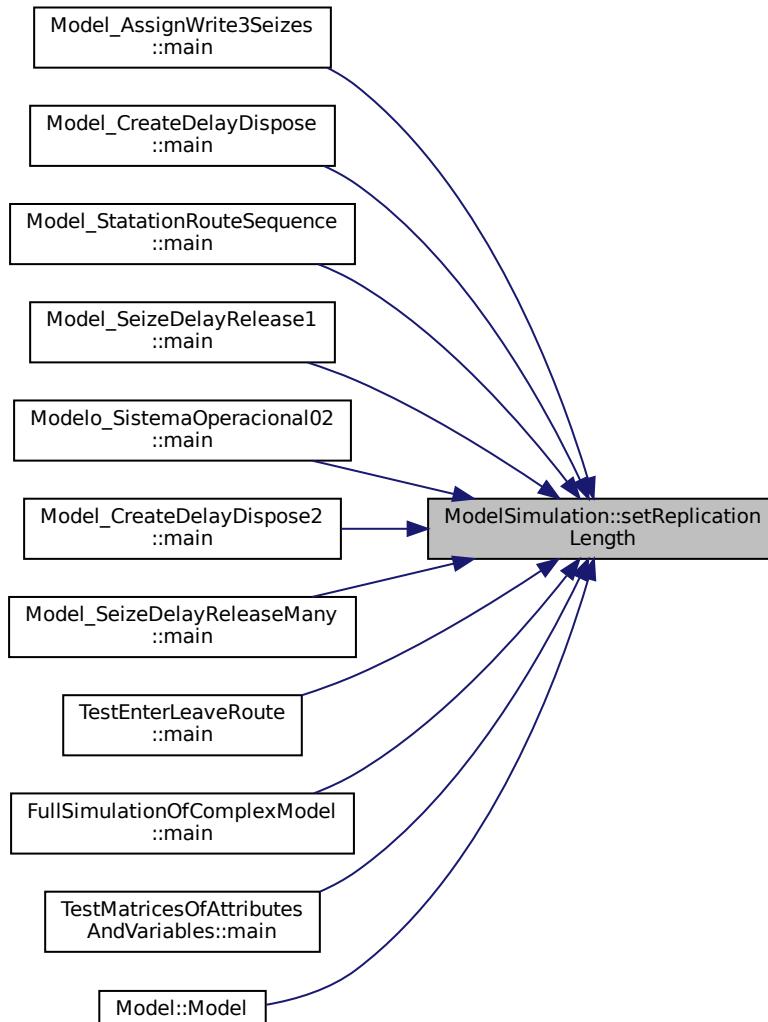
8.79.3.33 setReplicationLength() void ModelSimulation::setReplicationLength (double _replicationLength)

Definition at line 560 of file ModelSimulation.cpp.

```
00560
00561     this->_replicationLength = _replicationLength;
00562     _hasChanged = true;
00563 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), and [Model::Model\(\)](#).

Here is the caller graph for this function:



8.79.3.34 setReplicationLengthTimeUnit() void ModelSimulation::setReplicationLengthTimeUnit (Util::TimeUnit _replicationLengthTimeUnit)

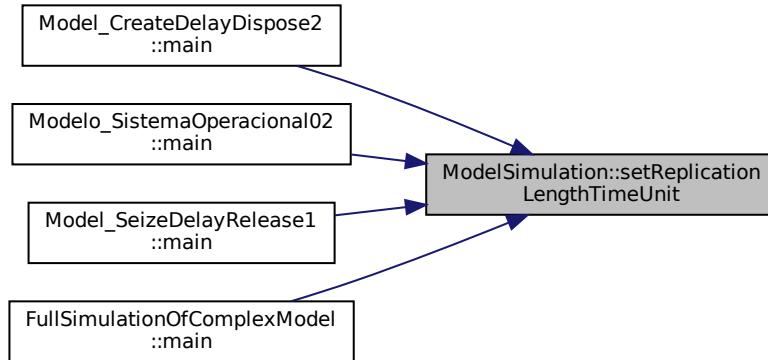
Definition at line 569 of file [ModelSimulation.cpp](#).

```

00569
00570     this->_replicationLengthTimeUnit = _replicationLengthTimeUnit;
00571     _hasChanged = true;
00572 }
```

Referenced by [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.79.3.35 setReporter() void ModelSimulation::setReporter (
 SimulationReporter_if * *_simulationReporter*)

Definition at line 507 of file [ModelSimulation.cpp](#).

```

00507
00508     this->_simulationReporter = _simulationReporter;
00509 }
```

8.79.3.36 setShowReportsAfterReplication() void ModelSimulation::setShowReportsAfterReplication (
 bool *showReportsAfterReplication*)

Definition at line 519 of file [ModelSimulation.cpp](#).

```

00519
00520     this->_showReportsAfterReplication = showReportsAfterReplication;
00521 }
```

8.79.3.37 setShowReportsAfterSimulation() void ModelSimulation::setShowReportsAfterSimulation (
 bool *showReportsAfterSimulation*)

Definition at line 527 of file [ModelSimulation.cpp](#).

```

00527
00528     this->_showReportsAfterSimulation = showReportsAfterSimulation;
00529 }
```

8.79.3.38 setStepByStep() void ModelSimulation::setStepByStep (bool _stepByStep)

Definition at line 471 of file [ModelSimulation.cpp](#).

```
00471
00472     this->_stepByStep = _stepByStep;
00473 }
```

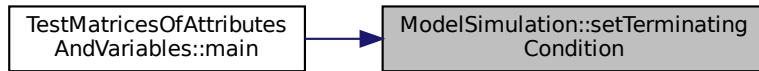
8.79.3.39 setTerminatingCondition() void ModelSimulation::setTerminatingCondition (std::string _terminatingCondition)

Definition at line 596 of file [ModelSimulation.cpp](#).

```
00596
00597     this->_terminatingCondition = _terminatingCondition;
00598     _hasChanged = true;
00599 }
```

Referenced by [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the caller graph for this function:



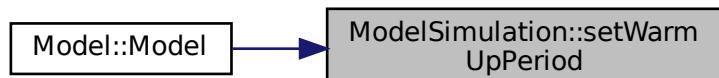
8.79.3.40 setWarmUpPeriod() void ModelSimulation::setWarmUpPeriod (double _warmUpPeriod)

Definition at line 578 of file [ModelSimulation.cpp](#).

```
00578
00579     this->_warmUpPeriod = _warmUpPeriod;
00580     _hasChanged = true;
00581 }
```

Referenced by [Model::Model\(\)](#).

Here is the caller graph for this function:



8.79.3.41 setWarmUpPeriodTimeUnit() void ModelSimulation::setWarmUpPeriodTimeUnit (Util::TimeUnit _warmUpPeriodTimeUnit)

Definition at line 587 of file [ModelSimulation.cpp](#).

```
00587
00588     this->_warmUpPeriodTimeUnit = _warmUpPeriodTimeUnit;
00589     _hasChanged = true;
00590 }
```

8.79.3.42 show() std::string ModelSimulation::show ()

Definition at line 38 of file [ModelSimulation.cpp](#).

```
00038
00039     {
00040         return "numberOfReplications=" + std::to_string(_numberOfReplications) +
00041             ",replicationLength=" + std::to_string(_replicationLength) + " " +
00042             Util::StrTimeUnit(this->_replicationLengthTimeUnit) +
00043             ",terminatingCondition=\"" + this->_terminatingCondition + "\" " +
00044             ",warmupTime=" + std::to_string(this->_warmUpPeriod) + " " +
00045             Util::StrTimeUnit(this->_warmUpPeriodTimeUnit);
00046     }
00047 }
```

References [Util::StrTimeUnit\(\)](#).

Here is the call graph for this function:



8.79.3.43 start() void ModelSimulation::start ()

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

Definition at line 73 of file [ModelSimulation.cpp](#).

```
00073
00074     if (!_isPaused) { // begin of a new simulation
00075         Util::SetIndent(0); //force indentation
00076         if (!_model->check()) {
00077             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Model check failed. Cannot start
simulation.");
00078             return;
00079         }
00080         _initSimulation();
00081         _model->getOnEvents()->NotifySimulationStartHandlers(new SimulationEvent(0, nullptr));
00082         _currentReplicationNumber = 1;
00083         Util::IncIndent();
00084         _initReplication();
00085     } else { // continue after a pause
00086         _model->getTracer()->trace("Replication resumed", Util::TraceLevel::modelSimulationEvent);
00087         _model->getOnEvents()->NotifySimulationPausedStartHandlers(new SimulationEvent(0, nullptr));
00088     }
00089     _isRunning = true;
```

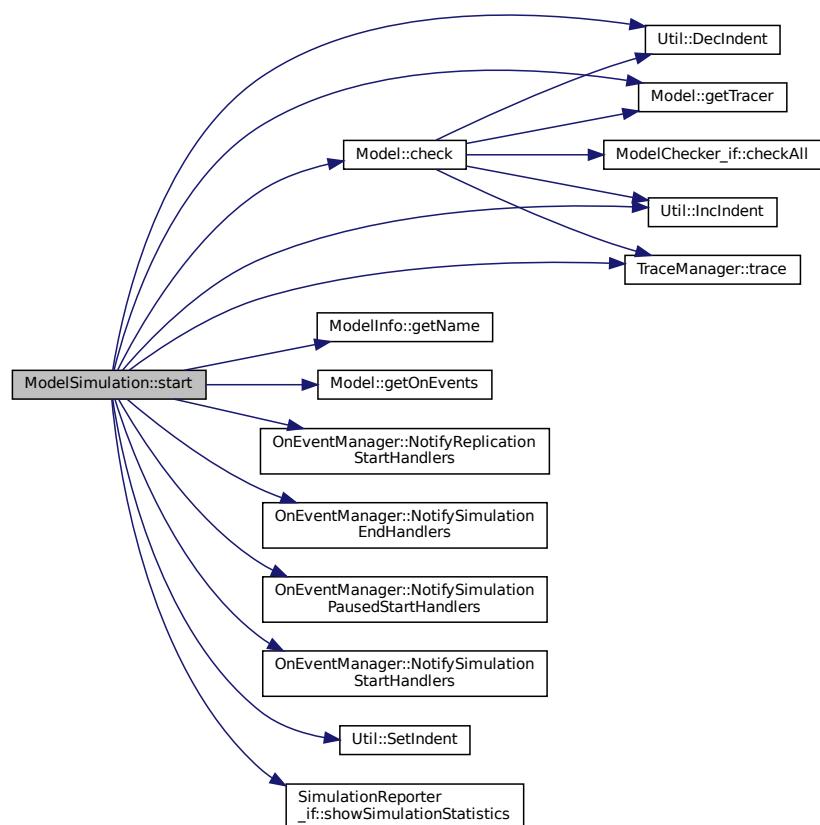
```

00090     _isPaused = false;
00091     //std::chrono::time_point<std::chrono::_V2::system_clock, std::chrono::duration<long int,
00092     std::ratio < 1, 1000000000 > replicationStartTime = std::chrono::high_resolution_clock::now();
00093     do {
00094         Util::SetIndent(1);
00095         _model->getOnEvents()->NotifyReplicationStartHandlers(new
00096             SimulationEvent(_currentReplicationNumber, nullptr));
00097         // main simulation loop
00098         Util::IncIndent();
00099         while (!_isReplicationEndCondition() && !_pauseRequested) {
00100             _stepSimulation();
00101             if (!_pauseRequested) {
00102                 Util::SetIndent(1); // force
00103                 _replicationEnded();
00104                 _currentReplicationNumber++;
00105                 if (_currentReplicationNumber <= _numberOfReplications) {
00106                     _initReplication();
00107                 }
00108             } while (_currentReplicationNumber <= _numberOfReplications && !_pauseRequested);
00109             if (!_pauseRequested) {
00110                 if (this->_showReportsAfterSimulation)
00111                     _simulationReporter->showSimulationStatistics(); //_cStatsSimulation);
00112             Util::DecIndent();
00113             _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, "Simulation of model \""
00114             _info->getName() + "\" has finished.\n");
00115             _model->getOnEvents()->NotifySimulationEndHandlers(new SimulationEvent(0, nullptr));
00116         } else {
00117             _pauseRequested = false;
00118             _isPaused = true;
00119         }
00120     } _isRunning = false;
00121 }
```

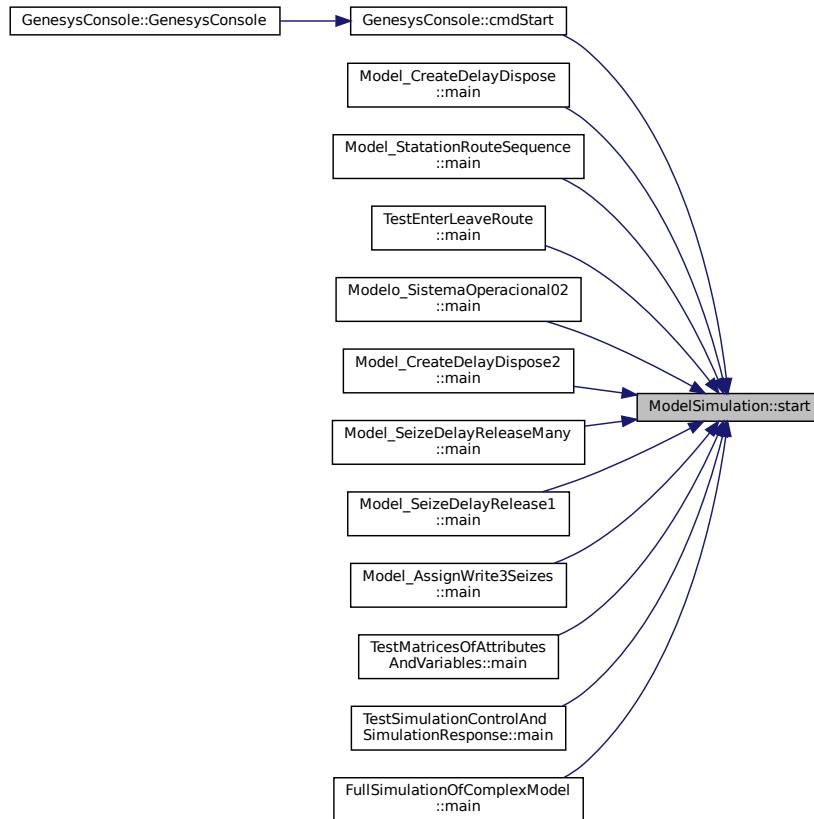
References [Model::check\(\)](#), [Util::DecIndent\(\)](#), [Util::errorFatal](#), [ModelInfo::getName\(\)](#), [Model::getOnEvents\(\)](#), [Model::getTracer\(\)](#), [Util::IncIndent\(\)](#), [Util::modelSimulationEvent](#), [OnEventManager::NotifyReplicationStartHandlers\(\)](#), [OnEventManager::NotifySimulationEndHandlers\(\)](#), [OnEventManager::NotifySimulationPausedStartHandlers\(\)](#), [OnEventManager::NotifySimulationStartHandlers\(\)](#), [Util::SetIndent\(\)](#), [SimulationReporter_if::showSimulationStatistics\(\)](#), and [TraceManager::trace\(\)](#).

Referenced by [GenesysConsole::cmdStart\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.79.3.44 step() void ModelSimulation::step ()

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

Definition at line 428 of file [ModelSimulation.cpp](#).

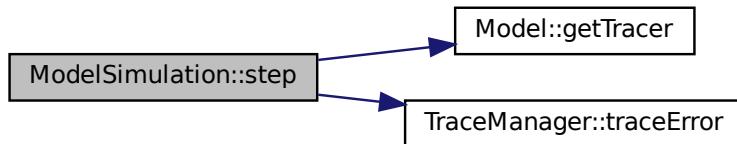
```

00428     {
00429     if (_simulationIsInitiated && _replicationIsInitiated) {
00430         if (!isReplicationEndCondition()) {
00431             try {
00432                 this->stepSimulation();
00433             } catch (std::exception *e) {
00434                 _model->getTracer()->traceError((*e), "Error on simulation step");
00435             }
00436         }
00437     }
00438 }
  
```

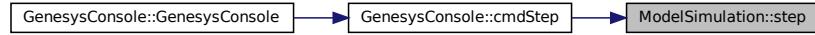
References [Model::getTracer\(\)](#), and [TraceManager::traceError\(\)](#).

Referenced by [GenesysConsole::cmdStep\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



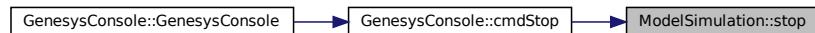
8.79.3.45 `stop()` void ModelSimulation::stop ()

Definition at line 440 of file [ModelSimulation.cpp](#).

```
00440     {
00441     this->_isPaused = false;
00442     this->_isRunning = false;
00443     this->_replicationIsInitiated = false;
00444     this->_simulationIsInitiated = false;
00445 }
```

Referenced by [GenesysConsole::cmdStop\(\)](#).

Here is the caller graph for this function:



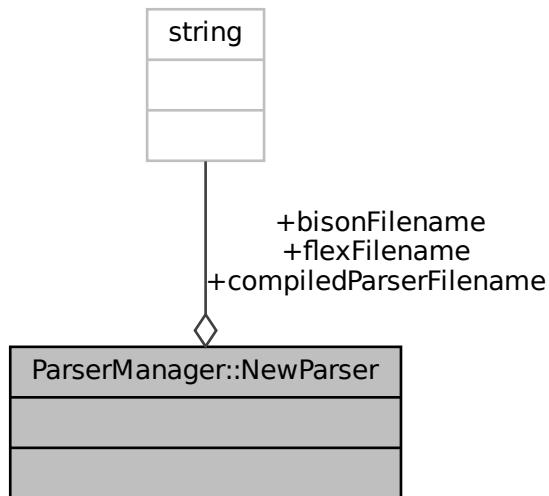
The documentation for this class was generated from the following files:

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

8.80 ParserManager::NewParser Struct Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager::NewParser:



Public Attributes

- `std::string bisonFilename`
- `std::string flexFilename`
- `std::string compiledParserFilename`

8.80.1 Detailed Description

Definition at line 24 of file [ParserManager.h](#).

8.80.2 Member Data Documentation

8.80.2.1 `bisonFilename` `std::string ParserManager::NewParser::bisonFilename`

Definition at line 25 of file [ParserManager.h](#).

8.80.2.2 compiledParserFilename std::string ParserManager::NewParser::compiledParserFilename

Definition at line 27 of file [ParserManager.h](#).

8.80.2.3 flexFilename std::string ParserManager::NewParser::flexFilename

Definition at line 26 of file [ParserManager.h](#).

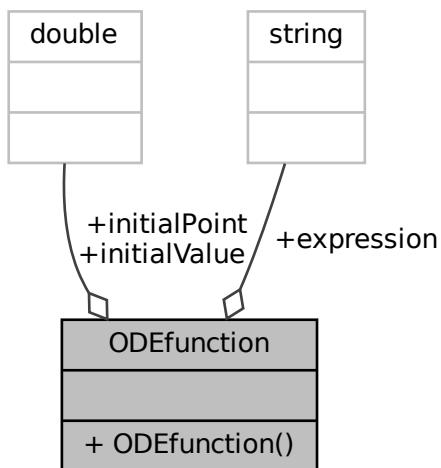
The documentation for this struct was generated from the following file:

- [ParserManager.h](#)

8.81 ODEfunction Class Reference

```
#include <OLD_ODEelement.h>
```

Collaboration diagram for ODEfunction:

**Public Member Functions**

- `ODEfunction (std::string expression, double initialPoint, double initialValue)`

Public Attributes

- `std::string expression`
- `double initialPoint`
- `double initialValue`

8.81.1 Detailed Description

Definition at line 21 of file [OLD_ODEElement.h](#).

8.81.2 Constructor & Destructor Documentation

8.81.2.1 ODEfunction()

```
ODEfunction::ODEfunction (
    std::string expression,
    double initialPoint,
    double initialValue ) [inline]
```

Definition at line 24 of file [OLD_ODEElement.h](#).

```
00024
00025     this->expression = expression;
00026     this->initialPoint = initialPoint;
00027     this->initialValue = initialValue;
00028 }
```

References [expression](#), [initialPoint](#), and [initialValue](#).

8.81.3 Member Data Documentation

8.81.3.1 expression std::string ODEfunction::expression

Definition at line 29 of file [OLD_ODEElement.h](#).

Referenced by [OLD_ODEElement::_check\(\)](#), and [ODEfunction\(\)](#).

8.81.3.2 initialPoint double ODEfunction::initialPoint

Definition at line 30 of file [OLD_ODEElement.h](#).

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

8.81.3.3 initialValue double ODEfunction::initialValue

Definition at line 31 of file [OLD_ODEElement.h](#).

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

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

- [OLD_ODEElement.h](#)

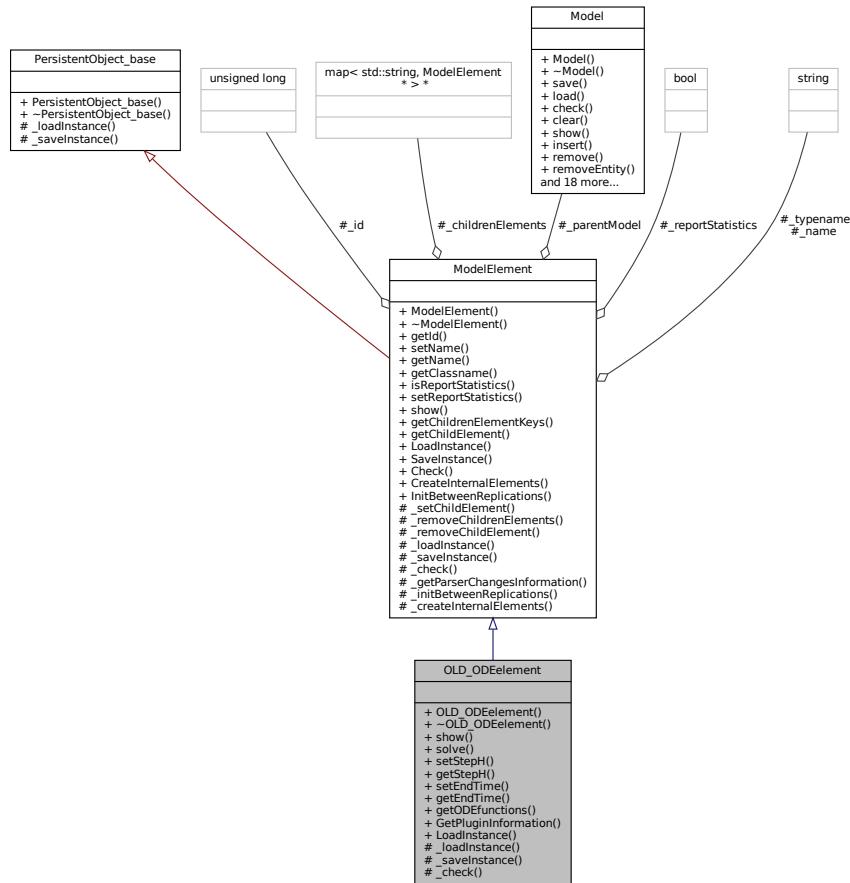
8.82 OLD_ODElement Class Reference

```
#include <OLD_ODElement.h>
```

Inheritance diagram for OLD_ODElement:



Collaboration diagram for OLD_ODEelement:



Public Member Functions

- `OLD_ODEElement (Model *model, std::string name="")`
 - `virtual ~OLD_ODEElement ()=default`
 - `virtual std::string show ()`
 - `double solve ()`
 - `void setStepH (double _h)`
 - `double getStepH () const`
 - `void setEndTime (double _endTime)`
 - `double getEndTime () const`
 - `List< ODEfunction * > * getODEfunctions () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
 - static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool loadInstance (std::map< std::string, std::string > *fields)
 - virtual std::map< std::string, std::string > * saveInstance ()
 - virtual bool check (std::string *errorMessage)

Additional Inherited Members

8.82.1 Detailed Description

Definition at line 34 of file [OLD_ODElement.h](#).

8.82.2 Constructor & Destructor Documentation

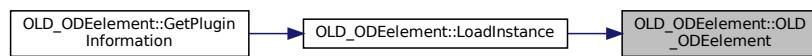
8.82.2.1 OLD_ODElement() `OLD_ODElement::OLD_ODElement (`
 `Model * model,`
 `std::string name = "")`

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

```
00017     Util::TypeOf<OLD_ODElement>(), name) {  
00018     //elems = elems;  
00019 }
```

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

Here is the caller graph for this function:



8.82.2.2 ~OLD_ODElement() `virtual OLD_ODElement::~OLD_ODElement () [virtual], [default]`

8.82.3 Member Function Documentation

8.82.3.1 `_check()` `bool OLD_ODElement::_check (std::string * errorMessage) [protected], [virtual]`

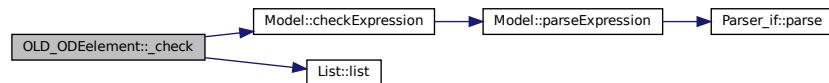
Reimplemented from [ModelElement](#).

Definition at line 78 of file [OLD_ODElement.cpp](#).

```
00078
00079     bool result = true;
00080     unsigned short i = 0;
00081     ODEFUNCTION* func;
00082     for (std::list<ODEFUNCTION*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->list()->end(); it++) {
00083         func = (*it);
00084         result &= \_parentModel->checkExpression\(func->expression, "expression\[" + std::to\_string\(i++\) + "\]", errorMessage\);
00085     }
00086     return result;
00087 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), [ODEfunction::expression](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.82.3.2 `_loadInstance()` `bool OLD_ODElement::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 70 of file [OLD_ODElement.cpp](#).

```
00070
00071     return ModelElement::_loadInstance(fields);
00072 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.82.3.3 `_saveInstance()` `std::map< std::string, std::string > * OLD_ODElement::_saveInstance() [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 74 of file [OLD_ODElement.cpp](#).

```
00074
00075     return ModelElement::_saveInstance();
00076 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.82.3.4 `getEndTime()` `double OLD_ODElement::getEndTime() const`

Definition at line 47 of file [OLD_ODElement.cpp](#).

```
00047
00048     return _endTime;
00049 }
```

8.82.3.5 `getODEfunctions()` `List< ODEfunction * > * OLD_ODElement::getODEfunctions() const`

Definition at line 51 of file [OLD_ODElement.cpp](#).

```
00051
00052     return _ODEfunctions;
00053 }
```

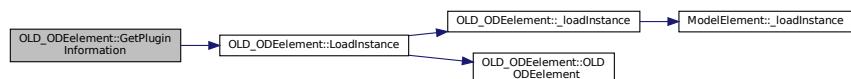
8.82.3.6 `GetPluginInformation()` `PluginInformation * OLD_ODElement::GetPluginInformation() [static]`

Definition at line 55 of file [OLD_ODElement.cpp](#).

```
00055
00056     PluginInformation* info = new PluginInformation(Util::TypeOf<OLD_ODElement>(),
00057     &OLD_ODElement::LoadInstance);
00058     return info;
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.82.3.7 `getStepH()` double OLD_ODEelement::getStepH () const

Definition at line 39 of file [OLD_ODElement.cpp](#).

```
00039     {
00040         return _stepH;
00041     }
```

8.82.3.8 `LoadInstance()` ModelElement * OLD_ODEelement::LoadInstance (

```
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

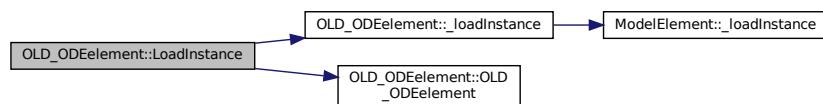
Definition at line 60 of file [OLD_ODElement.cpp](#).

```
00060     {
00061         OLD_ODElement* newElement = new OLD_ODElement(model);
00062         try {
00063             newElement->_loadInstance(fields);
00064         } catch (const std::exception& e) {
00065         }
00066     }
00067     return newElement;
00068 }
```

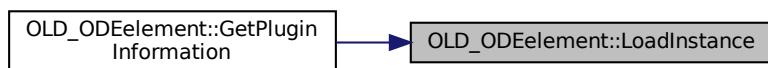
References [_loadInstance\(\)](#), and [OLD_ODElement\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.82.3.9 `setEndTime()` void OLD_ODEelement::setEndTime (

```
    double _endTime )
```

Definition at line 43 of file [OLD_ODElement.cpp](#).

```
00043     {
00044         this->_endTime = _endTime;
00045     }
```

8.82.3.10 setStepH() void OLD_ODElement::setStepH (double _h)

Definition at line 35 of file [OLD_ODElement.cpp](#).

```
00035     {
00036         this->_stepH = _h;
00037     }
```

8.82.3.11 show() std::string OLD_ODElement::show () [virtual]

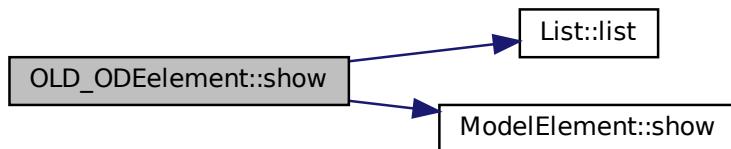
Reimplemented from [ModelElement](#).

Definition at line 21 of file [OLD_ODElement.cpp](#).

```
00021     {
00022         std::string txt = ModelElement::show();
00023         unsigned short i = 0;
00024         for (std::list<ODEfunction*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->list()->end(); it++) {
00025             txt += ",func[" + std::to_string(i) + "]=" + (*it)->expression + "\\",(func[" +
00026             std::to_string(i) + "][" + std::to_string((*it)->initialPoint) + "]=" +
00027             std::to_string((*it)->initialValue) + ")";
00028         }
00029         return txt;
00030     }
```

References [List< T >::list\(\)](#), and [ModelElement::show\(\)](#).

Here is the call graph for this function:



8.82.3.12 solve() double OLD_ODElement::solve ()

Definition at line 30 of file [OLD_ODElement.cpp](#).

```
00030     {
00032         return 0.0; // dummy
00033     }
```

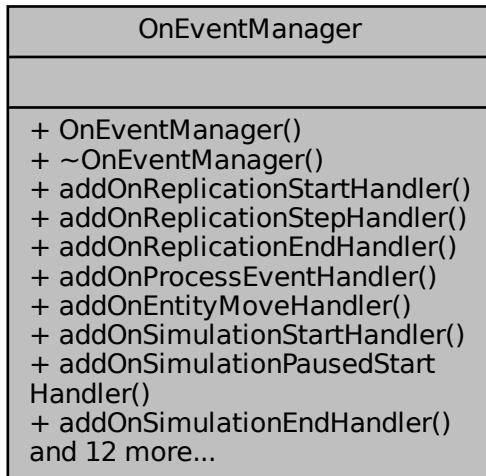
The documentation for this class was generated from the following files:

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

8.83 OnEventManager Class Reference

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

Collaboration diagram for OnEventManager:



Public Member Functions

- `OnEventManager ()`
- `virtual ~OnEventManager ()=default`
- `void addOnReplicationStartHandler (simulationEventHandler EventHandler)`
- `void addOnReplicationStepHandler (simulationEventHandler EventHandler)`
- `void addOnReplicationEndHandler (simulationEventHandler EventHandler)`
- `void addOnProcessEventHandler (simulationEventHandler EventHandler)`
- `void addOnEntityMoveHandler (simulationEventHandler EventHandler)`
- `void addOnSimulationStartHandler (simulationEventHandler EventHandler)`
- `void addOnSimulationPausedStartHandler (simulationEventHandler EventHandler)`
- `void addOnSimulationEndHandler (simulationEventHandler EventHandler)`
- `void addOnEntityRemoveHandler (simulationEventHandler EventHandler)`
- `void addOnBreakpointHandler (simulationEventHandler EventHandler)`
- template<typename Class >
`void addOnProcessEventHandler (Class *object, void(Class::*function)(SimulationEvent *))`
- `void NotifyReplicationStartHandlers (SimulationEvent *se)`
- `void NotifyReplicationStepHandlers (SimulationEvent *se)`
- `void NotifyReplicationEndHandlers (SimulationEvent *se)`
- `void NotifyProcessEventHandlers (SimulationEvent *se)`
- `void NotifyEntityMoveHandlers (SimulationEvent *se)`
- `void NotifySimulationStartHandlers (SimulationEvent *se)`
- `void NotifySimulationPausedStartHandlers (SimulationEvent *se)`
- `void NotifySimulationEndHandlers (SimulationEvent *se)`
- `void NotifyBreakpointHandlers (SimulationEvent *se)`

8.83.1 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.

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

8.83.2 Constructor & Destructor Documentation

8.83.2.1 `OnEventManager()` `OnEventManager::OnEventManager ()`

Definition at line 18 of file `OnEventManager.cpp`.

```
00018 {  
00019 }
```

8.83.2.2 `~OnEventManager()` `virtual OnEventManager::~OnEventManager () [virtual], [default]`

8.83.3 Member Function Documentation

8.83.3.1 `addOnBreakpointHandler()` `void OnEventManager::addOnBreakpointHandler (simulationEventHandler EventHandler)`

Definition at line 58 of file `OnEventManager.cpp`.

```
00058 {  
00059     _addOnHandler(_onBreakpointHandlers, EventHandler);  
00060 }
```

8.83.3.2 `addOnEntityMoveHandler()` `void OnEventManager::addOnEntityMoveHandler (simulationEventHandler EventHandler)`

Definition at line 38 of file `OnEventManager.cpp`.

```
00038 {  
00039     _addOnHandler(_onEntityMoveHandlers, EventHandler);  
00040 }
```

8.83.3.3 `addOnEntityRemoveHandler()` `void OnEventManager::addOnEntityRemoveHandler (simulationEventHandler EventHandler)`

8.83.3.4 addOnProcessEventHandler() [1/2] `template<typename Class >`

```
void OnEventManager::addOnProcessEventHandler (
    Class * object,
    void(Class::*)(SimulationEvent *) function )
```

Definition at line 106 of file [OnEventManager.h](#).

```
00106
00107     {
00108         simulationEventHandlerMethod handlerMethod = std::bind(function, object,
00109             std::placeholders::_1);
00110         // \todo: if handlerMethod already insert, should not insert it again. Problem to solve <...>
00111         for function
00112             //if (_onProcessEventHandlerMethods->find(handlerMethod) ==
00113             _onProcessEventHandlerMethods->list()->end())
00114             this->_onProcessEventHandlerMethods->insert(handlerMethod);
00115             // trying unique to solve the issue
00116             //this->_onProcessEventHandlerMethods->list()->unique(); // does not work
00117             // \todo: probably to override == operator for type simulationEventHandlerMethod
00118 // ...
00119 }
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



8.83.3.5 addOnProcessEventHandler() [2/2] `void OnEventManager::addOnProcessEventHandler (` `simulationEventHandler EventHandler)`

Definition at line 34 of file [OnEventManager.cpp](#).

```
00034
00035     _addOnHandler(_onEntityMoveHandlers, EventHandler);
00036 }
```

Referenced by [BaseConsoleGenesysApplication::setDefaultEventHandlers\(\)](#).

Here is the caller graph for this function:



8.83.3.6 addOnReplicationEndHandler() void OnEventManager::addOnReplicationEndHandler (simulationEventHandler EventHandler)

Definition at line 42 of file [OnEventManager.cpp](#).

```
00042     _addOnHandler(_onReplicationEndHandlers, EventHandler);  
00043 }  
00044 }
```

8.83.3.7 addOnReplicationStartHandler() void OnEventManager::addOnReplicationStartHandler (simulationEventHandler EventHandler)

Definition at line 26 of file [OnEventManager.cpp](#).

```
00026     _addOnHandler(_onReplicationStartHandlers, EventHandler);  
00027 }  
00028 }
```

8.83.3.8 addOnReplicationStepHandler() void OnEventManager::addOnReplicationStepHandler (simulationEventHandler EventHandler)

Definition at line 30 of file [OnEventManager.cpp](#).

```
00030     _addOnHandler(_onReplicationStepHandlers, EventHandler);  
00031 }  
00032 }
```

8.83.3.9 addOnSimulationEndHandler() void OnEventManager::addOnSimulationEndHandler (simulationEventHandler EventHandler)

Definition at line 54 of file [OnEventManager.cpp](#).

```
00054     _addOnHandler(_onSimulationEndHandlers, EventHandler);  
00055 }  
00056 }
```

8.83.3.10 addOnSimulationPausedStartHandler() void OnEventManager::addOnSimulationPausedStartHandler (simulationEventHandler EventHandler)

Definition at line 50 of file [OnEventManager.cpp](#).

```
00050     _addOnHandler(_onSimulationPausedStartHandlers, EventHandler);  
00051 }  
00052 }
```

8.83.3.11 addOnSimulationStartHandler() void OnEventManager::addOnSimulationStartHandler (simulationEventHandler EventHandler)

Definition at line 46 of file [OnEventManager.cpp](#).

```
00046     _addOnHandler(_onSimulationStartHandlers, EventHandler);  
00047 }  
00048 }
```

8.83.3.12 NotifyBreakpointHandlers() void OnEventManager::NotifyBreakpointHandlers (SimulationEvent * se)

Definition at line 108 of file [OnEventManager.cpp](#).

```
00108     this->_NotifyHandlers(this->_onBreakpointHandlers, se);  
00109 }  
00110 }
```

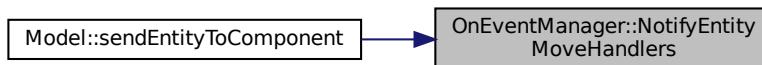
8.83.3.13 NotifyEntityMoveHandlers() void OnEventManager::NotifyEntityMoveHandlers (SimulationEvent * se)

Definition at line 87 of file [OnEventManager.cpp](#).

```
00087     this->_NotifyHandlers(this->_onEntityMoveHandlers, se);  
00088 }  
00089 }
```

Referenced by [Model::sendEntityToComponent\(\)](#).

Here is the caller graph for this function:



8.83.3.14 NotifyProcessEventHandlers() void OnEventManager::NotifyProcessEventHandlers (SimulationEvent * se)

Definition at line 91 of file [OnEventManager.cpp](#).

```
00091     {  
00092         this->_NotifyHandlers(this->_onProcessEventHandlers, se);  
00093         this->_NotifyHandlerMethods(this->_onEventHandlerMethods, se);  
00094     }
```

8.83.3.15 NotifyReplicationEndHandlers() void OnEventManager::NotifyReplicationEndHandlers (SimulationEvent * se)

Definition at line 83 of file [OnEventManager.cpp](#).

```
00083     {  
00084         this->_NotifyHandlers(this->_onReplicationEndHandlers, se);  
00085     }
```

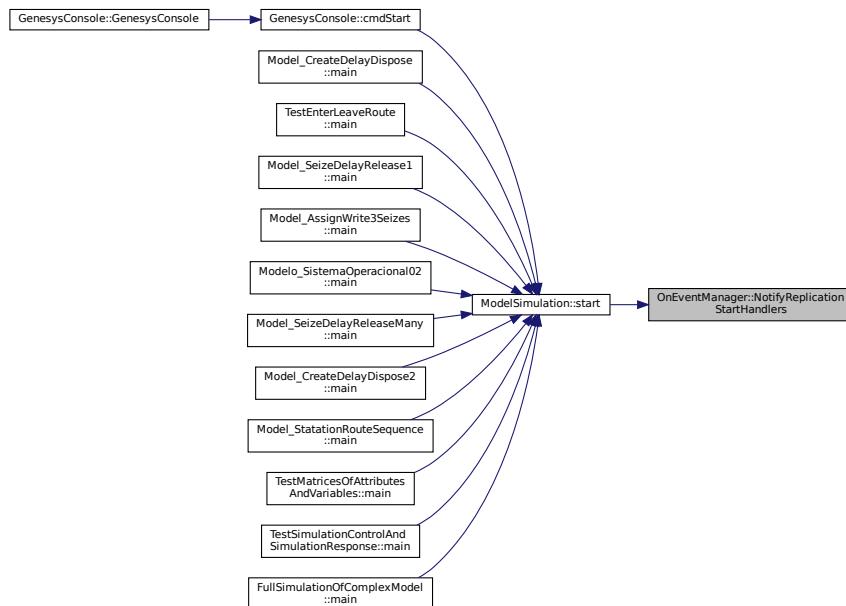
8.83.3.16 NotifyReplicationStartHandlers() void OnEventManager::NotifyReplicationStartHandlers (SimulationEvent * se)

Definition at line 74 of file [OnEventManager.cpp](#).

```
00074
00075     this->_NotifyHandlers(this->_onReplicationStartHandlers, se);
00076
00077 }
```

Referenced by [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



8.83.3.17 NotifyReplicationStepHandlers() void OnEventManager::NotifyReplicationStepHandlers (SimulationEvent * se)

Definition at line 79 of file [OnEventManager.cpp](#).

```
00079
00080     this->_NotifyHandlers(this->_onReplicationStepHandlers, se);
00081 }
```

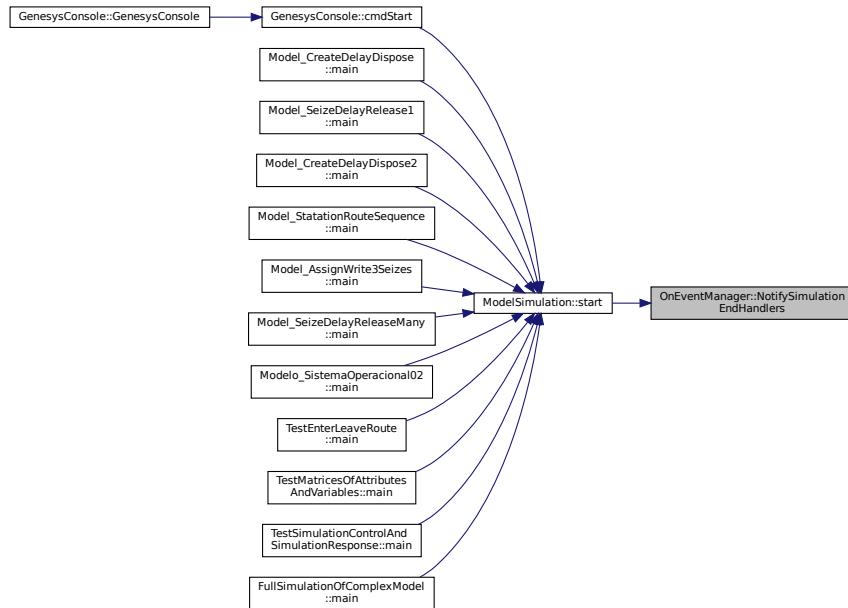
8.83.3.18 NotifySimulationEndHandlers() void OnEventManager::NotifySimulationEndHandlers (SimulationEvent * se)

Definition at line 104 of file [OnEventManager.cpp](#).

```
00104
00105     this->_NotifyHandlers(this->_onSimulationEndHandlers, se);
00106 }
```

Referenced by [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



8.83.3.19 NotifySimulationPausedStartHandlers() void OnEventManager::NotifySimulationPausedStartHandlers (SimulationEvent * se)

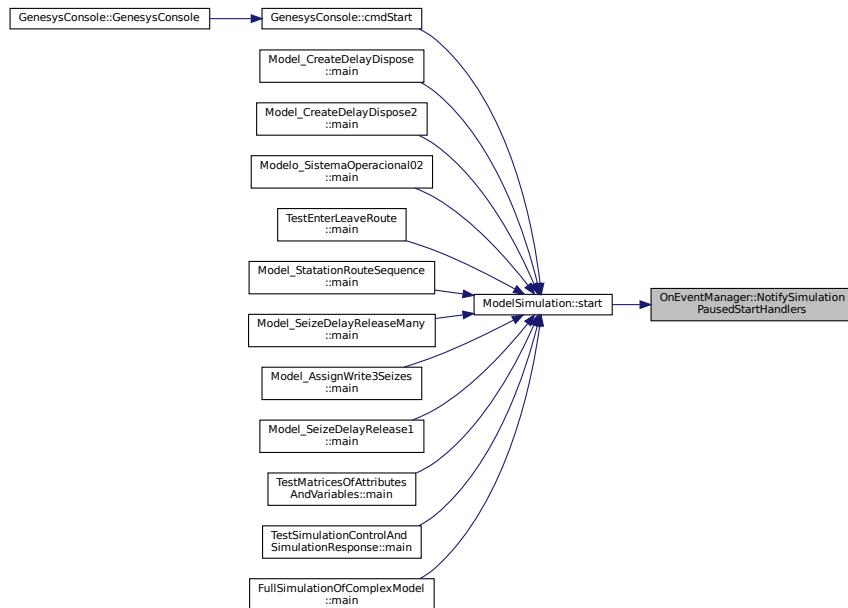
Definition at line 100 of file [OnEventManager.cpp](#).

```

00100
00101     this->_NotifyHandlers(this->_onSimulationPausedStartHandlers, se);
00102 }
```

Referenced by [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



8.83.3.20 NotifySimulationStartHandlers() void OnEventManager::NotifySimulationStartHandlers ([SimulationEvent * se](#))

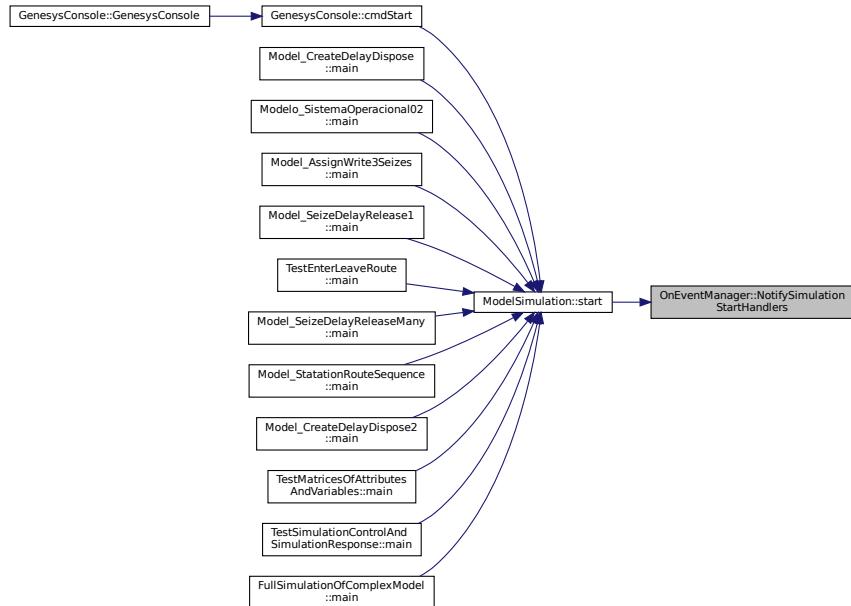
Definition at line 96 of file [OnEventManager.cpp](#).

```

00096
00097     this->_NotifyHandlers(this->_onSimulationStartHandlers, se);
00098 }
```

Referenced by [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



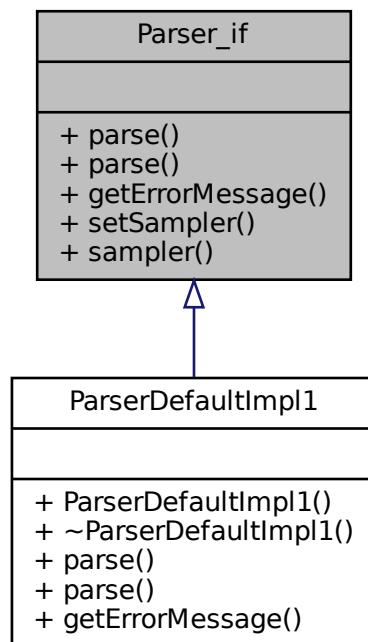
The documentation for this class was generated from the following files:

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

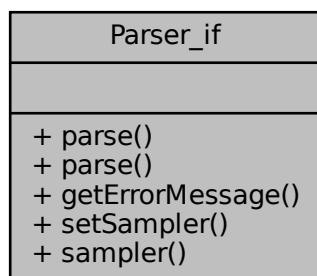
8.84 Parser_if Class Reference

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

Inheritance diagram for Parser_if:



Collaboration 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
- virtual void `setSampler` (Sampler_if *sampler)=0
- virtual Sampler_if * `sampler` () const =0

8.84.1 Detailed Description

Definition at line 20 of file [Parser_if.h](#).

8.84.2 Member Function Documentation

8.84.2.1 `getErrorMessage()` `virtual std::string* Parser_if::getErrorMessage () [pure virtual]`

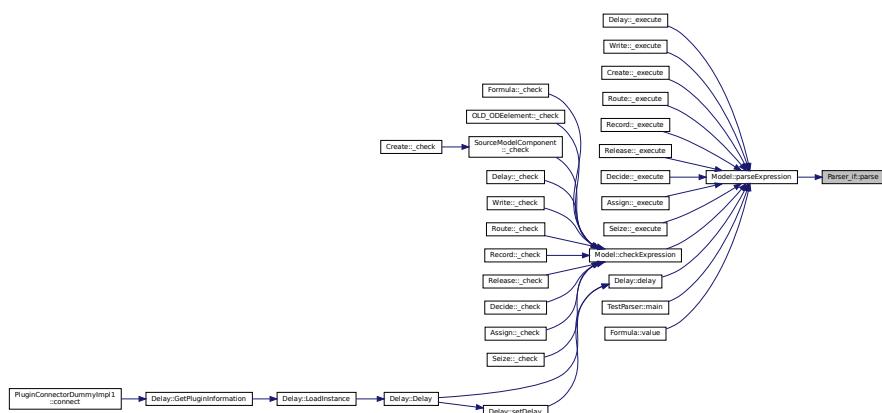
Implemented in [ParserDefaultImpl1](#).

8.84.2.2 `parse()` [1/2] `virtual double Parser_if::parse (const std::string expression) [pure virtual]`

Implemented in [ParserDefaultImpl1](#).

Referenced by [Model::parseExpression\(\)](#).

Here is the caller graph for this function:



8.84.2.3 `parse()` [2/2] `virtual double Parser_if::parse (const std::string expression, bool * success, std::string * errorMessage) [pure virtual]`

Implemented in [ParserDefaultImpl1](#).

8.84.2.4 sampler() virtual `Sampler_if*` `Parser_if::sampler() const [pure virtual]`

8.84.2.5 setSampler() virtual void `Parser_if::setSampler(`
`Sampler_if * sampler) [pure virtual]`

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

- `Parser_if.h`

8.85 ParserChangesInformation Class Reference

```
#include <ParserChangesInformation.h>
```

Collaboration diagram for ParserChangesInformation:

ParserChangesInformation
+ ParserChangesInformation() + ~ParserChangesInformation() + getLexicalElementToRemove() + getLexicalElementToAdd() + getLexIncludeToRemove() + getLexIncludeToAdd() + getProductionToRemove() + getProductionToAdd() + getTypeobjToRemove() + getTypeobjToAdd() + getTokensToRemove() + getTokensToAdd()

Public Types

- `typedef std::string token`
- `typedef std::string typeobj`
- `typedef std::string lexinclude`
- `typedef std::pair< std::string, std::string > production`
- `typedef std::pair< std::string, std::string > lexicalelement`

Public Member Functions

- `ParserChangesInformation ()`
- `virtual ~ParserChangesInformation ()=default`
- `std::list< ParserChangesInformation::lexicalelement * > * getLexicalElementToRemove () const`
- `std::list< ParserChangesInformation::lexicalelement * > * getLexicalElementToAdd () const`
- `std::list< ParserChangesInformation::lexinclude * > * getLexIncludeToRemove () const`
- `std::list< ParserChangesInformation::lexinclude * > * getLexIncludeToAdd () const`
- `std::list< ParserChangesInformation::production * > * getProductionToRemove () const`
- `std::list< ParserChangesInformation::production * > * getProductionToAdd () const`
- `std::list< ParserChangesInformation::typeobj * > * getTypeobjToRemove () const`
- `std::list< ParserChangesInformation::typeobj * > * getTypeobjToAdd () const`
- `std::list< ParserChangesInformation::token * > * getTokensToRemove () const`
- `std::list< ParserChangesInformation::token * > * getTokensToAdd () const`

8.85.1 Detailed Description

Definition at line 20 of file [ParserChangesInformation.h](#).

8.85.2 Member Typedef Documentation

8.85.2.1 `lexicalelement` `typedef std::pair<std::string, std::string> ParserChangesInformation::lexicalelement`

Definition at line 26 of file [ParserChangesInformation.h](#).

8.85.2.2 `lexinclude` `typedef std::string ParserChangesInformation::lexinclude`

Definition at line 24 of file [ParserChangesInformation.h](#).

8.85.2.3 `production` `typedef std::pair<std::string, std::string> ParserChangesInformation::production`

Definition at line 25 of file [ParserChangesInformation.h](#).

8.85.2.4 `token` `typedef std::string ParserChangesInformation::token`

Definition at line 22 of file [ParserChangesInformation.h](#).

8.85.2.5 typeobj `typedef std::string ParserChangesInformation::typeobj`

Definition at line 23 of file [ParserChangesInformation.h](#).

8.85.3 Constructor & Destructor Documentation**8.85.3.1 ParserChangesInformation()** `ParserChangesInformation::ParserChangesInformation ()`

Definition at line 16 of file [ParserChangesInformation.cpp](#).

```
00016 {  
00017 }
```

8.85.3.2 ~ParserChangesInformation() `virtual ParserChangesInformation::~ParserChangesInformation () [virtual], [default]`**8.85.4 Member Function Documentation****8.85.4.1 getLexicalElementToAdd()** `std::list< ParserChangesInformation::lexicalelement * > * ParserChangesInformation::getLexicalElementToAdd () const`

Definition at line 23 of file [ParserChangesInformation.cpp](#).

```
00023 {  
00024     return _lexicalElementToAdd;  
00025 }
```

8.85.4.2 getLexicalElementToRemove() `std::list< ParserChangesInformation::ParserChangesInformation::lexicalelement * > * ParserChangesInformation::getLexicalElementToRemove () const`

Definition at line 19 of file [ParserChangesInformation.cpp](#).

```
00019 {  
00020     return _lexicalElementToRemove;  
00021 }
```

8.85.4.3 getLexIncludeToAdd() `std::list< ParserChangesInformation::lexinclude * > * ParserChangesInformation::getLexIncludeToAdd () const`

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

```
00031 {  
00032     return _lexIncludeToAdd;  
00033 }
```

8.85.4.4 getLexIncludeToRemove() std::list< ParserChangesInformation::lexinclude * > * ParserChangesInformation::getLexIncludeToRemove () const

Definition at line 27 of file ParserChangesInformation.cpp.

```
00027
00028     return _lexIncludeToRemove;
00029 }
```

8.85.4.5 getProductionToAdd() std::list< ParserChangesInformation::production * > * ParserChangesInformation::getProductionToAdd () const

Definition at line 39 of file ParserChangesInformation.cpp.

```
00039
00040     return _productionToAdd;
00041 }
```

8.85.4.6 getProductionToRemove() std::list< ParserChangesInformation::production * > * ParserChangesInformation::getProductionToRemove () const

Definition at line 35 of file ParserChangesInformation.cpp.

```
00035
00036     return _productionToRemove;
00037 }
```

8.85.4.7 getTokensToAdd() std::list< ParserChangesInformation::token * > * ParserChangesInformation::getTokensToAdd () const

Definition at line 55 of file ParserChangesInformation.cpp.

```
00055
00056     return _tokensToAdd;
00057 }
```

8.85.4.8 getTokensToRemove() std::list< ParserChangesInformation::token * > * ParserChangesInformation::getTokensToRemove () const

Definition at line 51 of file ParserChangesInformation.cpp.

```
00051
00052     return _tokensToRemove;
00053 }
```

8.85.4.9 getTypeobjToAdd() std::list< ParserChangesInformation::typeobj * > * ParserChangesInformation::getTypeobjToAdd () const

Definition at line 47 of file ParserChangesInformation.cpp.

```
00047
00048     return _typeobjToAdd;
00049 }
```

```
8.85.4.10 getTypeobjToRemove() std::list< ParserChangesInformation::typeobj * > * Parser->
ChangesInformation::getTypeobjToRemove ( ) const
```

Definition at line 43 of file [ParserChangesInformation.cpp](#).

```
00043
00044     return _typeobjToRemove;
00045 }
```

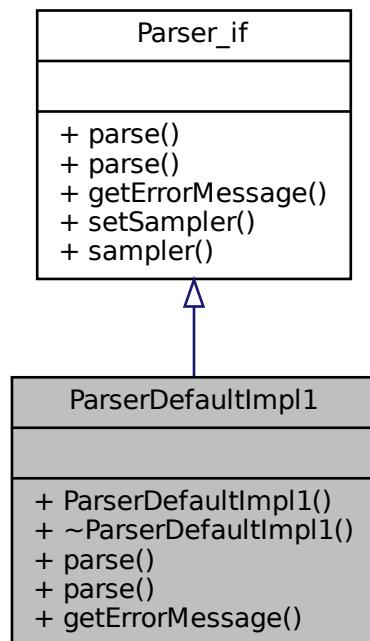
The documentation for this class was generated from the following files:

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

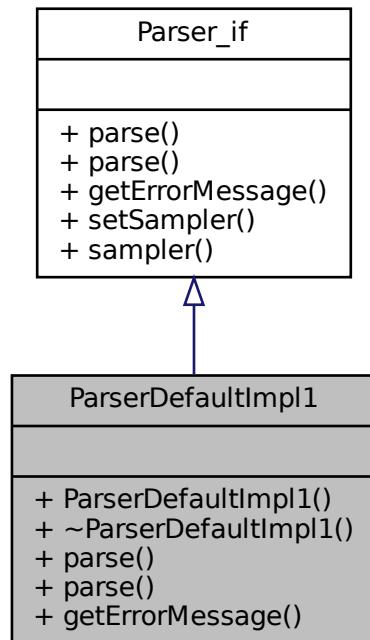
8.86 ParserDefaultImpl1 Class Reference

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

Inheritance diagram for ParserDefaultImpl1:



Collaboration diagram for ParserDefaultImpl1:



Public Member Functions

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

8.86.1 Detailed Description

Definition at line 21 of file [ParserDefaultImpl1.h](#).

8.86.2 Constructor & Destructor Documentation

8.86.2.1 ParserDefaultImpl1()

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

Definition at line 18 of file [ParserDefaultImpl1.cpp](#).

```
00018     _model = model;
00019
00020 }
```

8.86.2.2 ~ParserDefaultImpl1() virtual ParserDefaultImpl1::~ParserDefaultImpl1 () [virtual], [default]

8.86.3 Member Function Documentation

8.86.3.1 getErrorMessage() std::string * ParserDefaultImpl1::getErrorMessage () [virtual]

Implements [Parser_if](#).

Definition at line 27 of file [ParserDefaultImpl1.cpp](#).

```
00027
00028     std::string* errorMsg = new std::string();
00029     return errorMsg; /* @ \todo: */
00030 }
```

8.86.3.2 parse() [1/2] double ParserDefaultImpl1::parse (
 const std::string expression) [virtual]

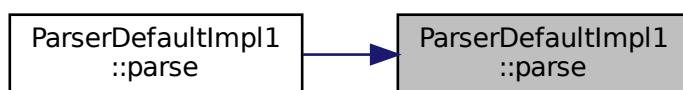
Implements [Parser_if](#).

Definition at line 22 of file [ParserDefaultImpl1.cpp](#).

```
00022
00023     double result = std::atof(expression.c_str()); // may throw exception
00024     return result;
00025 }
```

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

Here is the caller graph for this function:



```
8.86.3.3 parse() [2/2] double ParserDefaultImpl1::parse (
    const std::string expression,
    bool * success,
    std::string * errorMessage ) [virtual]
```

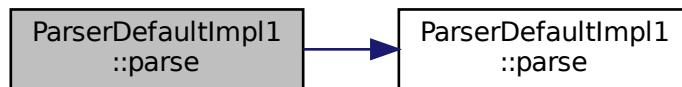
Implements [Parser_if](#).

Definition at line 32 of file [ParserDefaultImpl1.cpp](#).

```
00032 {
00033     try {
00034         double result = this->parse(expression);
00035         std::string temp(""); /* \todo: CHECK SCOPE OF VARIABLE */
00036         errorMessage = &temp;
00037         *success = true;
00038         return result;
00039     } catch (...) {
00040         std::string temp("Error parsing...");
00041         errorMessage = &temp;
00042         *success = false;
00043         return 0.0;
00044     }
00045 }
```

References [parse\(\)](#).

Here is the call graph for this function:



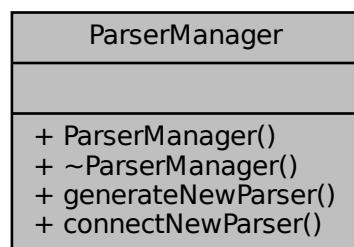
The documentation for this class was generated from the following files:

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

8.87 ParserManager Class Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager:



Classes

- struct [GenerateNewParserResult](#)
- struct [NewParser](#)

Public Member Functions

- [ParserManager \(\)](#)
- virtual [~ParserManager \(\)=default](#)
- [ParserManager::GenerateNewParserResult generateNewParser \(ParserChangesInformation *changes\)](#)
- bool [connectNewParser \(ParserManager::NewParser newParser\)](#)

8.87.1 Detailed Description

Definition at line 21 of file [ParserManager.h](#).

8.87.2 Constructor & Destructor Documentation

8.87.2.1 [ParserManager\(\)](#) ParserManager::ParserManager ()

Definition at line 18 of file [ParserManager.cpp](#).

```
00018     {
00019 }
```

8.87.2.2 [~ParserManager\(\)](#) virtual ParserManager::~ParserManager () [virtual], [default]

8.87.3 Member Function Documentation

8.87.3.1 [connectNewParser\(\)](#) bool ParserManager::connectNewParser (ParserManager::NewParser newParser)

8.87.3.2 [generateNewParser\(\)](#) ParserManager::GenerateNewParserResult ParserManager::generateNewParser (ParserChangesInformation * changes)

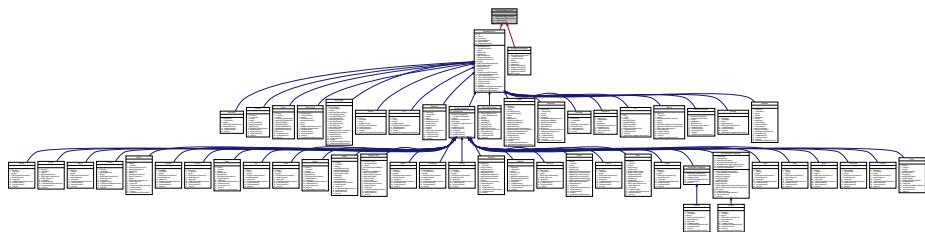
The documentation for this class was generated from the following files:

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

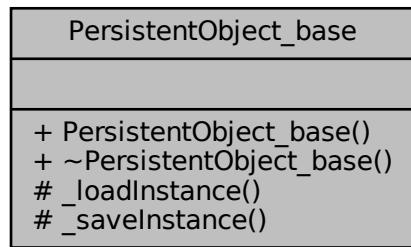
8.88 PersistentObject_base Class Reference

```
#include <PersistentObject_base.h>
```

Inheritance diagram for PersistentObject_base:



Collaboration diagram for PersistentObject_base:



Public Member Functions

- [PersistentObject_base \(\)](#)
- virtual [~PersistentObject_base \(\)](#)=default

Protected Member Functions

- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)=0](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)=0](#)

8.88.1 Detailed Description

Definition at line 19 of file [PersistentObject_base.h](#).

8.88.2 Constructor & Destructor Documentation

8.88.2.1 PersistentObject_base() PersistentObject_base::PersistentObject_base () [inline]

Definition at line 22 of file [PersistentObject_base.h](#).

```
00022     {  
00023 }
```

8.88.2.2 ~PersistentObject_base() virtual PersistentObject_base::~PersistentObject_base () [virtual], [default]**8.88.3 Member Function Documentation****8.88.3.1 _loadInstance()** virtual bool PersistentObject_base::_loadInstance (std::map< std::string, std::string > * fields) [protected], [pure virtual]

Implemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Sequence](#), [Decide](#), [Hold](#), [Schedule](#), [Record](#), [Release](#), [Route](#), [Station](#), [Create](#), [Failure](#), [File](#), [PickStation](#), [Attribute](#), [Set](#), [Search](#), [Access](#), [Batch](#), [ModelElement](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [ModelComponent](#), [EntityType](#), [OLD_ODElement](#), [Start](#), [PickUp](#), [Signal](#), [SourceModelComponent](#), [Dispose](#), [Exit](#), [Remove](#), [Stop](#), [Storage](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [StatisticsCollector](#), [Counter](#), [Formula](#), [SeizableItemRequest](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

8.88.3.2 _saveInstance() virtual std::map<std::string, std::string>* PersistentObject_base::_saveInstance () [protected], [pure virtual]

Implemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Sequence](#), [Decide](#), [Hold](#), [Schedule](#), [Record](#), [Release](#), [Route](#), [Station](#), [Create](#), [Failure](#), [File](#), [PickStation](#), [Attribute](#), [Set](#), [Search](#), [Access](#), [Batch](#), [ModelElement](#), [Store](#), [Write](#), [Delay](#), [Match](#), [Unstore](#), [DropOff](#), [EntityType](#), [OLD_ODElement](#), [SourceModelComponent](#), [Start](#), [Dispose](#), [ModelComponent](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [Storage](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [StatisticsCollector](#), [Counter](#), [Formula](#), [SeizableItemRequest](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

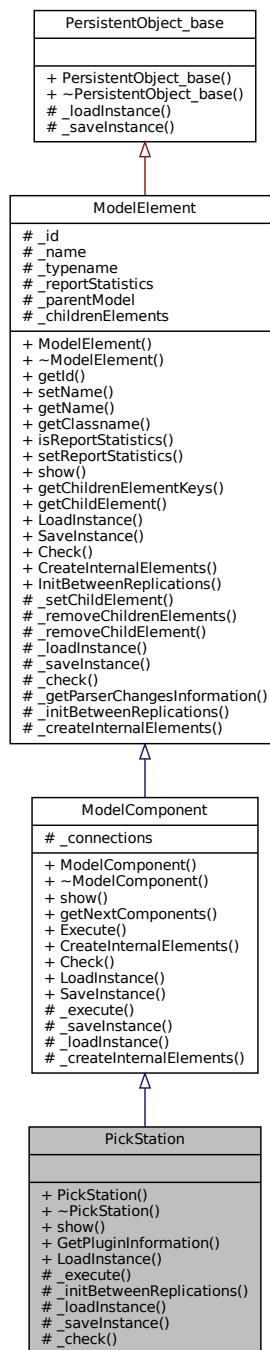
The documentation for this class was generated from the following file:

- [PersistentObject_base.h](#)

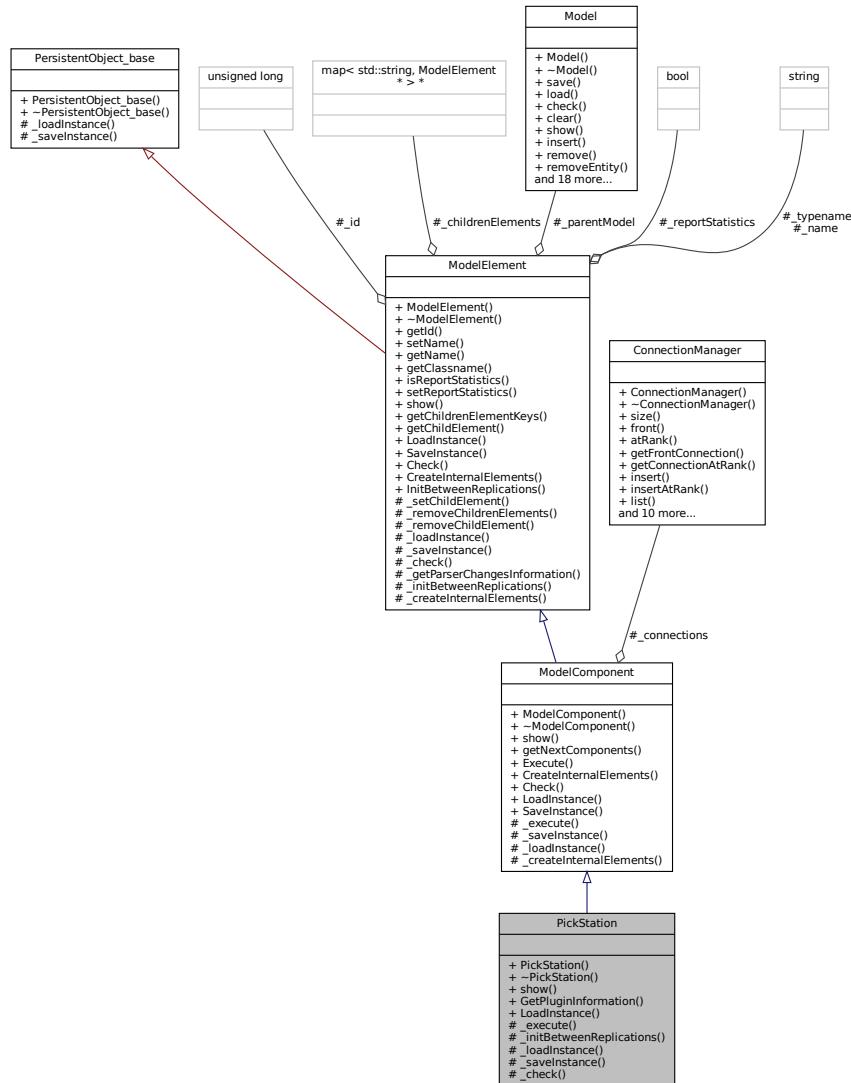
8.89 PickStation Class Reference

```
#include <PickStation.h>
```

Inheritance diagram for PickStation:



Collaboration diagram for PickStation:



Public Member Functions

- **PickStation (Model *model, std::string name="")**
- virtual **~PickStation ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.89.1 Detailed Description

PickStation module DESCRIPTION The **PickStation** module allows an entity to select a particular station from the multiple stations specified. This module picks among the group of stations based on the selection logic defined with the module. The entity may then route, transport, convey, or connect to the station specified. If the method chosen is connect, the selected station is assigned to an entity attribute. The station selection process is based on the minimum or maximum value of a variety of system variables and expressions. TYPICAL USES A part sent to a processing station based on machine's availability at each station A loan application sent to a set of loan officers based on the number sent to each officer A customer selecting among cashier lines based on the least number waiting in each line PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Test Condition Test condition to use for the station selection process, either Minimum or Maximum. Number En **Route** to **Station** The number of entities transferring to the station is considered in the station selection process. Number in **Queue** The number of entities in the queue at the station is considered in the station selection process. Number of Resources Busy The number of busy resources at the station is considered in the station selection process. Expression Determines if an additional user-defined expression is considered in the station selection process. Transfer Type Determines how an entity will be transferred out of this module to its next destination station—either **Route**, Convey, Transport, or Connect. Save **Attribute** Defines the name of the attribute that will store the station name that is selected, visible when the transfer method is Connect. **Route** Time Move time of the entity from its current station to the station determined through this module. Units Time units for route-time parameters.

Definition at line 62 of file [PickStation.h](#).

8.89.2 Constructor & Destructor Documentation

8.89.2.1 PickStation() `PickStation::PickStation (`
 `Model * model,`
 `std::string name = "")`

Definition at line 18 of file [PickStation.cpp](#).

```
00018     : ModelComponent (model,
00019         Util::TypeOf<PickStation>(), name) {
```

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

Here is the caller graph for this function:



8.89.2.2 ~PickStation() `virtual PickStation::~PickStation () [virtual], [default]`

8.89.3 Member Function Documentation

8.89.3.1 `_check()` `bool PickStation::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 57 of file [PickStation.cpp](#).

```
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
00061 }
```

8.89.3.2 `_execute()` `void PickStation::_execute (Entity * entity) [protected], [virtual]`

Implements [ModelComponent](#).

Definition at line 35 of file [PickStation.cpp](#).

```
00035     {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.89.3.3 `_initBetweenReplications()` `void PickStation::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 48 of file [PickStation.cpp](#).

```
00048     {
00049 }
```

8.89.3.4 `_loadInstance()` `bool PickStation::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

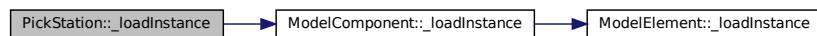
Definition at line 40 of file [PickStation.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.89.3.5 `_saveInstance()` `std::map< std::string, std::string > * PickStation::_saveInstance () [protected], [virtual]`

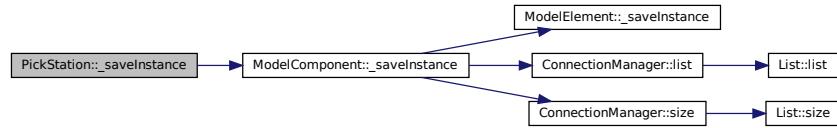
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [PickStation.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



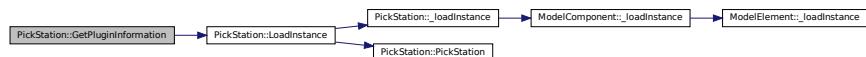
8.89.3.6 GetPluginInformation() `PluginInformation * PickStation::GetPluginInformation () [static]`

Definition at line 63 of file `PickStation.cpp`.

```
00063     {
00064         PluginInformation* info = new PluginInformation(Util::TypeOf<PickStation>(),
00065             &PickStation::LoadInstance);
00066         // ...
00067         return info;
00068     }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.89.3.7 LoadInstance() `ModelComponent * PickStation::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

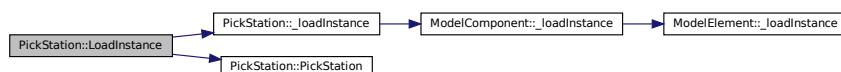
Definition at line 25 of file `PickStation.cpp`.

```
00025     {
00026         PickStation* newComponent = new PickStation(model);
00027         try {
00028             newComponent->_loadInstance(fields);
00029         } catch (const std::exception& e) {
00030         }
00031     }
00032     return newComponent;
00033 }
```

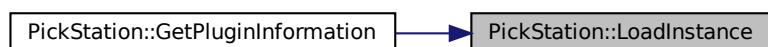
References [_loadInstance\(\)](#), and [PickStation\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.89.3.8 `show()` `std::string PickStation::show() [virtual]`

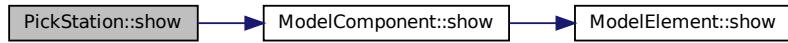
Reimplemented from [ModelComponent](#).

Definition at line 21 of file [PickStation.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



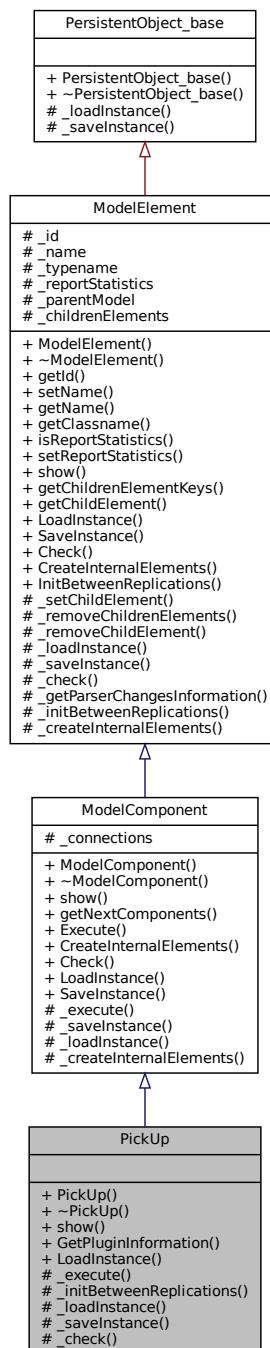
The documentation for this class was generated from the following files:

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

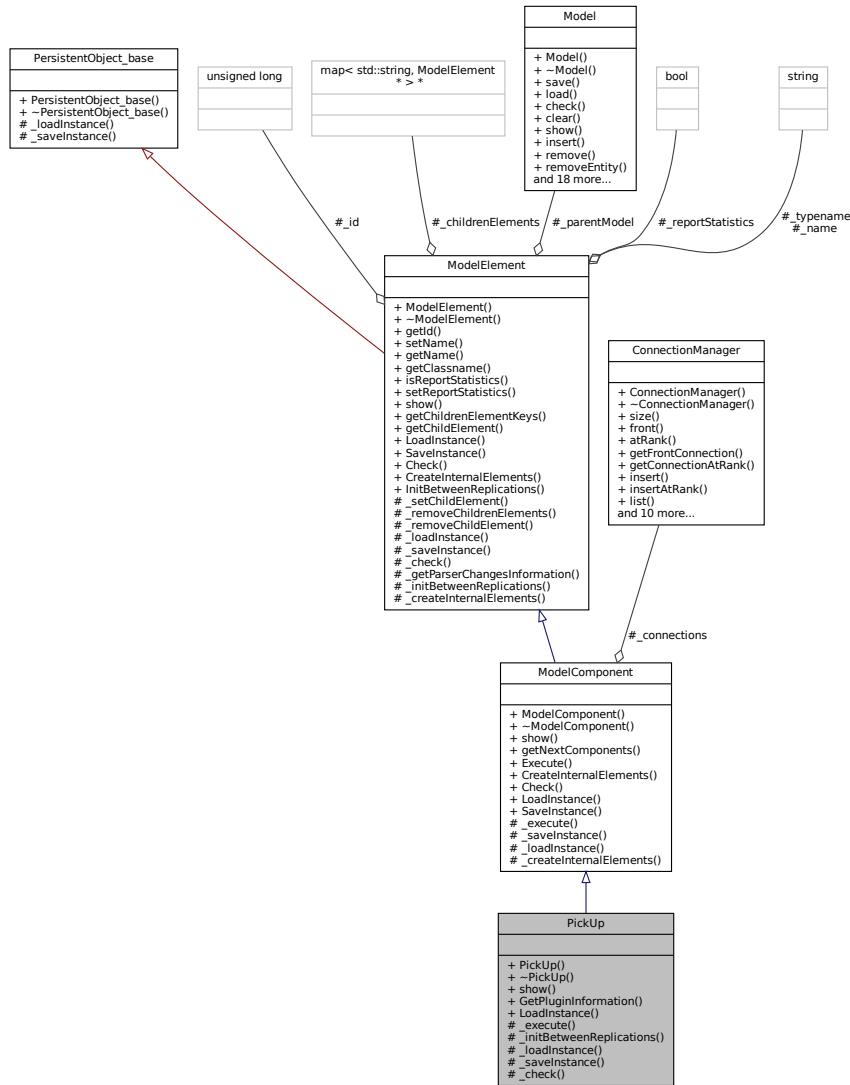
8.90 PickUp Class Reference

```
#include <PickUp.h>
```

Inheritance diagram for PickUp:



Collaboration diagram for PickUp:



Public Member Functions

- **PickUp** (**Model** *model, std::string name="")
- virtual **~PickUp** ()=default
- virtual std::string **show** ()

Static Public Member Functions

- static **PluginInformation** * **GetPluginInformation** ()
- static **ModelComponent** * **LoadInstance** (**Model** *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void **_execute** (**Entity** *entity)
- virtual void **_initBetweenReplications** ()
- virtual bool **_loadInstance** (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * **_savemode** ()
- virtual bool **_check** (std::string *errorMessage)

Additional Inherited Members

8.90.1 Detailed Description

Pickup module DESCRIPTION The Pickup module removes a number of consecutive entities from a given queue starting at a specified rank in the queue. The entities that are picked up are added to the end of the incoming entity's group. TYPICAL USES Gathering an order from various queue locations Gathering completed forms for an office order Picking up students at a bus stop for school PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Quantity Number of entities to pick up. Queue Name Name of the queue from which the entities will be picked up, starting at the specified rank. Starting Rank Starting rank of the entities to pick up from the queue, Queue Name.

Definition at line 38 of file [PickUp.h](#).

8.90.2 Constructor & Destructor Documentation

```
8.90.2.1 PickUp() PickUp::PickUp (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [PickUp.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<PickUp>(), name) {
00019 }
```

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

Here is the caller graph for this function:



```
8.90.2.2 ~PickUp() virtual PickUp::~PickUp ( ) [virtual], [default]
```

8.90.3 Member Function Documentation

8.90.3.1 `_check()` `bool PickUp::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 57 of file [PickUp.cpp](#).

```
00057     {  
00058     bool resultAll = true;  
00059     //...  
00060     return resultAll;  
00061 }
```

8.90.3.2 `_execute()` `void PickUp::_execute (Entity * entity) [protected], [virtual]`

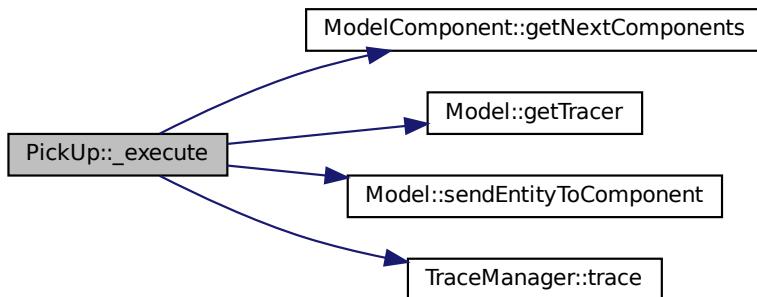
Implements [ModelComponent](#).

Definition at line 35 of file [PickUp.cpp](#).

```
00035     {  
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),  
0.0);  
00038 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent\(\)](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.90.3.3 `_initBetweenReplications()` `void PickUp::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 48 of file [PickUp.cpp](#).

```
00048     {  
00049 }
```

8.90.3.4 `_loadInstance()` `bool PickUp::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelComponent](#).

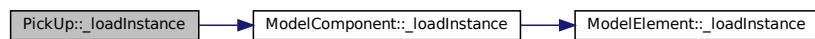
Definition at line 40 of file [PickUp.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

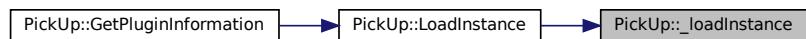
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.90.3.5 `_saveInstance()` `std::map< std::string, std::string > * PickUp::_saveInstance () [protected], [virtual]`

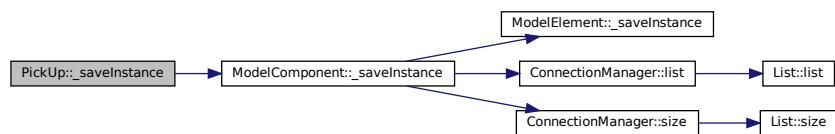
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [PickUp.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



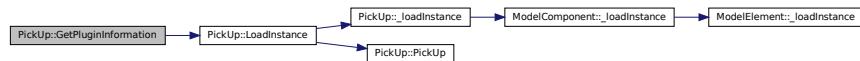
8.90.3.6 GetPluginInformation() `PluginInformation * PickUp::GetPluginInformation () [static]`

Definition at line 63 of file `PickUp.cpp`.

```
00063 {  
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<PickUp>(), &PickUp::LoadInstance);  
00065     // ...  
00066     return info;  
00067 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.90.3.7 LoadInstance() `ModelComponent * PickUp::LoadInstance (`

```
Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

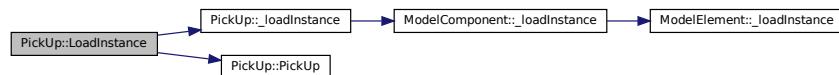
Definition at line 25 of file `PickUp.cpp`.

```
00025 {  
00026     PickUp* newComponent = new PickUp(model);  
00027     try {  
00028         newComponent->_loadInstance(fields);  
00029     } catch (const std::exception& e) {  
00030     }  
00031     return newComponent;  
00032 }  
00033 }
```

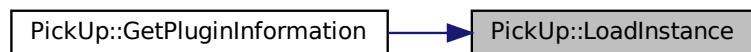
References `_loadInstance()`, and `PickUp()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.90.3.8 show() std::string PickUp::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [PickUp.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



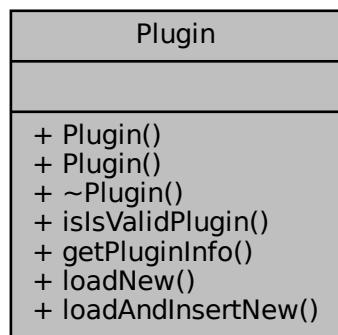
The documentation for this class was generated from the following files:

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

8.91 Plugin Class Reference

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

Collaboration diagram for Plugin:



Public Member Functions

- `Plugin (std::string filename_so_dll)`
- `Plugin (StaticGetPluginInformation getInformation)`
- `virtual ~Plugin ()=default`
- `bool isValidPlugin () const`
- `PluginInformation * getPluginInfo () const`
- `ModelElement * loadNew (Model *model, std::map< std::string, std::string > *fields)`
- `bool loadAndInsertNew (Model *model, std::map< std::string, std::string > *fields)`

8.91.1 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

Definition at line 30 of file `Plugin.h`.

8.91.2 Constructor & Destructor Documentation

8.91.2.1 Plugin() [1/2] `Plugin::Plugin (`
 `std::string filename_so_dll)`

8.91.2.2 Plugin() [2/2] `Plugin::Plugin (`
 `StaticGetPluginInformation getInformation)`

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

```
00023
00024     this->_StatMethodGetInformation = getInformation;
00025     try {
00026         PluginInformation* infos = _StatMethodGetInformation();
00027         this->_pluginInfo = infos;
00028         this->_isValidPlugin = true;
00029     } catch (...) {
00030         this->_isValidPlugin = false;
00031     }
00032 }
```

8.91.2.3 ~Plugin() `virtual Plugin::~Plugin () [virtual], [default]`

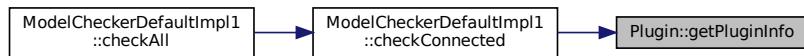
8.91.3 Member Function Documentation

8.91.3.1 getPluginInfo() `PluginInformation * Plugin::getPluginInfo () const`Definition at line 34 of file [Plugin.cpp](#).

```
00034
00035     return _pluginInfo;
00036 }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#).

Here is the caller graph for this function:

**8.91.3.2 isValidPlugin()** `bool Plugin::isValidPlugin () const`Definition at line 38 of file [Plugin.cpp](#).

```
00038
00039     return _isValidPlugin;
00040 }
```

8.91.3.3 loadAndInsertNew() `bool Plugin::loadAndInsertNew (`

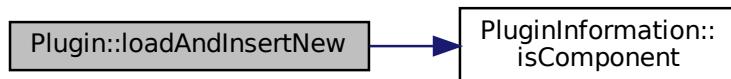
```
    Model * model,
    std::map< std::string, std::string > * fields )
```

Definition at line 56 of file [Plugin.cpp](#).

```
00056
00057     if (this->_pluginInfo->isComponent()) {
00058         ModelComponent* newComp = _loadNewComponent(model, fields);
00059         if (newComp != nullptr) {
00060             //model->getTraceManager()->trace(newComp->show());
00061             return true; //model->components()->insert(newComp);
00062         }
00063     } else {
00064         ModelElement* newElem = _loadNewElement(model, fields);
00065         if (newElem != nullptr) {
00066             //model->getTraceManager()->trace(newElem->show());
00067             return true; //model->elements()->insert(this->_pluginInfo->pluginTypename(), newElem);
00068         }
00069     }
00070     return false;
00071 }
```

References [PluginInformation::isComponent\(\)](#).

Here is the call graph for this function:



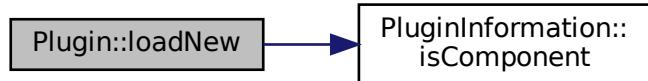
```
8.91.3.4 loadNew() ModelElement * Plugin::loadNew (
    Model * model,
    std::map< std::string, std::string > * fields )
```

Definition at line 48 of file [Plugin.cpp](#).

```
00048     if (this->_pluginInfo->isComponent()) {
00049         return _loadNewComponent(model, fields);
00050     } else {
00051         return _loadNewElement(model, fields);
00052     }
00053 }
```

References [PluginInformation::isComponent\(\)](#).

Here is the call graph for this function:



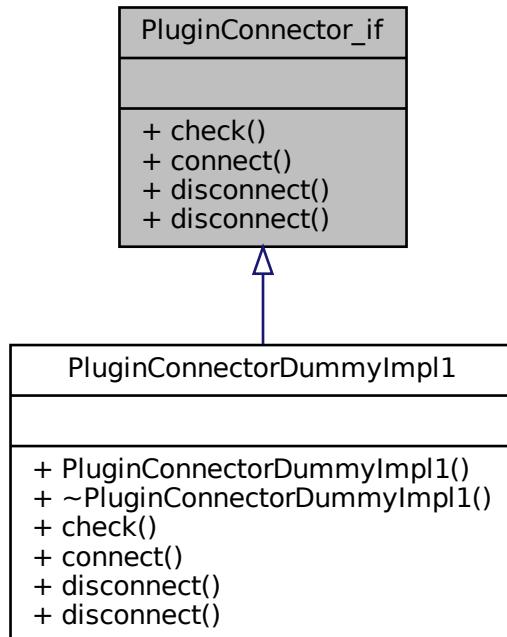
The documentation for this class was generated from the following files:

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

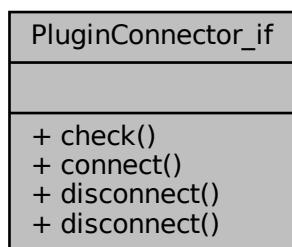
8.92 **PluginConnector_if** Class Reference

```
#include <PluginConnector_if.h>
```

Inheritance diagram for PluginConnector_if:



Collaboration diagram for PluginConnector_if:



Public Member Functions

- virtual `Plugin * check (const std::string dynamicLibraryFilename)=0`
- virtual `Plugin * connect (const std::string dynamicLibraryFilename)=0`
- virtual `bool disconnect (const std::string dynamicLibraryFilename)=0`
- virtual `bool disconnect (Plugin *plugin)=0`

8.92.1 Detailed Description

Definition at line 22 of file [PluginConnector_if.h](#).

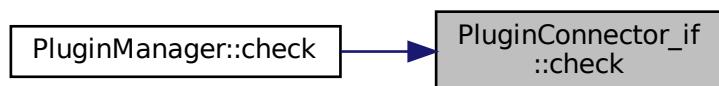
8.92.2 Member Function Documentation

8.92.2.1 check() virtual [Plugin](#)* PluginConnector_if::check (const std::string *dynamicLibraryFilename*) [pure virtual]

Implemented in [PluginConnectorDummyImpl1](#).

Referenced by [PluginManager::check\(\)](#).

Here is the caller graph for this function:

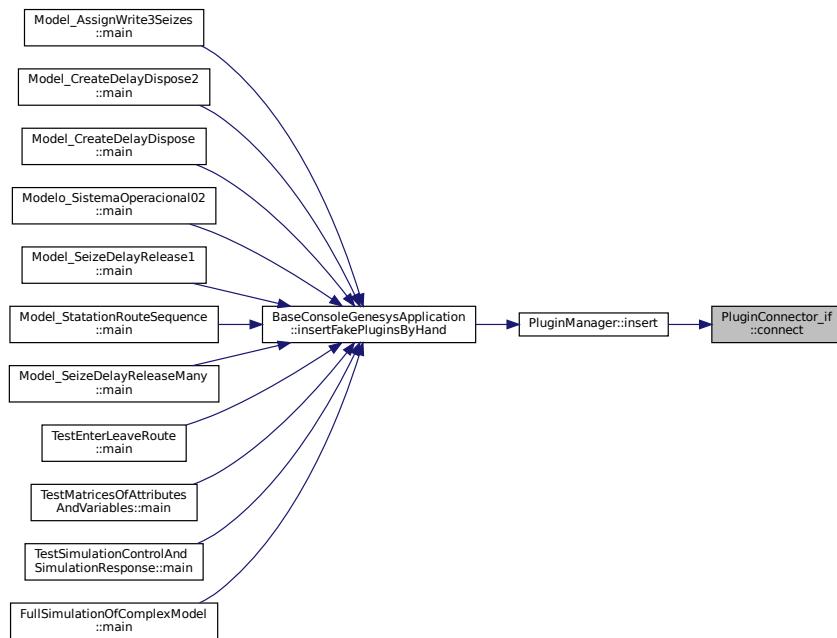


8.92.2.2 connect() virtual [Plugin](#)* PluginConnector_if::connect (const std::string *dynamicLibraryFilename*) [pure virtual]

Implemented in [PluginConnectorDummyImpl1](#).

Referenced by [PluginManager::insert\(\)](#).

Here is the caller graph for this function:

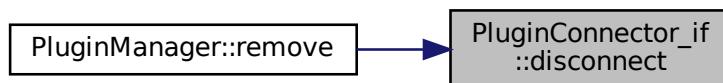


8.92.2.3 disconnect() [1/2] `virtual bool PluginConnector_if::disconnect (const std::string dynamicLibraryFilename) [pure virtual]`

Implemented in [PluginConnectorDummyImpl1](#).

Referenced by [PluginManager::remove\(\)](#).

Here is the caller graph for this function:



8.92.2.4 disconnect() [2/2] `virtual bool PluginConnector_if::disconnect (Plugin * plugin) [pure virtual]`

Implemented in [PluginConnectorDummyImpl1](#).

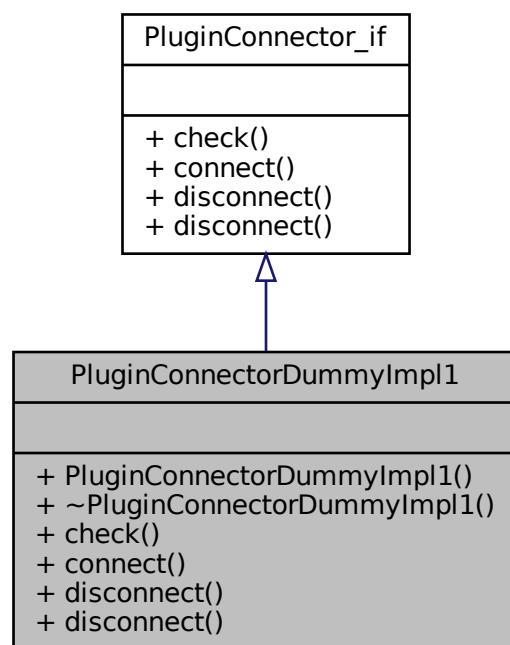
The documentation for this class was generated from the following file:

- [PluginConnector_if.h](#)

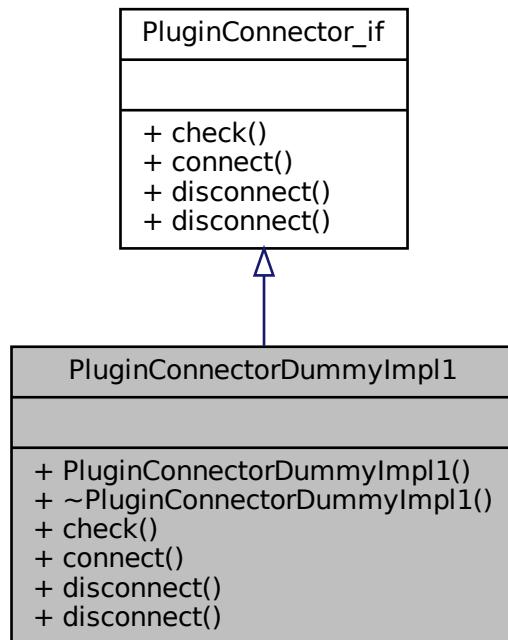
8.93 PluginConnectorDummyImpl1 Class Reference

```
#include <PluginConnectorDummyImpl1.h>
```

Inheritance diagram for PluginConnectorDummyImpl1:



Collaboration diagram for PluginConnectorDummyImpl1:



Public Member Functions

- `PluginConnectorDummyImpl1 ()`
- `virtual ~PluginConnectorDummyImpl1 ()=default`
- `virtual Plugin * check (const std::string dynamicLibraryFilename)`
- `virtual Plugin * connect (const std::string dynamicLibraryFilename)`
- `virtual bool disconnect (const std::string dynamicLibraryFilename)`
- `virtual bool disconnect (Plugin *plugin)`

8.93.1 Detailed Description

Definition at line 20 of file [PluginConnectorDummyImpl1.h](#).

8.93.2 Constructor & Destructor Documentation

8.93.2.1 PluginConnectorDummyImpl1() `PluginConnectorDummyImpl1::PluginConnectorDummyImpl1 ()`

Definition at line 72 of file [PluginConnectorDummyImpl1.cpp](#).

```

00072
00073     {

```

8.93.2.2 ~PluginConnectorDummyImpl1() virtual PluginConnectorDummyImpl1::~PluginConnector<→
DummyImpl1 () [virtual], [default]

8.93.3 Member Function Documentation

8.93.3.1 check() `Plugin * PluginConnectorDummyImpl1::check (const std::string dynamicLibraryFilename) [virtual]`

@

Todo :To implement

Implements [PluginConnector_if](#).

Definition at line 75 of file [PluginConnectorDummyImpl1.cpp](#).

```
00075     return nullptr;
00076 }
00077 }
```

8.93.3.2 connect() `Plugin * PluginConnectorDummyImpl1::connect (const std::string dynamicLibraryFilename) [virtual]`

Implements [PluginConnector_if](#).

Definition at line 79 of file [PluginConnectorDummyImpl1.cpp](#).

```
00079
00080     std::string fn = getFileName(dynamicLibraryFilename);
00081     StaticGetPluginInformation GetInfo = nullptr;
00082     Plugin* pluginResult = nullptr;
00083 // model elements
00084     if (fn == "attribute.so")
00085         GetInfo = &Attribute::GetPluginInformation;
00086     else if (fn == "assign.so")
00087         GetInfo = &Assign::GetPluginInformation;
00088     else if (fn == "counter.so")
00089         GetInfo = &Counter::GetPluginInformation;
00090     else if (fn == "entitygroup.so")
00091         GetInfo = &EntityGroup::GetPluginInformation;
00092     else if (fn == "entitytype.so")
00093         GetInfo = &EntityType::GetPluginInformation;
00094     else if (fn == "formula.so")
00095         GetInfo = &Formula::GetPluginInformation;
00096     // else if (fn == "ode.so")
00097     //     GetInfo = &OLD_ODElement::GetPluginInformation;
00098     else if (fn == "queue.so")
00099         GetInfo = &Queue::GetPluginInformation;
00100    else if (fn == "resource.so")
00101        GetInfo = &Resource::GetPluginInformation;
00102    else if (fn == "set.so")
00103        GetInfo = &Set::GetPluginInformation;
00104    else if (fn == "statisticscollector.so")
00105        GetInfo = &StatisticsCollector::GetPluginInformation;
00106    else if (fn == "station.so")
00107        GetInfo = &Station::GetPluginInformation;
00108    else if (fn == "variable.so")
00109        GetInfo = &Variable::GetPluginInformation;
00110 // model components
00111    else if (fn == "create.so")
00112        GetInfo = &Create::GetPluginInformation;
00113    else if (fn == "write.so")
00114        GetInfo = &Write::GetPluginInformation;
00115    else if (fn == "decide.so")
00116        GetInfo = &Decide::GetPluginInformation;
```

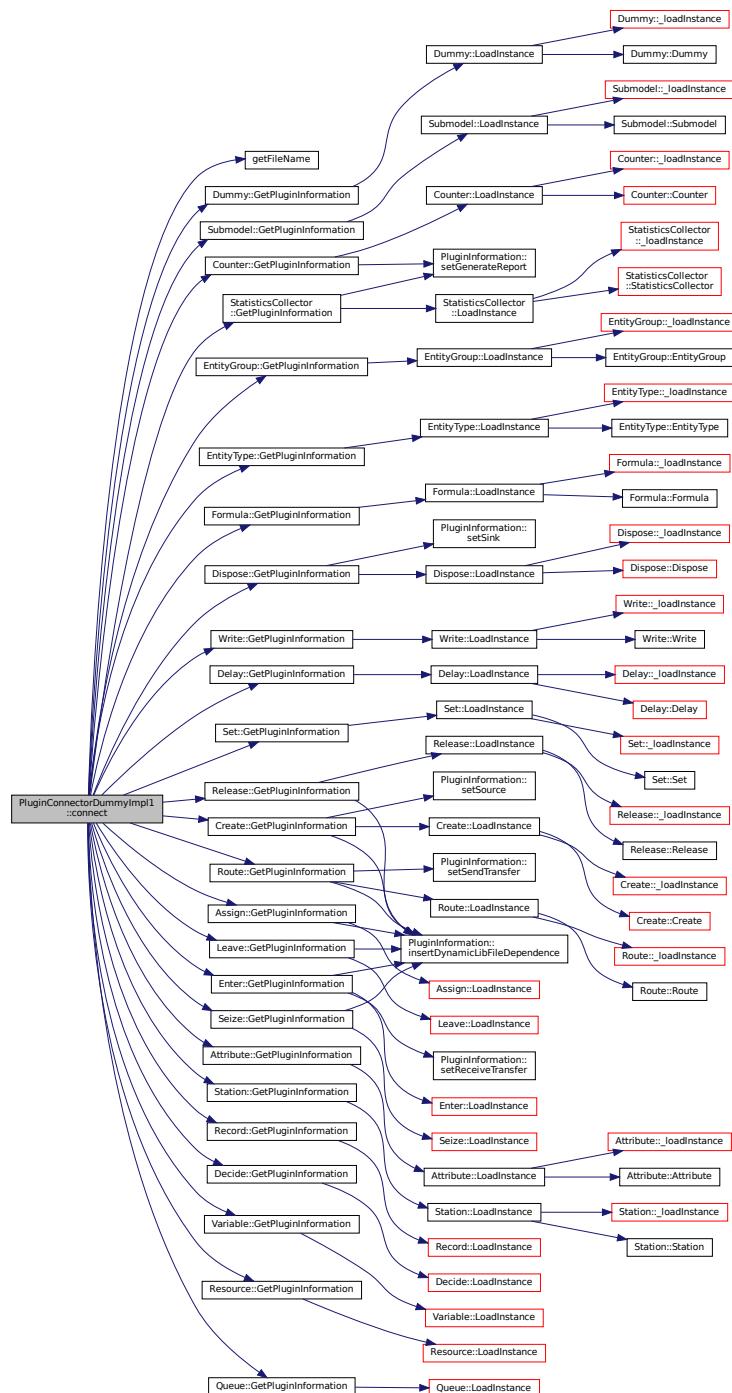
```

00117     else if (fn == "delay.so")
00118         GetInfo = &Delay::GetPluginInformation;
00119     else if (fn == "dispose.so")
00120         GetInfo = &Dispose::GetPluginInformation;
00121     else if (fn == "dummy.so")
00122         GetInfo = &Dummy::GetPluginInformation;
00123     else if (fn == "record.so")
00124         GetInfo = &Record::GetPluginInformation;
00125     else if (fn == "release.so")
00126         GetInfo = &Release::GetPluginInformation;
00127     else if (fn == "seize.so")
00128         GetInfo = &Seize::GetPluginInformation;
00129     else if (fn == "route.so")
00130         GetInfo = &Route::GetPluginInformation;
00131     else if (fn == "enter.so")
00132         GetInfo = &Enter::GetPluginInformation;
00133     else if (fn == "leave.so")
00134         GetInfo = &Leave::GetPluginInformation;
00135     // else if (fn == "lsode.so")
00136     //     GetInfo = &LSODE::GetPluginInformation;
00137     // else if (fn == "markovchain.so")
00138     //     GetInfo = &MarkovChain::GetPluginInformation;
00139     // else if (fn == "hold.so")
00140     //     GetInfo = &Hold::GetPluginInformation;
00141     // else if (fn == "schedule.so")
00142     //     GetInfo = &Schedule::GetPluginInformation;
00143     // else if (fn == "dropoff.so")
00144     //     GetInfo = &DropOff::GetPluginInformation;
00145     // else if (fn == "batch.so")
00146     //     GetInfo = &Batch::GetPluginInformation;
00147     // else if (fn == "separate.so")
00148     //     GetInfo = &Separate::GetPluginInformation;
00149 else if (fn == "submodel.so")
00150     GetInfo = &Submodel::GetPluginInformation;
00151 // else if (fn == "match.so")
00152 //     GetInfo = &Match::GetPluginInformation;
00153 // else if (fn == "pickup.so")
00154 //     GetInfo = &PickUp::GetPluginInformation;
00155 //else if (fn == "read.so")
00156 //     GetInfo = &Read::GetPluginInformation;
00157 // else if (fn == "remove.so")
00158 //     GetInfo = &Remove::GetPluginInformation;
00159 // else if (fn == "search.so")
00160 //     GetInfo = &Search::GetPluginInformation;
00161 // else if (fn == "signal.so")
00162 //     GetInfo = &Signal::GetPluginInformation;
00163 // else if (fn == "store.so")
00164 //     GetInfo = &Store::GetPluginInformation;
00165 // else if (fn == "unstore.so")
00166 //     GetInfo = &Unstore::GetPluginInformation;
00167 //else if (fn == "expression.so")
00168 //     GetInfo = &Expression::GetPluginInformation;
00169 // else if (fn == "failure.so")
00170 //     GetInfo = &Failure::GetPluginInformation;
00171 // else if (fn == "file.so")
00172 //     GetInfo = &File::GetPluginInformation;
00173 // else if (fn == "storage.so")
00174 //     GetInfo = &Storage::GetPluginInformation;
00175 // else if (fn == "pickstation.so")
00176 //     GetInfo = &PickStation::GetPluginInformation;
00177 // else if (fn == "sequence.so")
00178 //     GetInfo = &Sequence::GetPluginInformation;
00179 // else if (fn == "access.so")
00180 //     GetInfo = &Access::GetPluginInformation;
00181 // else if (fn == "exit.so")
00182 //     GetInfo = &Exit::GetPluginInformation;
00183 // else if (fn == "start.so")
00184 //     GetInfo = &Start::GetPluginInformation;
00185 // else if (fn == "stop.so")
00186 //     GetInfo = &Stop::GetPluginInformation;
00187 //else if (fn == "conveyour.so")
00188 //     GetInfo = &Conveyour::GetPluginInformation;
00189 //else if (fn == "segment.so")
00190 //     GetInfo = &Segment::GetPluginInformation;
00191
00192 //else if (fn=="")
00193
00194     if (GetInfo != nullptr) {
00195         pluginResult = new Plugin(GetInfo);
00196     }
00197     return pluginResult;
00198 }
```

References [getFileName\(\)](#), [Dummy::GetPluginInformation\(\)](#), [Submodel::GetPluginInformation\(\)](#), [Counter::GetPluginInformation\(\)](#), [EntityGroup::GetPluginInformation\(\)](#), [StatisticsCollector::GetPluginInformation\(\)](#), [EntityType::GetPluginInformation\(\)](#), [Formula::GetPluginInformation\(\)](#), [Dispose::GetPluginInformation\(\)](#), [Write::GetPluginInformation\(\)](#), [Delay::GetPluginInformation\(\)](#),

`Set::GetPluginInformation()`, `Route::GetPluginInformation()`, `Attribute::GetPluginInformation()`, `Release::GetPluginInformation()`, `Station::GetPluginInformation()`, `Create::GetPluginInformation()`, `Record::GetPluginInformation()`, `Decide::GetPluginInformation()`, `Variable::GetPluginInformation()`, `Assign::GetPluginInformation()`, `Leave::GetPluginInformation()`, `Resource::GetPluginInformation()`, `Queue::GetPluginInformation()`, `Enter::GetPluginInformation()`, and `Seize::GetPluginInformation()`.

Here is the call graph for this function:



8.93.3.3 disconnect() [1/2] `bool PluginConnectorDummyImpl1::disconnect (const std::string dynamicLibraryFilename) [virtual]`

Implements [PluginConnector_if](#).

Definition at line 200 of file [PluginConnectorDummyImpl1.cpp](#).

```
00200
00201     return true;
00202 }
```

8.93.3.4 disconnect() [2/2] `bool PluginConnectorDummyImpl1::disconnect (Plugin * plugin) [virtual]`

Implements [PluginConnector_if](#).

Definition at line 204 of file [PluginConnectorDummyImpl1.cpp](#).

```
00204
00205     return true;
00206 }
```

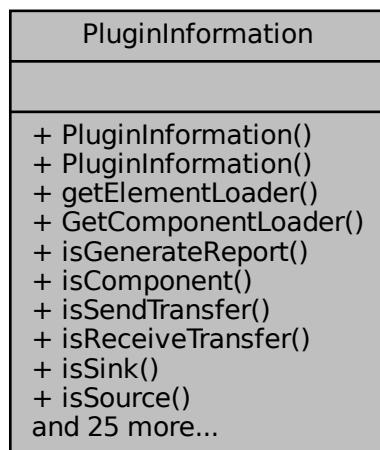
The documentation for this class was generated from the following files:

- [PluginConnectorDummyImpl1.h](#)
- [PluginConnectorDummyImpl1.cpp](#)

8.94 PluginInformation Class Reference

```
#include <PluginInformation.h>
```

Collaboration diagram for PluginInformation:



Public Member Functions

- `PluginInformation (std::string pluginTypename, StaticLoaderComponentInstance componentloader)`
- `PluginInformation (std::string pluginTypename, StaticLoaderElementInstance elementloader)`
- `StaticLoaderElementInstance getElementLoader () const`
- `StaticLoaderComponentInstance GetComponentLoader () const`
- `bool isGenerateReport () const`
- `bool isComponent () const`
- `bool isSendTransfer () const`
- `bool isReceiveTransfer () const`
- `bool isSink () const`
- `bool isSource () const`
- `std::string getObservation () const`
- `std::string getVersion () const`
- `std::string getDate () const`
- `std::string getAuthor () const`
- `std::string getPluginTypename () const`
- `void insertDynamicLibFileDependence (std::string filename)`
- `void setDynamicLibFilenameDependencies (std::list< std::string > *dynamicLibFilenameDependencies)`
- `std::list< std::string > * getDynamicLibFilenameDependencies () const`
- `void setGenerateReport (bool generateReport)`
- `void setSendTransfer (bool sendTransfer)`
- `void setReceiveTransfer (bool receiveTransfer)`
- `void setSink (bool Sink)`
- `void setSource (bool Source)`
- `void setObservation (std::string observation)`
- `void setVersion (std::string version)`
- `void setDate (std::string date)`
- `void setAuthor (std::string author)`
- `void setMaximumOutputs (unsigned short _maximumOutputs)`
- `unsigned short getMaximumOutputs () const`
- `void setMinimumOutputs (unsigned short _minimumOutputs)`
- `unsigned short getMinimumOutputs () const`
- `void setMaximumInputs (unsigned short _maximumInputs)`
- `unsigned short getMaximumInputs () const`
- `void setMinimumInputs (unsigned short _minimumInputs)`
- `unsigned short getMinimumInputs () const`

8.94.1 Detailed Description

Definition at line 33 of file [PluginInformation.h](#).

8.94.2 Constructor & Destructor Documentation

8.94.2.1 PluginInformation() [1/2] `PluginInformation::PluginInformation (std::string pluginTypename, StaticLoaderComponentInstance componentloader)`

Todo :

Definition at line 18 of file [PluginInformation.cpp](#).

```
00018     {
00019     this->_componentloader = componentloader;
00020     this->_elementloader = nullptr;
00021     this->_isComponent = true;
00022     this->_pluginTypename = pluginTypename;
00023 }
```

8.94.2.2 PluginInformation() [2/2] `PluginInformation::PluginInformation (std::string pluginTypename, StaticLoaderElementInstance elementloader)`

Definition at line 25 of file [PluginInformation.cpp](#).

```
00025     {
00026     this->_componentloader = nullptr;
00027     this->_elementloader = elementloader;
00028     this->_isComponent = false;
00029     this->_pluginTypename = pluginTypename;
00030 }
```

8.94.3 Member Function Documentation

8.94.3.1 getAuthor() `std::string PluginInformation::getAuthor () const`

Definition at line 76 of file [PluginInformation.cpp](#).

```
00076     {
00077     return _author;
00078 }
```

8.94.3.2 GetComponentLoader() `StaticLoaderComponentInstance PluginInformation::GetComponentLoader () const`

Definition at line 36 of file [PluginInformation.cpp](#).

```
00036     {
00037     return _componentloader;
00038 }
```

8.94.3.3 getDate() `std::string PluginInformation::getDate () const`

Definition at line 72 of file [PluginInformation.cpp](#).

```
00072     {
00073     return _date;
00074 }
```

8.94.3.4 getDynamicLibFilenameDependencies() `std::list< std::string > * PluginInformation::getDynamicLibFilenameDependencies () const`

Definition at line 92 of file [PluginInformation.cpp](#).

```
00092 {  
00093     return _dynamicLibFilenameDependencies;  
00094 }
```

8.94.3.5 getElementLoader() `staticLoaderElementInstance PluginInformation::getElementLoader () const`

Definition at line 32 of file [PluginInformation.cpp](#).

```
00032 {  
00033     return _elementloader;  
00034 }
```

8.94.3.6 getMaximumInputs() `unsigned short PluginInformation::getMaximumInputs () const`

Definition at line 152 of file [PluginInformation.cpp](#).

```
00152 {  
00153     return _maximumInputs;  
00154 }
```

8.94.3.7 getMaximumOutputs() `unsigned short PluginInformation::getMaximumOutputs () const`

Definition at line 136 of file [PluginInformation.cpp](#).

```
00136 {  
00137     return _maximumOutputs;  
00138 }
```

8.94.3.8 getMinimumInputs() `unsigned short PluginInformation::getMinimumInputs () const`

Definition at line 160 of file [PluginInformation.cpp](#).

```
00160 {  
00161     return _minimumInputs;  
00162 }
```

8.94.3.9 getMinimumOutputs() `unsigned short PluginInformation::getMinimumOutputs () const`

Definition at line 144 of file [PluginInformation.cpp](#).

```
00144 {  
00145     return _minimumOutputs;  
00146 }
```

8.94.3.10 getObservation() std::string PluginInformation::getObservation () constDefinition at line 64 of file [PluginInformation.cpp](#).

```
00064
00065     return _observation;
00066 }
```

8.94.3.11 getPluginTypename() std::string PluginInformation::getPluginTypename () constDefinition at line 80 of file [PluginInformation.cpp](#).

```
00080
00081     return _pluginTypename;
00082 }
```

8.94.3.12 getVersion() std::string PluginInformation::getVersion () constDefinition at line 68 of file [PluginInformation.cpp](#).

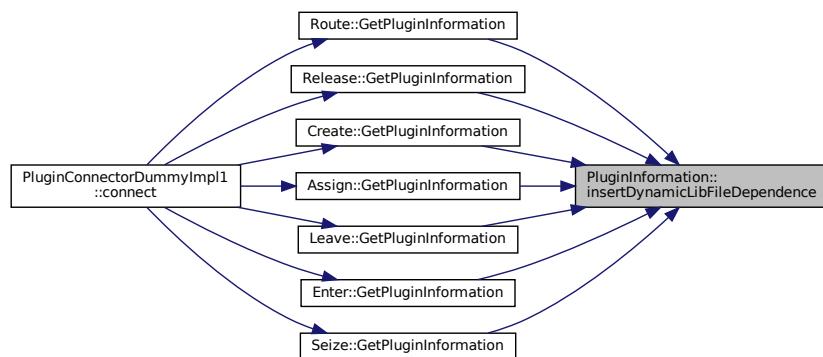
```
00068
00069     return _version;
00070 }
```

8.94.3.13 insertDynamicLibFileDependence() void PluginInformation::insertDynamicLibFileDependence (std::string filename)Definition at line 84 of file [PluginInformation.cpp](#).

```
00084
00085     _dynamicLibFilenameDependencies->insert (_dynamicLibFilenameDependencies->end(), filename);
00086 }
```

Referenced by [Route::GetPluginInformation\(\)](#), [Release::GetPluginInformation\(\)](#), [Create::GetPluginInformation\(\)](#), [Assign::GetPluginInformation\(\)](#), [Leave::GetPluginInformation\(\)](#), [Enter::GetPluginInformation\(\)](#), and [Seize::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



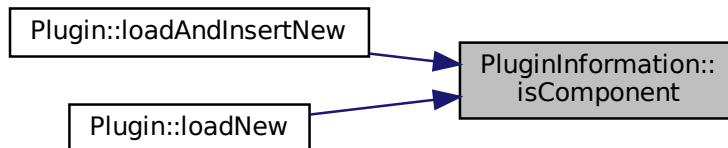
8.94.3.14 **isComponent()** `bool PluginInformation::isComponent () const`

Definition at line 44 of file [PluginInformation.cpp](#).

```
00044     {  
00045         return _isComponent;  
00046     }
```

Referenced by [Plugin::loadAndInsertNew\(\)](#), and [Plugin::loadNew\(\)](#).

Here is the caller graph for this function:



8.94.3.15 **isGenerateReport()** `bool PluginInformation::isGenerateReport () const`

Definition at line 40 of file [PluginInformation.cpp](#).

```
00040     {  
00041         return _generateReport;  
00042     }
```

8.94.3.16 **isReceiveTransfer()** `bool PluginInformation::isReceiveTransfer () const`

Definition at line 52 of file [PluginInformation.cpp](#).

```
00052     {  
00053         return _receiveTransfer;  
00054     }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#).

Here is the caller graph for this function:



8.94.3.17 `isSendTransfer()` `bool PluginInformation::isSendTransfer () const`

Definition at line 48 of file [PluginInformation.cpp](#).

```
00048     {  
00049         return _sendTransfer;  
00050     }
```

8.94.3.18 `isSink()` `bool PluginInformation::isSink () const`

Definition at line 56 of file [PluginInformation.cpp](#).

```
00056     {  
00057         return _isSink;  
00058     }
```

8.94.3.19 `isSource()` `bool PluginInformation::isSource () const`

Definition at line 60 of file [PluginInformation.cpp](#).

```
00060     {  
00061         return _isSource;  
00062     }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#).

Here is the caller graph for this function:

**8.94.3.20 `setAuthor()`** `void PluginInformation::setAuthor (std::string author)`

Definition at line 128 of file [PluginInformation.cpp](#).

```
00128     {  
00129         this->_author = author;  
00130     }
```

8.94.3.21 `setDate()` `void PluginInformation::setDate (std::string date)`

Definition at line 124 of file [PluginInformation.cpp](#).

```
00124     {  
00125         this->_date = date;  
00126     }
```

8.94.3.22 setDynamicLibFilenameDependencies() void PluginInformation::setDynamicLibFilenameDependencies (std::list< std::string > * dynamicLibFilenameDependencies)

Definition at line 88 of file [PluginInformation.cpp](#).

```
00088     {
00089         this->_dynamicLibFilenameDependencies = dynamicLibFilenameDependencies;
00090     }
```

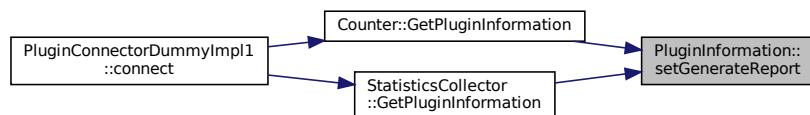
8.94.3.23 setGenerateReport() void PluginInformation::setGenerateReport (bool generateReport)

Definition at line 96 of file [PluginInformation.cpp](#).

```
00096     {
00097         this->_generateReport = generateReport;
00098     }
```

Referenced by [Counter::GetPluginInformation\(\)](#), and [StatisticsCollector::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



8.94.3.24 setMaximumInputs() void PluginInformation::setMaximumInputs (unsigned short _maximumInputs)

Definition at line 148 of file [PluginInformation.cpp](#).

```
00148     {
00149         this->_maximumInputs = _maximumInputs;
00150     }
```

8.94.3.25 setMaximumOutputs() void PluginInformation::setMaximumOutputs (unsigned short _maximumOutputs)

Definition at line 132 of file [PluginInformation.cpp](#).

```
00132     {
00133         this->_maximumOutputs = _maximumOutputs;
00134     }
```

```
8.94.3.26 setMinimumInputs() void PluginInformation::setMinimumInputs (  
    unsigned short _minimumInputs )
```

Definition at line 156 of file [PluginInformation.cpp](#).

```
00156  
00157     this->_minimumInputs = _minimumInputs;  
00158 }
```

```
8.94.3.27 setMinimumOutputs() void PluginInformation::setMinimumOutputs (   
    unsigned short _minimumOutputs )
```

Definition at line 140 of file [PluginInformation.cpp](#).

```
00140  
00141     this->_minimumOutputs = _minimumOutputs;  
00142 }
```

```
8.94.3.28 setObservation() void PluginInformation::setObservation (   
    std::string observation )
```

Definition at line 116 of file [PluginInformation.cpp](#).

```
00116  
00117     this->_observation = observation;  
00118 }
```

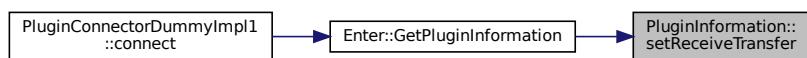
```
8.94.3.29 setReceiveTransfer() void PluginInformation::setReceiveTransfer (   
    bool receiveTransfer )
```

Definition at line 104 of file [PluginInformation.cpp](#).

```
00104  
00105     this->_receiveTransfer = receiveTransfer;  
00106 }
```

Referenced by [Enter::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



8.94.3.30 setSendTransfer() void PluginInformation::setSendTransfer (bool sendTransfer)

Definition at line 100 of file [PluginInformation.cpp](#).

```
00100
00101     this->_sendTransfer = sendTransfer;
00102 }
```

Referenced by [Route::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



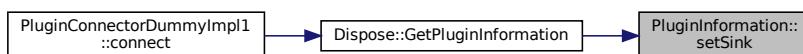
8.94.3.31 setSink() void PluginInformation::setSink (bool Sink)

Definition at line 108 of file [PluginInformation.cpp](#).

```
00108
00109     _isSink = Sink;
00110 }
```

Referenced by [Dispose::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



8.94.3.32 setSource() void PluginInformation::setSource (bool Source)

Definition at line 112 of file [PluginInformation.cpp](#).

```
00112
00113     _isSource = Source;
00114 }
```

Referenced by [Create::GetPluginInformation\(\)](#).

Here is the caller graph for this function:



```
8.94.3.33 setVersion() void PluginInformation::setVersion (
    std::string version )
```

Definition at line 120 of file [PluginInformation.cpp](#).

```
00120
00121     this->_version = version;
00122 }
```

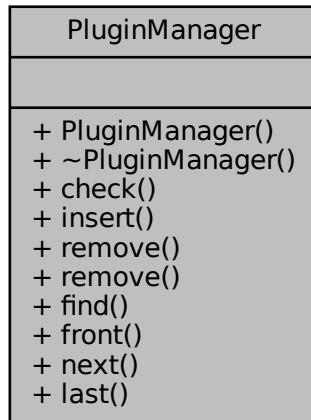
The documentation for this class was generated from the following files:

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

8.95 PluginManager Class Reference

```
#include <PluginManager.h>
```

Collaboration diagram for PluginManager:



Public Member Functions

- `PluginManager (Simulator *simulator)`
- virtual `~PluginManager ()`=default
- `bool check (const std::string dynamicLibraryFilename)`
- `Plugin * insert (const std::string dynamicLibraryFilename)`
- `bool remove (const std::string dynamicLibraryFilename)`
- `bool remove (Plugin *plugin)`
- `Plugin * find (std::string pluginTypeName)`
- `Plugin * front ()`
- `Plugin * next ()`
- `Plugin * last ()`

8.95.1 Detailed Description

Definition at line 24 of file [PluginManager.h](#).

8.95.2 Constructor & Destructor Documentation

8.95.2.1 [PluginManager\(\)](#) PluginManager::PluginManager (

`Simulator * simulator)`

Definition at line 20 of file [PluginManager.cpp](#).

```
00020 {  
00021     _simulator = simulator;  
00022     this->_pluginConnector = new Traits<PluginConnector_if>::Implementation();  
00023 }
```

8.95.2.2 [~PluginManager\(\)](#) virtual PluginManager::~PluginManager () [virtual], [default]

8.95.3 Member Function Documentation

8.95.3.1 [check\(\)](#) bool PluginManager::check (

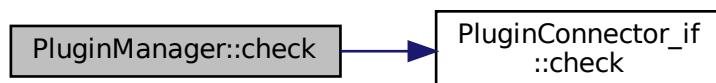
`const std::string dynamicLibraryFilename)`

Definition at line 80 of file [PluginManager.cpp](#).

```
00080 {  
00081     Plugin* plugin;  
00082     try {  
00083         plugin = _pluginConnector->check(dynamicLibraryFilename);  
00084     } catch (...) {  
00085         return false;  
00086     }  
00087     return (plugin != nullptr);  
00088 }
```

References [PluginConnector_if::check\(\)](#).

Here is the call graph for this function:



8.95.3.2 find() `Plugin * PluginManager::find (std::string pluginTypeName)`

Definition at line 127 of file [PluginManager.cpp](#).

```
00127
00128     for (std::list<Plugin*>::iterator it = this->_plugins->list\(\)->begin(); it != _plugins->list\(\)->end(); it++) {
00129         if ((*it)->getPluginInfo\(\)->getPluginTypename\(\) == pluginTypeName) {
00130             return (*it);
00131         }
00132     }
00134     return nullptr;
00135 }
```

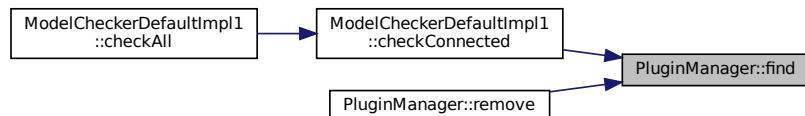
References [List< T >::list\(\)](#).

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), and [remove\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.95.3.3 front() `Plugin * PluginManager::front ()`

Definition at line 137 of file [PluginManager.cpp](#).

```
00137
00138
00139     return this->\_plugins->front\(\);
00140 }
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



8.95.3.4 insert() `Plugin * PluginManager::insert (const std::string dynamicLibraryFilename)`

Definition at line 90 of file `PluginManager.cpp`.

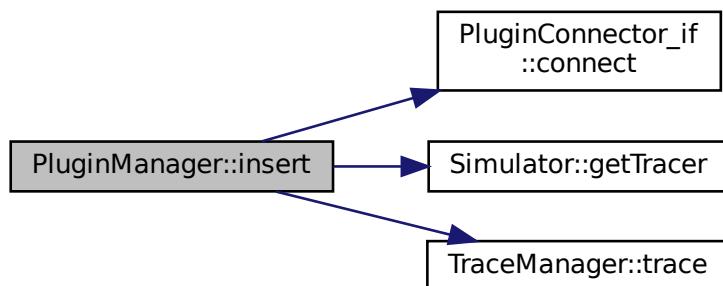
```

00090
00091     Plugin* plugin;
00092     try {
00093         plugin = _pluginConnector->connect(dynamicLibraryFilename);
00094         if (plugin != nullptr)
00095             _insert(plugin);
00096         else {
00097             _simulator->getTracer()->trace(Util::TraceLevel::errorRecover, "Plugin from file \\" + dynamicLibraryFilename + "\\" could not be loaded.");
00098         }
00099     } catch (...) {
00100     }
00101     return nullptr;
00102 }
00103 return plugin;
00104 }
```

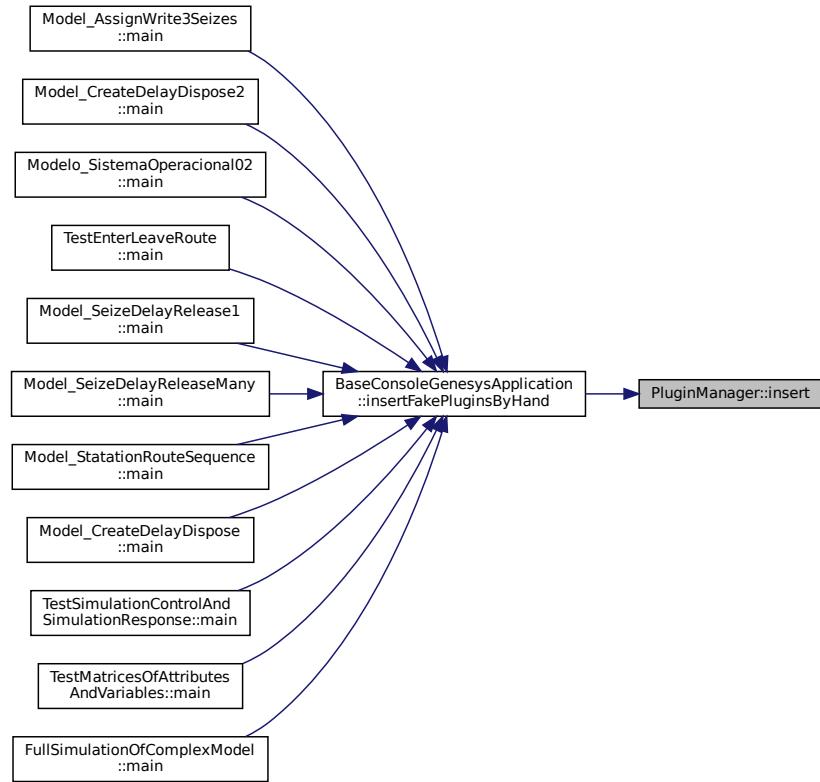
References `PluginConnector_if::connect()`, `Util::errorRecover`, `Simulator::getTracer()`, and `TraceManager::trace()`.

Referenced by `BaseConsoleGenesysApplication::insertFakePluginsByHand()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.95.3.5 last() `Plugin * PluginManager::last ()`

Definition at line 147 of file [PluginManager.cpp](#).

```

00147     {
00148     return this->_plugins->last ();
00149 }
```

References [List< T >::last\(\)](#).

Here is the call graph for this function:



8.95.3.6 `next()` `Plugin * PluginManager::next ()`

Definition at line 142 of file [PluginManager.cpp](#).

```
00142     {
00143     00144         return _plugins->next ();
00145 }
```

References [List< T >::next\(\)](#).

Here is the call graph for this function:



8.95.3.7 `remove()` [1/2] `bool PluginManager::remove (const std::string dynamicLibraryFilename)`

Definition at line 106 of file [PluginManager.cpp](#).

```
00106
00107
00108     Plugin* pi = this->find(dynamicLibraryFilename);
00109     return remove(pi);
00110 }
```

References [find\(\)](#).

Here is the call graph for this function:



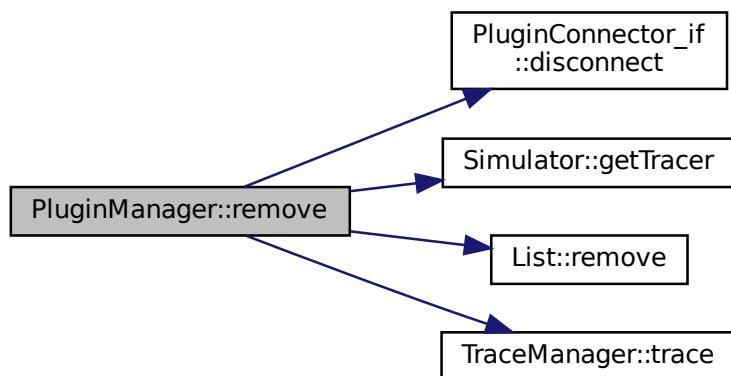
8.95.3.8 remove() [2/2] `bool PluginManager::remove (`
`Plugin * plugin)`

Definition at line 112 of file `PluginManager.cpp`.

```
00112     {
00113         if (plugin != nullptr) {
00114             _plugins->remove(plugin);
00115             try {
00116                 _pluginConnector->disconnect(plugin);
00117             } catch (...) {
00118                 return false;
00119             }
00120             _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin successfully
removed");
00121             return true;
00122         }
00123         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin could not be removed");
00124         return false;
00125     }
```

References `PluginConnector_if::disconnect()`, `Simulator::getTracer()`, `List< T >::remove()`, `Util::simulatorResult`, and `TraceManager::trace()`.

Here is the call graph for this function:



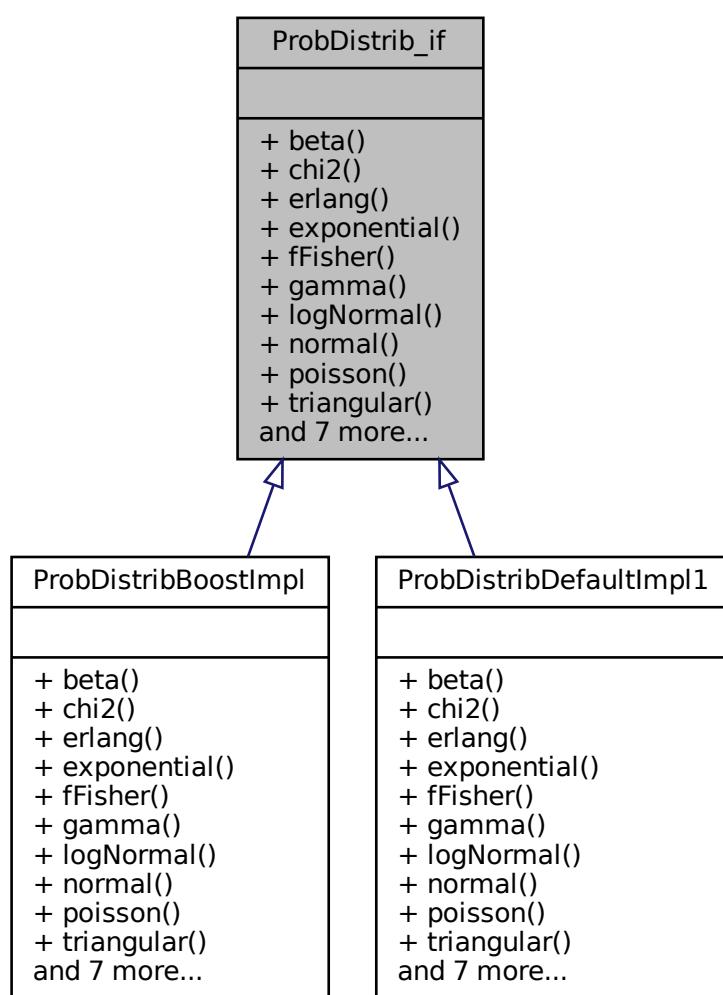
The documentation for this class was generated from the following files:

- `PluginManager.h`
- `PluginManager.cpp`

8.96 ProbDistrib_if Class Reference

```
#include <Prob_Distrib_if.h>
```

Inheritance diagram for ProbDistrib_if:



Collaboration diagram for ProbDistrib_if:

ProbDistrib_if
+ beta() + chi2() + erlang() + exponential() + fFisher() + gamma() + logNormal() + normal() + poisson() + triangular() and 7 more...

Public Member Functions

- virtual double `beta` (double x, double alpha, double beta)=0
- virtual double `chi2` (double x, double m)=0
- virtual double `erlang` (double x, double shape, double scale)=0
- virtual double `exponential` (double x, double mean)=0
- virtual double `fFisher` (double x, double k, double m)=0
- virtual double `gamma` (double x, double shape, double scale)=0
- virtual double `logNormal` (double x, double mean, double stddev)=0
- virtual double `normal` (double x, double mean, double stddev)=0
- virtual double `poisson` (double x, double mean)=0
- virtual double `triangular` (double x, double min, double mode, double max)=0
- virtual double `tStudent` (double x, double mean, double stddev, unsigned int degreeFreedom)=0
- virtual double `uniform` (double x, double min, double max)=0
- virtual double `weibull` (double x, double shape, double scale)=0
- virtual double `inverseChi2` (double cumulativeProbability, double m)=0
- virtual double `inverseFFisher` (double cumulativeProbability, double k, double m)=0
- virtual double `inverseNormal` (double cumulativeProbability, double mean, double stddev)=0
- virtual double `inverseTStudent` (double cumulativeProbability, double mean, double stddev, double degreeFreedom)=0

8.96.1 Detailed Description

Definition at line 17 of file `Prob_Distrib_if.h`.

8.96.2 Member Function Documentation

```
8.96.2.1 beta() virtual double ProbDistrib_if::beta (
    double x,
    double alpha,
    double beta ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.2 chi2() virtual double ProbDistrib_if::chi2 (
    double x,
    double m ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.3 erlang() virtual double ProbDistrib_if::erlang (
    double x,
    double shape,
    double scale ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.4 exponential() virtual double ProbDistrib_if::exponential (
    double x,
    double mean ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.5 fFisher() virtual double ProbDistrib_if::fFisher (
    double x,
    double k,
    double m ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.6 gamma() virtual double ProbDistrib_if::gamma (
    double x,
    double shape,
    double scale ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.7 inverseChi2() virtual double ProbDistrib_if::inverseChi2 (
    double cumulativeProbability,
    double m) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.8 inverseFFisher() virtual double ProbDistrib_if::inverseFFisher (
    double cumulativeProbability,
    double k,
    double m) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.9 inverseNormal() virtual double ProbDistrib_if::inverseNormal (
    double cumulativeProbability,
    double mean,
    double stddev) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.10 inverseTStudent() virtual double ProbDistrib_if::inverseTStudent (
    double cumulativeProbability,
    double mean,
    double stddev,
    double degreeFreedom) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.11 logNormal() virtual double ProbDistrib_if::logNormal (
    double x,
    double mean,
    double stddev) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.12 normal() virtual double ProbDistrib_if::normal (
    double x,
    double mean,
    double stddev) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.13 poisson() virtual double ProbDistrib_if::poisson (
    double x,
    double mean ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.14 triangular() virtual double ProbDistrib_if::triangular (
    double x,
    double min,
    double mode,
    double max ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.15 tStudent() virtual double ProbDistrib_if::tStudent (
    double x,
    double mean,
    double stddev,
    unsigned int degreeFreedom ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.16 uniform() virtual double ProbDistrib_if::uniform (
    double x,
    double min,
    double max ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

```
8.96.2.17 weibull() virtual double ProbDistrib_if::weibull (
    double x,
    double shape,
    double scale ) [pure virtual]
```

Implemented in [ProbDistribBoostImpl](#), and [ProbDistribDefaultImpl1](#).

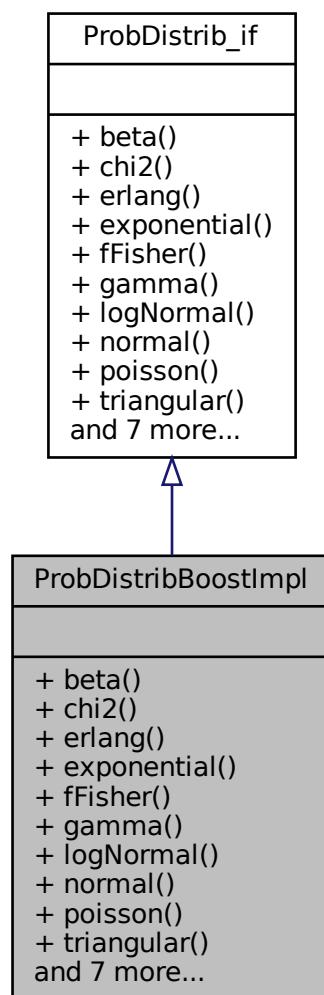
The documentation for this class was generated from the following file:

- [Prob_Distrib_if.h](#)

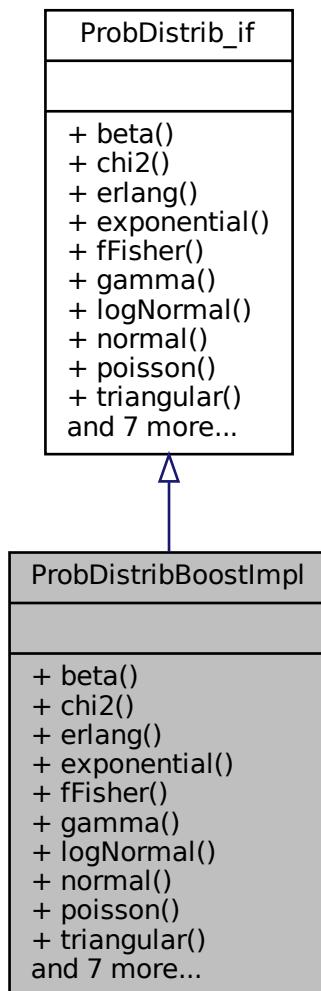
8.97 ProbDistribBoostImpl Class Reference

```
#include <ProbDistribBoostImpl.h>
```

Inheritance diagram for ProbDistribBoostImpl:



Collaboration diagram for ProbDistribBoostImpl:



Public Member Functions

- double **beta** (double x, double alpha, double beta)
- double **chi2** (double x, double m)
- double **erlang** (double x, double shape, double scale)
- double **exponential** (double x, double mean)
- double **fFisher** (double x, double k, double m)
- double **gamma** (double x, double shape, double scale)
- double **logNormal** (double x, double mean, double stddev)
- double **normal** (double x, double mean, double stddev)
- double **poisson** (double x, double mean)
- double **triangular** (double x, double min, double mode, double max)
- double **tStudent** (double x, double mean, double stddev, unsigned int degreeFreedom)
- double **uniform** (double x, double min, double max)
- double **weibull** (double x, double shape, double scale)

- double [inverseChi2](#) (double cumulativeProbability, double m)
- double [inverseFFisher](#) (double cumulativeProbability, double k, double m)
- double [inverseNormal](#) (double cumulativeProbability, double mean, double stddev)
- double [inverseTStudent](#) (double cumulativeProbability, double mean, double stddev, double degreeFreedom)

8.97.1 Detailed Description

Definition at line 22 of file [ProbDistribBoostImpl.h](#).

8.97.2 Member Function Documentation

8.97.2.1 beta() double ProbDistribBoostImpl::beta (

```
    double x,
    double alpha,
    double beta ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 52 of file [ProbDistribBoostImpl.cpp](#).

```
00052
00053     // beta_distribution<> my_beta(alpha, beta);
00054     return 0.0; // pdf(my_beta, x);
00055 }
```

8.97.2.2 chi2() double ProbDistribBoostImpl::chi2 (

```
    double x,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 82 of file [ProbDistribBoostImpl.cpp](#).

```
00082
00083     //chi_squared_distribution<> my_chi_squared(m);
00084     return 0.0; // pdf(my_chi_squared, x);
00085 }
```

```
8.97.2.3 erlang() double ProbDistribBoostImpl::erlang (
    double x,
    double shape,
    double scale ) [virtual]
```

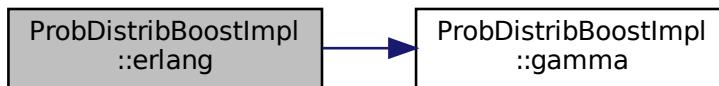
Implements [ProbDistrib_if](#).

Definition at line 36 of file [ProbDistribBoostImpl.cpp](#).

```
00036
00037     return ProbDistribBoostImpl::gamma(x, shape, scale); // \todo: NOT EXACTLY THIS... REDO
00038 }
```

References [gamma\(\)](#).

Here is the call graph for this function:



```
8.97.2.4 exponential() double ProbDistribBoostImpl::exponential (
    double x,
    double mean ) [virtual]
```

Implements [ProbDistrib_if](#).

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

```
00031
00032     assert(x >= 0);
00033     return 0.0; //mean * exp(-mean * x);
00034 }
```

```
8.97.2.5 fFisher() double ProbDistribBoostImpl::fFisher (
    double x,
    double k,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 77 of file [ProbDistribBoostImpl.cpp](#).

```
00077
00078     //fisher_f_distribution<> my_fisher_f(k, m);
00079     return 0.0; // pdf(my_fisher_f, x);
00080 }
```

```
8.97.2.6 gamma() double ProbDistribBoostImpl::gamma (
    double x,
    double shape,
    double scale ) [virtual]
```

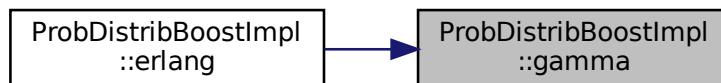
Implements [ProbDistrib_if](#).

Definition at line 47 of file [ProbDistribBoostImpl.cpp](#).

```
00047
00048     // gamma_distribution<> my_gamma(shape, scale);
00049     return 0.0; // pdf(my_gamma, x);
00050 }
```

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

Here is the caller graph for this function:



```
8.97.2.7 inverseChi2() double ProbDistribBoostImpl::inverseChi2 (
    double cumulativeProbability,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 107 of file [ProbDistribBoostImpl.cpp](#).

```
00107
00108     //chi_squared_distribution<> my_chi_squared(m);
00109     return 0.0; // quantile(my_chi_squared, cumulativeProbability);
00110 }
```

```
8.97.2.8 inverseFFisher() double ProbDistribBoostImpl::inverseFFisher (
    double cumulativeProbability,
    double k,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 102 of file [ProbDistribBoostImpl.cpp](#).

```
00102
00103     //fisher_f_distribution<> my_fisher_f(k, m);
00104     return 0.0; // quantile(my_fisher_f, cumulativeProbability);
00105 }
```

8.97.2.9 inverseNormal() double ProbDistribBoostImpl::inverseNormal (double cumulativeProbability, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 92 of file [ProbDistribBoostImpl.cpp](#).

```
00092
00093     //normal_distribution<> my_normal(mean, stddev);
00094     return 0.0; //quantile(my_normal, cumulativeProbability);
00095 }
```

8.97.2.10 inverseTStudent() double ProbDistribBoostImpl::inverseTStudent (double cumulativeProbability, double mean, double stddev, double degreeFreedom) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 97 of file [ProbDistribBoostImpl.cpp](#).

```
00097
00098     //students_t_distribution<> my_students_t(degreeFreedom);
00099     return 0.0; // quantile(my_students_t, cumulativeProbability);
00100 }
```

8.97.2.11 logNormal() double ProbDistribBoostImpl::logNormal (double x, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 62 of file [ProbDistribBoostImpl.cpp](#).

```
00062
00063     //lognormal_distribution<> my_lognormal(mean, stddev);
00064     return 0.0; // pdf(my_lognormal, x);
00065 }
```

8.97.2.12 normal() double ProbDistribBoostImpl::normal (double x, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 40 of file [ProbDistribBoostImpl.cpp](#).

```
00040
00041     double p1 = 1 / (stddev * std::sqrt(2 * M_PI));
00042     double rdf = (x - mean) / stddev;
00043     double p2 = std::exp(-0.5 * rdf * rdf);
00044     return p1*p2;
00045 }
```

8.97.2.13 poisson() double ProbDistribBoostImpl::poisson (double *x*, double *mean*) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 87 of file [ProbDistribBoostImpl.cpp](#).

```
00087
00088     //poisson_distribution<> my_poisson(mean);
00089     return 0.0; // pdf(my_poisson, x);
00090 }
```

8.97.2.14 triangular() double ProbDistribBoostImpl::triangular (

```
    double x,
    double min,
    double mode,
    double max ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 67 of file [ProbDistribBoostImpl.cpp](#).

```
00067
00068     //triangular_distribution<> my_triangular(min, mode, max);
00069     return 0.0; // pdf(my_triangular, x);
00070 }
```

8.97.2.15 tStudent() double ProbDistribBoostImpl::tStudent (

```
    double x,
    double mean,
    double stddev,
    unsigned int degreeFreedom ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 72 of file [ProbDistribBoostImpl.cpp](#).

```
00072
00073     {
00074         //students_t_distribution<> my_students_t(degreeFreedom);
00075         return 0.0; // pdf(my_students_t, x)*mean + stddev; //t-student SEEKS to be based on NORM(0,1)
00075 }
```

8.97.2.16 uniform() double ProbDistribBoostImpl::uniform (

```
    double x,
    double min,
    double max ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 24 of file [ProbDistribBoostImpl.cpp](#).

```
00024
00025     if (x >= min && x <= max)
00026         return 1.0 / (max - min);
00027     else
00028         return 0.0;
00029 }
```

```
8.97.2.17 weibull() double ProbDistribBoostImpl::weibull (
    double x,
    double shape,
    double scale ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 57 of file [ProbDistribBoostImpl.cpp](#).

```
00057
00058     // weibull_distribution<> my_weibull(shape, scale);
00059     return 0.0; // pdf(my_weibull, x);
00060 }
```

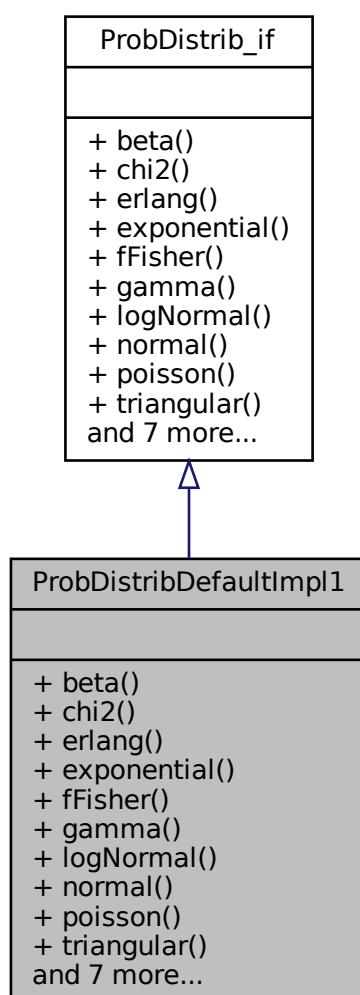
The documentation for this class was generated from the following files:

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

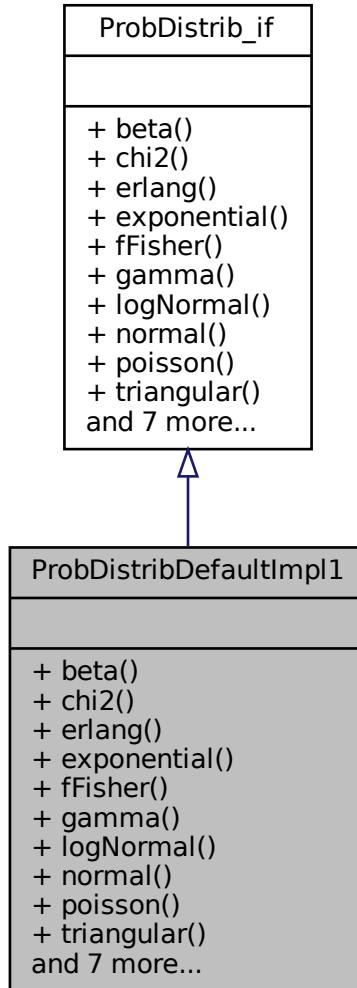
8.98 ProbDistribDefaultImpl1 Class Reference

```
#include <ProbDistribDefaultImpl1.h>
```

Inheritance diagram for ProbDistribDefaultImpl1:



Collaboration diagram for ProbDistribDefaultImpl1:



Public Member Functions

- double **beta** (double x, double alpha, double beta)
- double **chi2** (double x, double m)
- double **erlang** (double x, double shape, double scale)
- double **exponential** (double x, double mean)
- double **fFisher** (double x, double k, double m)
- double **gamma** (double x, double shape, double scale)
- double **logNormal** (double x, double mean, double stddev)
- double **normal** (double x, double mean, double stddev)
- double **poisson** (double x, double mean)
- double **triangular** (double x, double min, double mode, double max)
- double **tStudent** (double x, double mean, double stddev, unsigned int degreeFreedom)
- double **uniform** (double x, double min, double max)
- double **weibull** (double x, double shape, double scale)

- double [inverseChi2](#) (double cumulativeProbability, double m)
- double [inverseFFisher](#) (double cumulativeProbability, double k, double m)
- double [inverseNormal](#) (double cumulativeProbability, double mean, double stddev)
- double [inverseTStudent](#) (double cumulativeProbability, double mean, double stddev, double degreeFreedom)

8.98.1 Detailed Description

Definition at line 19 of file [ProbDistribDefaultImpl1.h](#).

8.98.2 Member Function Documentation

8.98.2.1 beta() double ProbDistribDefaultImpl1::beta (

```
    double x,
    double alpha,
    double beta ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 46 of file [ProbDistribDefaultImpl1.cpp](#).

```
00046
00047     // beta_distribution<> my_beta(alpha, beta);
00048     return 0.0; // pdf(my_beta, x);
00049 }
```

8.98.2.2 chi2() double ProbDistribDefaultImpl1::chi2 (

```
    double x,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 76 of file [ProbDistribDefaultImpl1.cpp](#).

```
00076
00077     //chi_squared_distribution<> my_chi_squared(m);
00078     return 0.0; // pdf(my_chi_squared, x);
00079 }
```

8.98.2.3 erlang() double ProbDistribDefaultImpl1::erlang (

```
    double x,
    double shape,
    double scale ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 30 of file [ProbDistribDefaultImpl1.cpp](#).

```
00030
00031     return ProbDistribDefaultImpl1::gamma(x, shape, scale); // \todo: NOT EXACTLY THIS... REDO
00032 }
```

References [gamma\(\)](#).

Here is the call graph for this function:



8.98.2.4 exponential() double ProbDistribDefaultImpl1::exponential (

```
    double x,
    double mean ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 25 of file [ProbDistribDefaultImpl1.cpp](#).

```
00025
00026     assert(x >= 0);
00027     return 0.0; //mean * exp(-mean * x);
00028 }
```

8.98.2.5 fFisher() double ProbDistribDefaultImpl1::fFisher (

```
    double x,
    double k,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 71 of file [ProbDistribDefaultImpl1.cpp](#).

```
00071
00072     //fisher_f_distribution<> my_fisher_f(k, m);
00073     return 0.0; // pdf(my_fisher_f, x);
00074 }
```

```
8.98.2.6 gamma() double ProbDistribDefaultImpl1::gamma (
    double x,
    double shape,
    double scale ) [virtual]
```

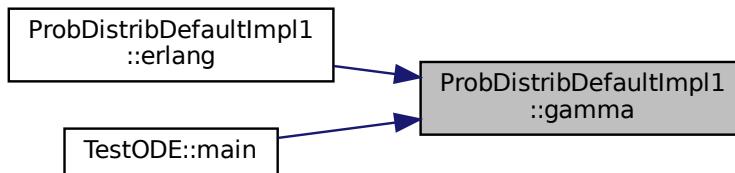
Implements [ProbDistrib_if](#).

Definition at line 41 of file [ProbDistribDefaultImpl1.cpp](#).

```
00041
00042     // gamma_distribution<> my_gamma(shape, scale);
00043     return 0.0; // pdf(my_gamma, x);
00044 }
```

Referenced by [erlang\(\)](#), and [TestODE::main\(\)](#).

Here is the caller graph for this function:



```
8.98.2.7 inverseChi2() double ProbDistribDefaultImpl1::inverseChi2 (
    double cumulativeProbability,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 101 of file [ProbDistribDefaultImpl1.cpp](#).

```
00101
00102     //chi_squared_distribution<> my_chi_squared(m);
00103     return 0.0; // quantile(my_chi_squared, cumulativeProbability);
00104 }
```

```
8.98.2.8 inverseFFisher() double ProbDistribDefaultImpl1::inverseFFisher (
    double cumulativeProbability,
    double k,
    double m ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 96 of file [ProbDistribDefaultImpl1.cpp](#).

```
00096
00097     //fisher_f_distribution<> my_fisher_f(k, m);
00098     return 0.0; // quantile(my_fisher_f, cumulativeProbability);
00099 }
```

8.98.2.9 inverseNormal() double ProbDistribDefaultImpl1::inverseNormal (double cumulativeProbability, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 86 of file [ProbDistribDefaultImpl1.cpp](#).

```
00086 {  
00087     //normal_distribution<> my_normal(mean, stddev);  
00088     return 0.0; //quantile(my_normal, cumulativeProbability);  
00089 }
```

8.98.2.10 inverseTStudent() double ProbDistribDefaultImpl1::inverseTStudent (double cumulativeProbability, double mean, double stddev, double degreeFreedom) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 91 of file [ProbDistribDefaultImpl1.cpp](#).

```
00091 {  
00092     //students_t_distribution<> my_students_t(degreeFreedom);  
00093     return 0.0; // quantile(my_students_t, cumulativeProbability);  
00094 }
```

8.98.2.11 logNormal() double ProbDistribDefaultImpl1::logNormal (double x, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 56 of file [ProbDistribDefaultImpl1.cpp](#).

```
00056 {  
00057     //lognormal_distribution<> my_lognormal(mean, stddev);  
00058     return 0.0; // pdf(my_lognormal, x);  
00059 }
```

8.98.2.12 normal() double ProbDistribDefaultImpl1::normal (double x, double mean, double stddev) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 34 of file [ProbDistribDefaultImpl1.cpp](#).

```
00034 {  
00035     double p1 = 1 / (stddev * std::sqrt(2 * M_PI));  
00036     double rdf = (x - mean) / stddev;  
00037     double p2 = std::exp(-0.5 * rdf * rdf);  
00038     return p1*p2;  
00039 }
```

8.98.2.13 poisson() double ProbDistribDefaultImpl1::poisson (double *x*, double *mean*) [virtual]

Implements [ProbDistrib_if](#).

Definition at line 81 of file [ProbDistribDefaultImpl1.cpp](#).

```
00081
00082     //poisson_distribution<> my_poisson(mean);
00083     return 0.0; // pdf(my_poisson, x);
00084 }
```

8.98.2.14 triangular() double ProbDistribDefaultImpl1::triangular (

```
double x,
double min,
double mode,
double max ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 61 of file [ProbDistribDefaultImpl1.cpp](#).

```
00061
00062     //triangular_distribution<> my_triangular(min, mode, max);
00063     return 0.0; // pdf(my_triangular, x);
00064 }
```

8.98.2.15 tStudent() double ProbDistribDefaultImpl1::tStudent (

```
double x,
double mean,
double stddev,
unsigned int degreeFreedom ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 66 of file [ProbDistribDefaultImpl1.cpp](#).

```
00066
00067     {
00068         //students_t_distribution<> my_students_t(degreeFreedom);
00069         return 0.0; // pdf(my_students_t, x)*mean + stddev; //t-student SEEKS to be based on NORM(0,1)
00069 }
```

8.98.2.16 uniform() double ProbDistribDefaultImpl1::uniform (

```
double x,
double min,
double max ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 18 of file [ProbDistribDefaultImpl1.cpp](#).

```
00018
00019     if (x >= min && x <= max)
00020         return 1.0 / (max - min);
00021     else
00022         return 0.0;
00023 }
```

```
8.98.2.17 weibull() double ProbDistribDefaultImpl1::weibull (
    double x,
    double shape,
    double scale ) [virtual]
```

Implements [ProbDistrib_if](#).

Definition at line 51 of file [ProbDistribDefaultImpl1.cpp](#).

```
00051
00052     // weibull_distribution<> my_weibull(shape, scale);
00053     return 0.0; // pdf(my_weibull, x);
00054 }
```

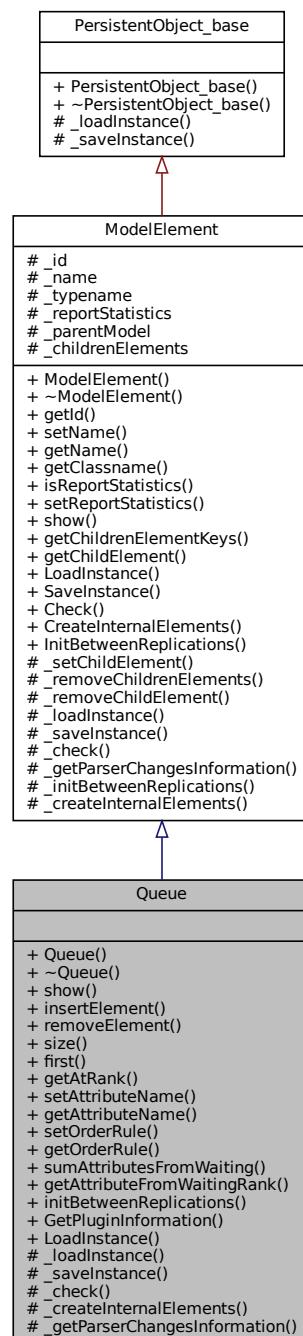
The documentation for this class was generated from the following files:

- [ProbDistribDefaultImpl1.h](#)
- [ProbDistribDefaultImpl1.cpp](#)

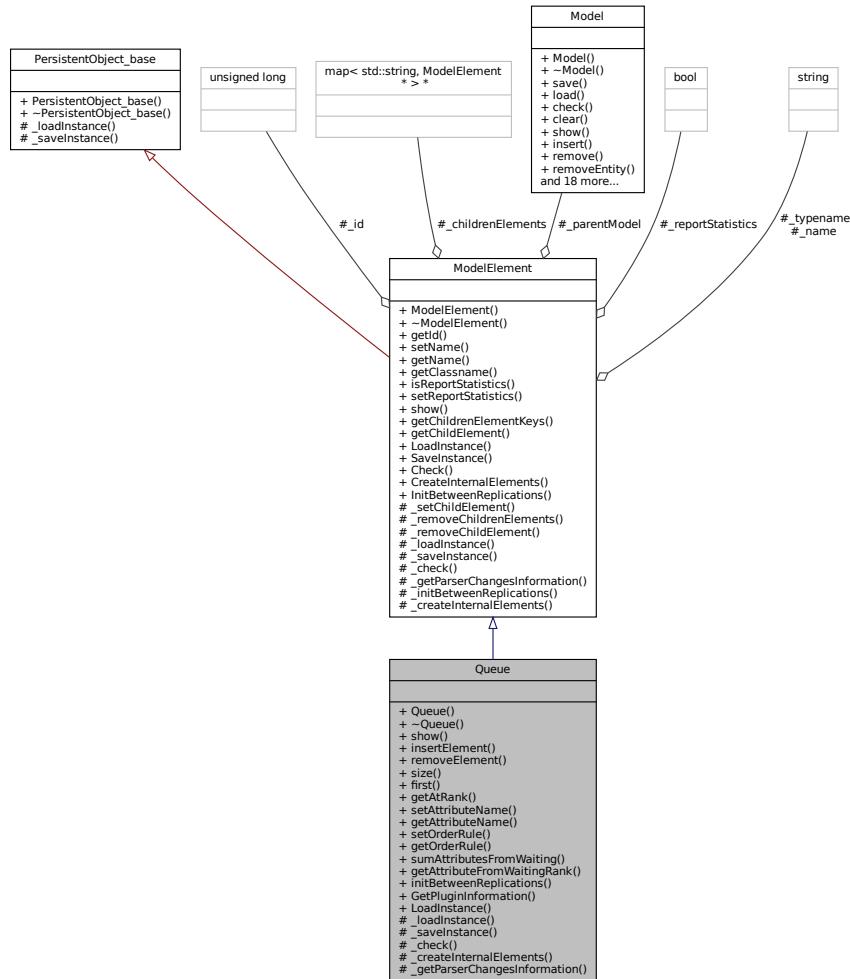
8.99 Queue Class Reference

```
#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 (Model *model, std::string name="")`
- `virtual ~Queue ()`
- `virtual std::string show ()`
- `void insertElement (Waiting *element)`
- `void removeElement (Waiting *element)`
- `unsigned int size ()`
- `Waiting * first ()`
- `Waiting * getAtRank (unsigned int rank)`
- `void setAttributeName (std::string _attributeName)`
- `std::string getAttributeName () const`
- `void setOrderRule (OrderRule _orderRule)`

- `Queue::OrderRule getOrderRule () const`
- `double sumAttributesFromWaiting (Util::identification attributeID)`
- `double getAttributeFromWaitingRank (unsigned int rank, Util::identification attributeID)`
- `void initBetweenReplications ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

- virtual `ParserChangesInformation * _getParserChangesInformation ()`

Additional Inherited Members

8.99.1 Detailed Description

Queue module DESCRIPTION This data module may be utilized to change the ranking rule for a specified queue. The default ranking rule for all queues is First In, First Out unless otherwise specified in this module. There is an additional field that allows the queue to be defined as shared. TYPICAL USES Stack of work waiting for a resource at a Process module Holding area for documents waiting to be collated at a **Batch** module Prompt Description Name The name of the queue whose characteristics are being defined. This name must be unique. Type Ranking rule for the queue, which can be based on an attribute. Types include First In, First Out; Last In, First Out; Lowest **Attribute** Value (first); and Highest **Attribute** Value (first). A low attribute value would be 0 or 1, while a high value may be 200 or 300. **Attribute** Name **Attribute** that will be evaluated for the Lowest **Attribute** Value or Highest **Attribute** Value types. Entities with lowest or highest values of the attribute will be ranked first in the queue, with ties being broken using the First In, First Out rule. Shared Check box that determines whether a specific queue is used in multiple places within the simulation model. Shared queues can only be used for seizing resources (for example, with the **Seize** module from the Advanced Process panel). Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this queue.

Definition at line 91 of file `Queue.h`.

8.99.2 Member Enumeration Documentation

8.99.2.1 OrderRule enum `Queue::OrderRule : int [strong]`

Enumerator

FIFO	
LIFO	
HIGHESTVALUE	
SMALLESTVALUE	

Definition at line 94 of file Queue.h.

```
00094     : int {
00095         FIFO = 1, LIFO = 2, HIGHESTVALUE = 3, SMALLESTVALUE = 4
00096     };

```

8.99.3 Constructor & Destructor Documentation

8.99.3.1 Queue() Queue::Queue (

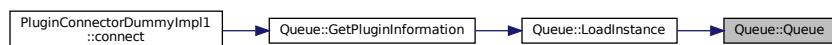
```
Model * model,
std::string name = "" )
```

Definition at line 18 of file Queue.cpp.

```
00018 : ModelElement(model, Util::TypeOf<Queue>(), name) {
00019 }
```

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

Here is the caller graph for this function:



8.99.3.2 ~Queue() Queue::~Queue () [virtual]

Definition at line 21 of file Queue.cpp.

```
00021     {
00022         //__parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), __cstatNumberInQueue);
00023         //__parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), __cstatTimeInQueue);
00024     }
```

8.99.4 Member Function Documentation

```
8.99.4.1 _check() bool Queue::_check (
    std::string * errorMessage) [protected], [virtual]
```

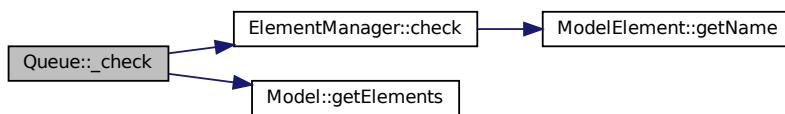
Reimplemented from [ModelElement](#).

Definition at line 129 of file [Queue.cpp](#).

```
00129
00130     return _parentModel->getElements()->check(Util::TypeOf<Attribute>(),
00131         "AttributeName", false, errorMessage);
00131 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



```
8.99.4.2 _createInternalElements() void Queue::_createInternalElements () [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

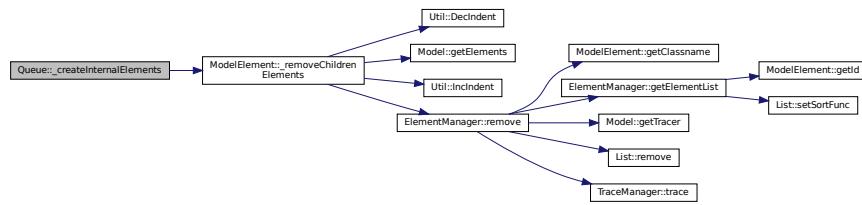
Reimplemented from [ModelElement](#).

Definition at line 133 of file [Queue.cpp](#).

```
00133
00134     if (_reportStatistics) {
00135         if (_cstatNumberInQueue == nullptr) {
00136             _cstatNumberInQueue = new StatisticsCollector(_parentModel, _name + "." + "NumberInQueue",
00137                 this); /* \todo: ++ WHY THIS INSERT "DISPOSE" AND "10ENTITYTYPE" STATCOLL ?? */
00138             _cstatTimeInQueue = new StatisticsCollector(_parentModel, _name + "." + "TimeInQueue",
00139                 this);
00140             _childrenElements->insert({"NumberInQueue", _cstatNumberInQueue});
00141             _childrenElements->insert({"TimeInQueue", _cstatTimeInQueue});
00142         } else if (_cstatNumberInQueue != nullptr) {
00143             // \todo: remove
00144             _removeChildrenElements();
00145             //_childrenElements->remove(_cstatNumberInQueue);
00146             //_childrenElements->remove(_cstatTimeInQueue);
00147             //_cstatNumberInQueue->~StatisticsCollector();
00148             //_cstatTimeInQueue->~StatisticsCollector();
00149     }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



8.99.4.3 _getParserChangesInformation() `ParserChangesInformation * Queue::_getParserChangesInformation() [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 151 of file [Queue.cpp](#).

```

00151
00152     ParserChangesInformation* changes = new ParserChangesInformation();
00153     //changes->getProductionToAdd()->insert(...);
00154     //changes->getTokensToAdd() ->insert(...);
00155     return changes;
00156 }
  
```

8.99.4.4 _loadInstance() `bool Queue::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 110 of file [Queue.cpp](#).

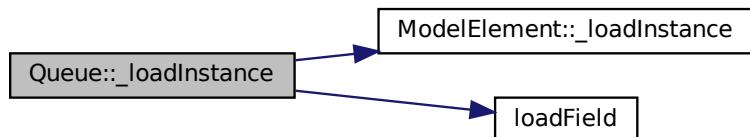
```

00110
00111     bool res = ModelElement::_loadInstance(fields);
00112     if (res) {
00113         try {
00114             this->_attributeName = loadField(fields, "attributeName", "");
00115             this->_orderRule = static_cast<OrderRule> (std::stoi(loadField(fields, "orderRule",
00116             std::to_string(static_cast<int> (OrderRule::FIFO)))));
00117         } catch (...) {
00118     }
00119     return res;
00120 }
  
```

References [ModelElement::_loadInstance\(\)](#), [FIFO](#), and [loadField\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.99.4.5 `_saveInstance()` `std::map< std::string, std::string > * Queue::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 122 of file [Queue.cpp](#).

```

00122
00123     std::map<std::string, std::string>* fields = ModelElement::\_saveInstance\(\);
00124     //Util::TypeOf<Queue>());
00125     if (_orderRule != OrderRule::FIFO) fields->emplace("orderRule", std::to_string(static\_cast<int>
00126     (

```

References [ModelElement::_saveInstance\(\)](#), and [FIFO](#).

Here is the call graph for this function:



8.99.4.6 `first()` `Waiting * Queue::first ()`

Definition at line 55 of file [Queue.cpp](#).

```

00055
00056     {
00056     return _list->front\(\);
00057 }
  
```

References [List< T >::front\(\)](#).

Here is the call graph for this function:



8.99.4.7 `getAtRank()` `Waiting * Queue::getAtRank (`
`unsigned int rank)`

Definition at line 59 of file [Queue.cpp](#).

```
00059
00060     return _list->getAtRank(rank);
00061 }
```

References [List< T >::getAtRank\(\)](#).

Here is the call graph for this function:



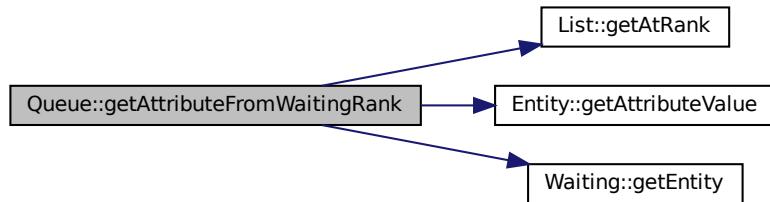
8.99.4.8 `getAttributeFromWaitingRank()` `double Queue::getAttributeFromWaitingRank (`
`unsigned int rank,`
`Util::identification attributeID)`

Definition at line 87 of file [Queue.cpp](#).

```
00087
00088     Waiting* wait = _list->getAtRank(rank);
00089     if (wait != nullptr) {
00090         return wait->getEntity()->getAttributeValue(attributeID);
00091     }
00092     return 0.0;
00093 }
```

References [List< T >::getAtRank\(\)](#), [Entity::getAttributeValue\(\)](#), and [Waiting::getEntity\(\)](#).

Here is the call graph for this function:



8.99.4.9 getAttributeName() `std::string Queue::getAttributeName() const`Definition at line 67 of file [Queue.cpp](#).

```
00067
00068     return _attributeName;
00069 }
```

8.99.4.10 getOrderRule() `Queue::OrderRule Queue::getOrderRule() const`Definition at line 75 of file [Queue.cpp](#).

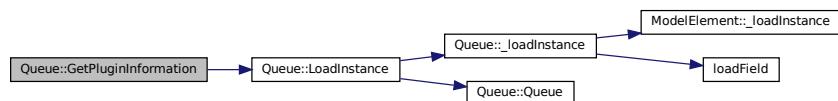
```
00075
00076     return _orderRule;
00077 }
```

8.99.4.11 GetPluginInformation() `PluginInformation * Queue::GetPluginInformation() [static]`Definition at line 95 of file [Queue.cpp](#).

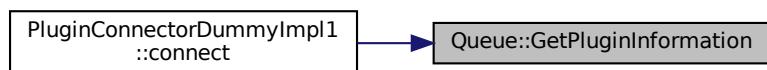
```
00095
00096     PluginInformation* info = new PluginInformation(Util::TypeOf<Queue>(), &Queue::LoadInstance);
00097     return info;
00098 }
```

References [LoadInstance\(\)](#).Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.99.4.12 initBetweenReplications() void Queue::initBetweenReplications ()

Definition at line 47 of file [Queue.cpp](#).

```
00047     {  
00048         this->_list->clear();  
00049     }
```

References [List< T >::clear\(\)](#).

Referenced by [Seize::_initBetweenReplications\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.99.4.13 insertElement()** void Queue::insertElement (Waiting * element)

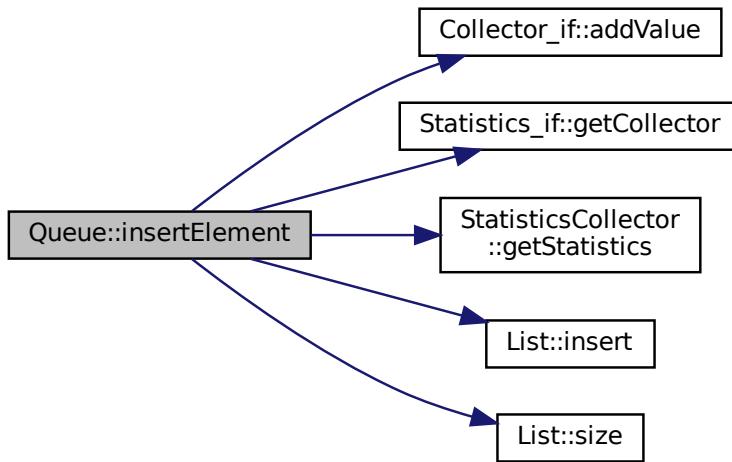
Definition at line 31 of file [Queue.cpp](#).

```
00031     {  
00032         _list->insert(element);  
00033         if (_reportStatistics)  
00034             this->_cstatNumberInQueue->getStatistics()->getCollector()->addValue(_list->size());  
00035     }
```

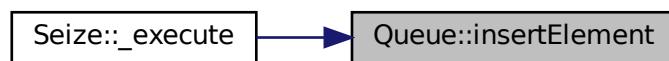
References [ModelElement::_reportStatistics](#), [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [List< T >::insert\(\)](#), and [List< T >::size\(\)](#).

Referenced by [Seize::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```

8.99.4.14 LoadInstance() ModelElement * Queue::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
  
```

Definition at line 100 of file [Queue.cpp](#).

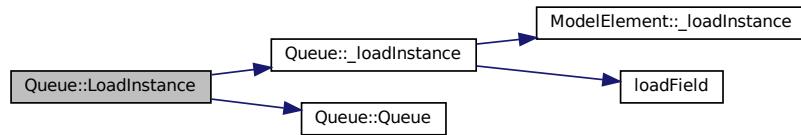
```

00100
00101     Queue* newElement = new Queue(model);
00102     try {
00103         newElement->loadInstance(fields);
00104     } catch (const std::exception& e) {
00105
00106     }
00107     return newElement;
00108 }
  
```

References [_loadInstance\(\)](#), and [Queue\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.99.4.15 removeElement() void Queue::removeElement (Waiting * element)

Definition at line 37 of file [Queue.cpp](#).

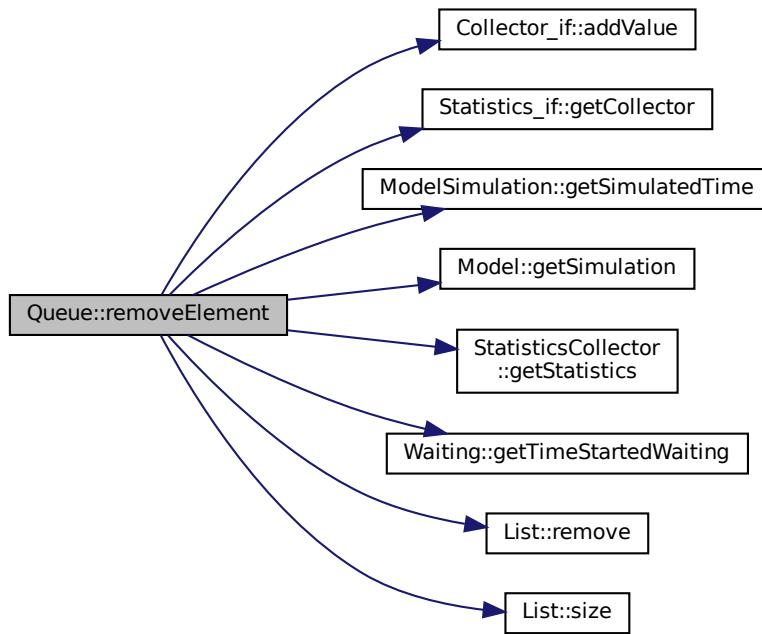
```

00037     {
00038     double tnow = _parentModel->getSimulation()->getSimulatedTime();
00039     _list->remove(element);
00040     if (_reportStatistics) {
00041         this->_cstatNumberInQueue->getStatistics()->getCollector()->addValue(_list->size());
00042         double timeInQueue = tnow - element->getTimeStartedWaiting();
00043         this->_cstatTimeInQueue->getStatistics()->getCollector()->addValue(timeInQueue);
00044     }
00045 }

```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [Waiting::getTimeStartedWaiting\(\)](#), [List< T >::remove\(\)](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.99.4.16 setAttributeName() `void Queue::setAttributeName (std::string _attributeName)`

Definition at line 63 of file [Queue.cpp](#).

```

00063
00064     this->_attributeName = _attributeName;
00065 }
```

8.99.4.17 setOrderRule() `void Queue::setOrderRule (OrderRule _orderRule)`

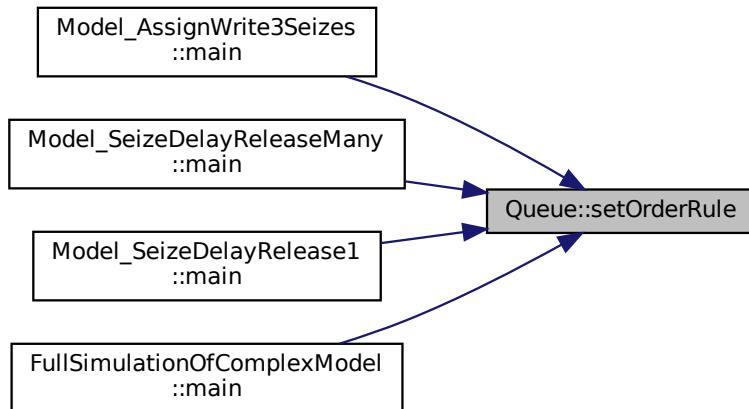
Definition at line 71 of file [Queue.cpp](#).

```

00071
00072     this->_orderRule = _orderRule;
00073 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.99.4.18 `show()` std::string Queue::show () [virtual]

Reimplemented from [ModelElement](#).

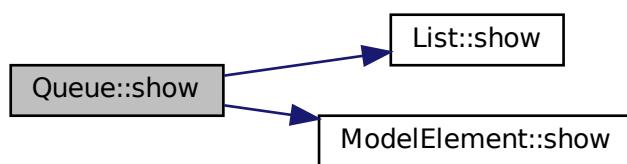
Definition at line 26 of file [Queue.cpp](#).

```

00026     {
00027     return ModelElement::show() +
00028         ",waiting=" + this->_list->show();
00029 }
```

References [List< T >::show\(\)](#), and [ModelElement::show\(\)](#).

Here is the call graph for this function:



8.99.4.19 size() `unsigned int Queue::size ()`

Definition at line 51 of file [Queue.cpp](#).

```
00051     {
00052         return _list->size();
00053     }
```

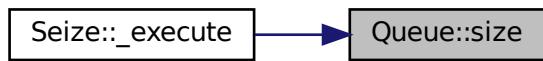
References [List< T >::size\(\)](#).

Referenced by [Seize::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.99.4.20 sumAttributesFromWaiting()** `double Queue::sumAttributesFromWaiting (`
 `Util::identification attributeID)`

Definition at line 79 of file [Queue.cpp](#).

```
00079
00080     double sum = 0.0;
00081     for (std::list<Waiting*>::iterator it = _list->list()->begin(); it != _list->list()->end(); it++)
00082     {
00083         sum += (*it)->getEntity()->getAttributeValue(attributeID);
00084     }
00085 }
```

References [List< T >::list\(\)](#).

Here is the call graph for this function:



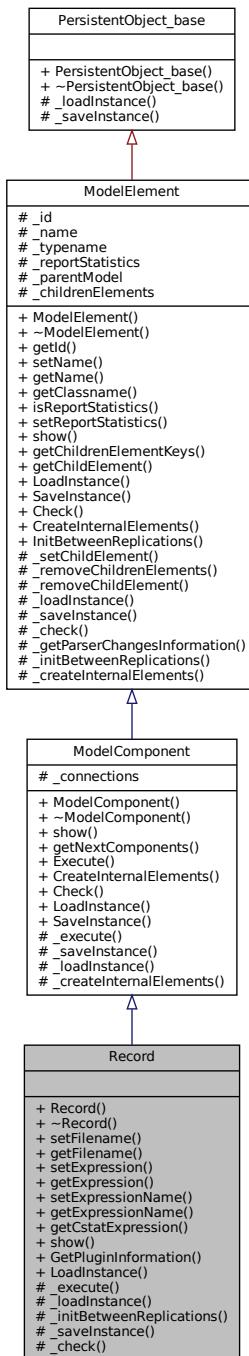
The documentation for this class was generated from the following files:

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

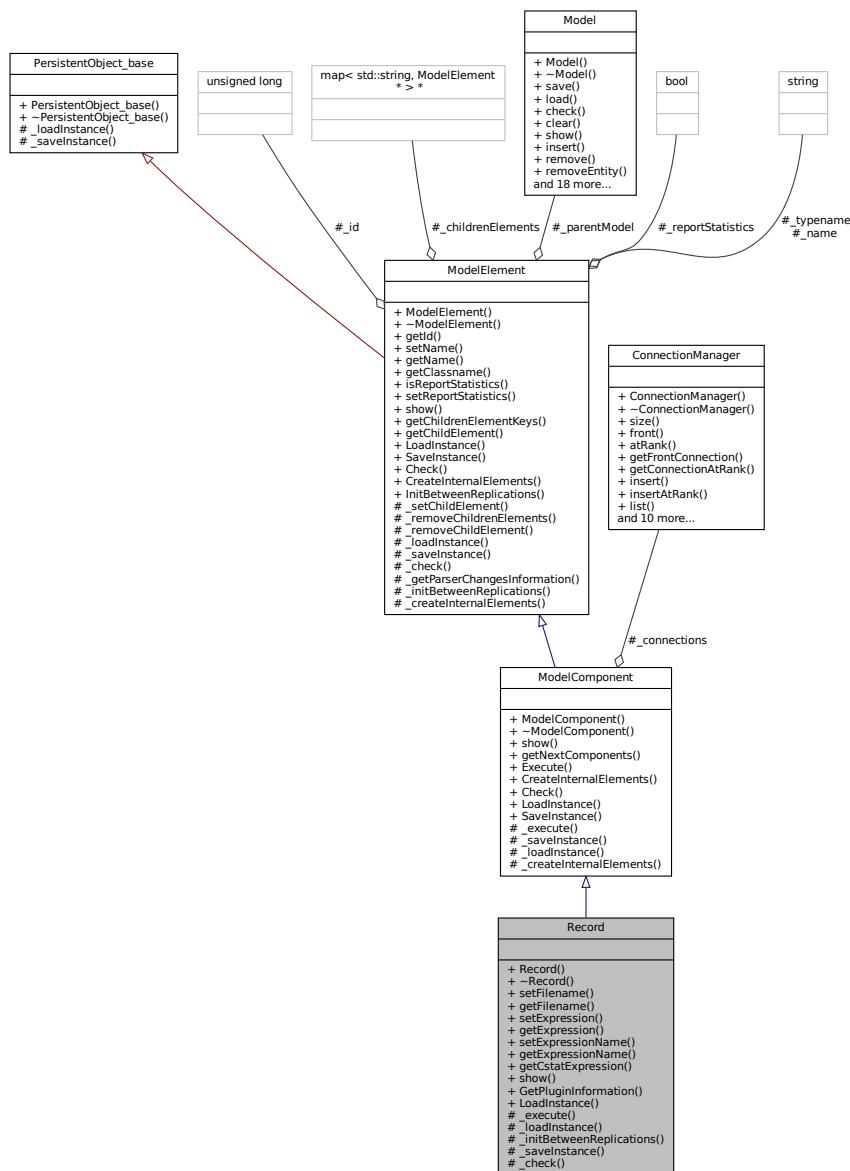
8.100 Record Class Reference

```
#include <Record.h>
```

Inheritance diagram for Record:



Collaboration diagram for Record:



Public Member Functions

- **Record (Model *model, std::string name="")**
- virtual **~Record ()**
- void **setFilename (std::string filename)**
- std::string **getFilename () const**
- void **setExpression (std::string expression)**
- std::string **getExpression () const**
- void **setExpressionName (std::string expressionName)**
- std::string **getExpressionName () const**
- **StatisticsCollector * getCstatExpression () const**
- virtual std::string **show ()**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.100.1 Detailed Description

Record module DESCRIPTION This module is used to collect statistics in the simulation model. Various types of observational statistics are available, including time between exits through the module, entity statistics (such as time or costing), general observations, and interval statistics (from some time stamp to the current simulation time). A count type of statistic is available as well. Tally and **Counter** sets can also be specified. TYPICAL USES Collect the number of jobs completed each hour Count how many orders have been late being fulfilled **Record** the time spent by priority customers in the main check-out line PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Type of observational (tally) or count statistic to be generated. Count will increase or decrease the value of the named statistic by the specified value. Entity Statistics will generate general entity statistics, such as time and costing/duration information. Time Interval will calculate and record the difference between a specified attribute's value and current simulation time. Time Between will track and record the time between entities entering the module. Expression will record the value of the specified expression. Attribute Name Name of the attribute whose value will be used for the interval statistics. Applies only when Type is Interval. Value Value that will be recorded to the observational statistic when Type is Expression or added to the counter when Type is Count. Tally Name This field defines the symbol name of the tally into which the observation is to be recorded. Applies only when Type is Time Interval, Time Between, or Expression. Counter This field defines the symbol name of the counter to Name increment/decrement. Applies only when Type is Counter. Record into Set Check box to specify whether or not a tally or counter set will be used. Tally Set Name Name of the tally set that will be used to record the observational-type statistic. Applies only when Type is Time Interval, Time Between, or Expression. Counter Set Name Name of the counter set that will be used to record the count-type statistic. Applies only when Type is Count. Set Index Index into the tally or counter set.

Definition at line 62 of file `Record.h`.

8.100.2 Constructor & Destructor Documentation

```
8.100.2.1 Record() Record::Record (
    Model * model,
    std::string name = "" )
```

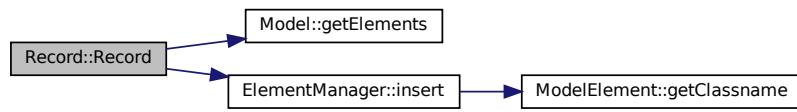
Definition at line 20 of file [Record.cpp](#).

```
00020 : ModelComponent(model, Util::TypeOf<Record>(), name) {
00021     _cstatExpression = new StatisticsCollector(_parentModel, _expressionName, this);
00022     _parentModel->getElements()->insert(_cstatExpression);
00023 }
```

References [ModelElement::_parentModel](#), [Model::getElements\(\)](#), and [ElementManager::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



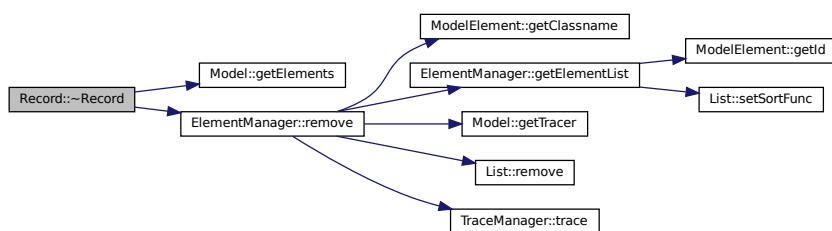
8.100.2.2 ~Record() Record::~Record () [virtual]

Definition at line 25 of file [Record.cpp](#).

```
00025 {
00026     _parentModel->getElements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatExpression);
00027 }
```

References [ModelElement::_parentModel](#), [Model::getElements\(\)](#), and [ElementManager::remove\(\)](#).

Here is the call graph for this function:



8.100.3 Member Function Documentation

8.100.3.1 `_check()` `bool Record::_check (std::string * errorMessage) [protected], [virtual]`

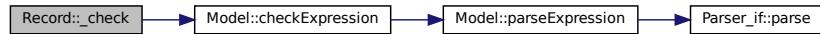
Reimplemented from [ModelElement](#).

Definition at line 98 of file [Record.cpp](#).

```
00098                                     {
00099     // when cheking the model (before simulating it), remove the file if exists
00100     std::remove(_filename.c_str());
00101     return _parentModel->checkExpression(_expression, "expression", errorMessage);
00102 }
```

References [ModelElement::_parentModel](#), and [Model::checkExpression\(\)](#).

Here is the call graph for this function:



8.100.3.2 `_execute()` `void Record::_execute (Entity * entity) [protected], [virtual]`

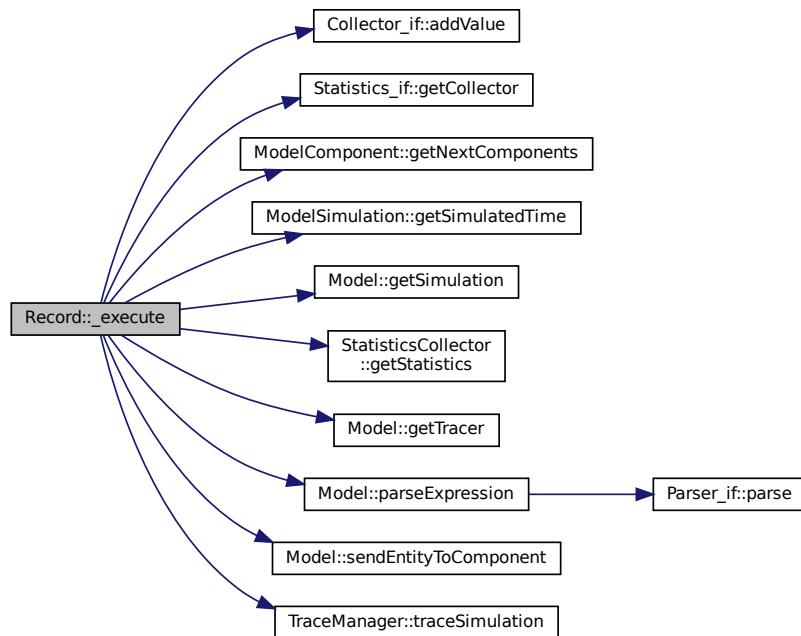
Implements [ModelComponent](#).

Definition at line 65 of file [Record.cpp](#).

```
00065                                     {
00066     double value = _parentModel->parseExpression(_expression);
00067     _cstatExpression->getStatistics()->getCollector()->addValue(value);
00068     std::ofstream file;
00069     file.open(_filename, std::ofstream::out | std::ofstream::app);
00070     file << value << std::endl;
00071     file.close(); // \todo: open and close for every data is not a good idea. Should open when
00072     // replication starts and close when it finishes.
00073     _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00074     entity, this, "Recording value " + std::to_string(value));
00075     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00076 }
```

References [ModelElement::_parentModel](#), [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [ModelComponent::getNextComponents\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [Model::getTracer\(\)](#), [Model::parseExpression\(\)](#), [Model::sendEntityToComponent\(\)](#), and [TraceManager::traceSimulation\(\)](#).

Here is the call graph for this function:



8.100.3.3 `_initBetweenReplications()` void Record::_initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 95 of file [Record.cpp](#).

```
00095
00096 }
```

8.100.3.4 `_loadInstance()` bool Record::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]

Reimplemented from [ModelComponent](#).

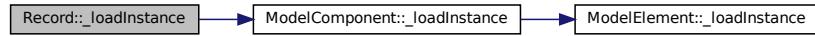
Definition at line 85 of file [Record.cpp](#).

```
00085
00086     bool res = ModelComponent::_loadInstance(fields);
00087     if (res) {
00088         this->_expression = ((*fields->find("expression0")) .second);
00089         this->_expressionName = ((*fields->find("expressionName0")) .second);
00090         this->_filename = ((*fields->find("fileName0")) .second);
00091     }
00092     return res;
00093 }
```

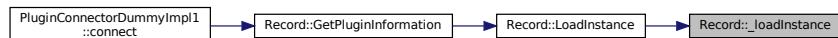
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.100.3.5 _saveInstance() `std::map< std::string, std::string > * Record::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelComponent](#).

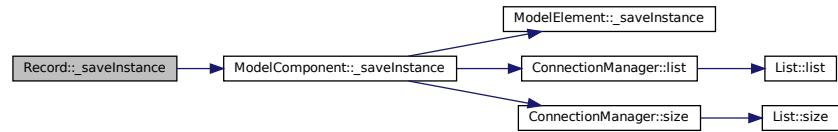
Definition at line 77 of file [Record.cpp](#).

```

00077
00078     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);
00079     //Util::TypeOf<Record>();
00080     fields->emplace("expression0", "\"" + this->_expression + "\"");
00081     fields->emplace("expressionName0", this->_expressionName);
00082     fields->emplace("fileName0", "\"" + this->_filename + "\"");
00083     return fields;
00084 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.100.3.6 getCstatExpression() `StatisticsCollector * Record::getCstatExpression () const`

Definition at line 45 of file [Record.cpp](#).

```

00045
00046     return _cstatExpression;
00047 }
```

8.100.3.7 getExpression() std::string Record::getExpression () constDefinition at line 61 of file [Record.cpp](#).

```
00061
00062     return _expression;
00063 }
```

8.100.3.8 getExpressionName() std::string Record::getExpressionName () constDefinition at line 41 of file [Record.cpp](#).

```
00041
00042     return _expressionName;
00043 }
```

8.100.3.9 getFiledname() std::string Record::getFiledname () constDefinition at line 53 of file [Record.cpp](#).

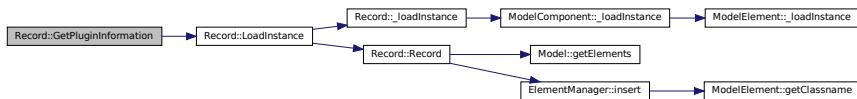
```
00053
00054     return _filename;
00055 }
```

8.100.3.10 GetPluginInformation() [PluginInformation](#) * Record::GetPluginInformation () [static]Definition at line 104 of file [Record.cpp](#).

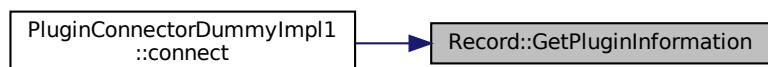
```
00104
00105     PluginInformation* info = new PluginInformation(Util::TypeOf<Record>(), &Record::LoadInstance);
00106     return info;
00107 }
```

References [LoadInstance\(\)](#).Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.100.3.11 LoadInstance() ModelComponent * Record::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

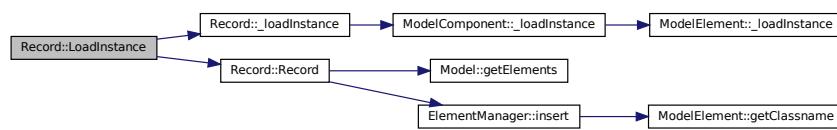
Definition at line 109 of file [Record.cpp](#).

```
00109
00110     Record* newComponent = new Record(model);
00111     try {
00112         newComponent->_loadInstance(fields);
00113     } catch (const std::exception& e) {
00114     }
00115 }
00116     return newComponent;
00117
00118 }
```

References [_loadInstance\(\)](#), and [Record\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



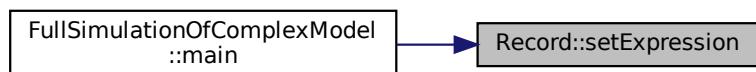
```
8.100.3.12 setExpression() void Record::setExpression (
    std::string expression )
```

Definition at line 57 of file [Record.cpp](#).

```
00057
00058     this->_expression = expression;
00059 }
```

Referenced by [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.100.3.13 setExpressionName() void Record::setExpressionName (std::string expressionName)

Definition at line 36 of file [Record.cpp](#).

```
00036 {  
00037     this->_expressionName = expressionName;  
00038     this->_cstatExpression->setName(expressionName);  
00039 }
```

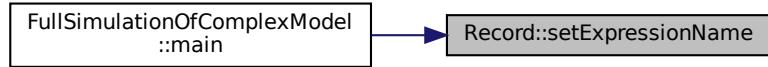
References [ModelElement::setName\(\)](#).

Referenced by [FullSimulationOfComplexModel::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



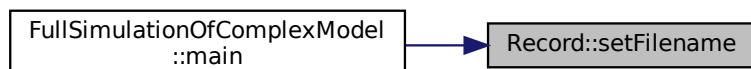
8.100.3.14 setFilename() void Record::setFilename (std::string filename)

Definition at line 49 of file [Record.cpp](#).

```
00049 {  
00050     this->_filename = filename;  
00051 }
```

Referenced by [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.100.3.15 show() std::string Record::show () [virtual]

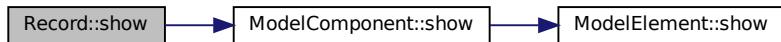
Reimplemented from [ModelComponent](#).

Definition at line 29 of file [Record.cpp](#).

```
00029         {
00030     return ModelComponent::show() +
00031             ",expressionName=\"" + this->_expressionName + "\"\" +
00032             ",expression=\"" + _expression + "\"\" +
00033             "filename=\"" + _filename + "\"\";
00034 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



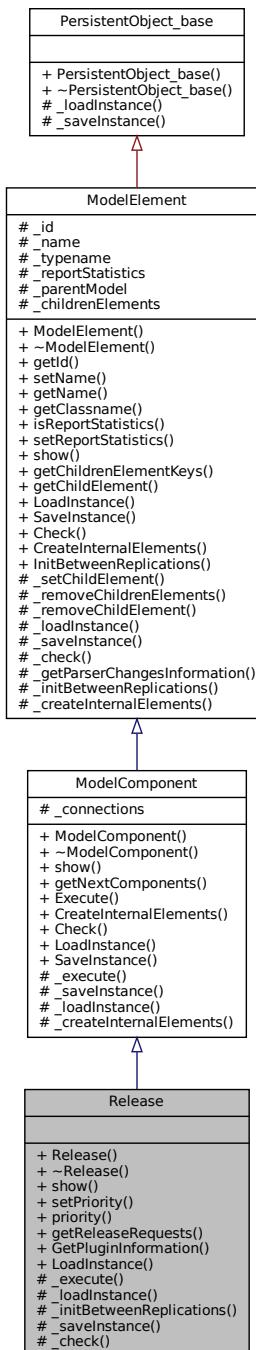
The documentation for this class was generated from the following files:

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

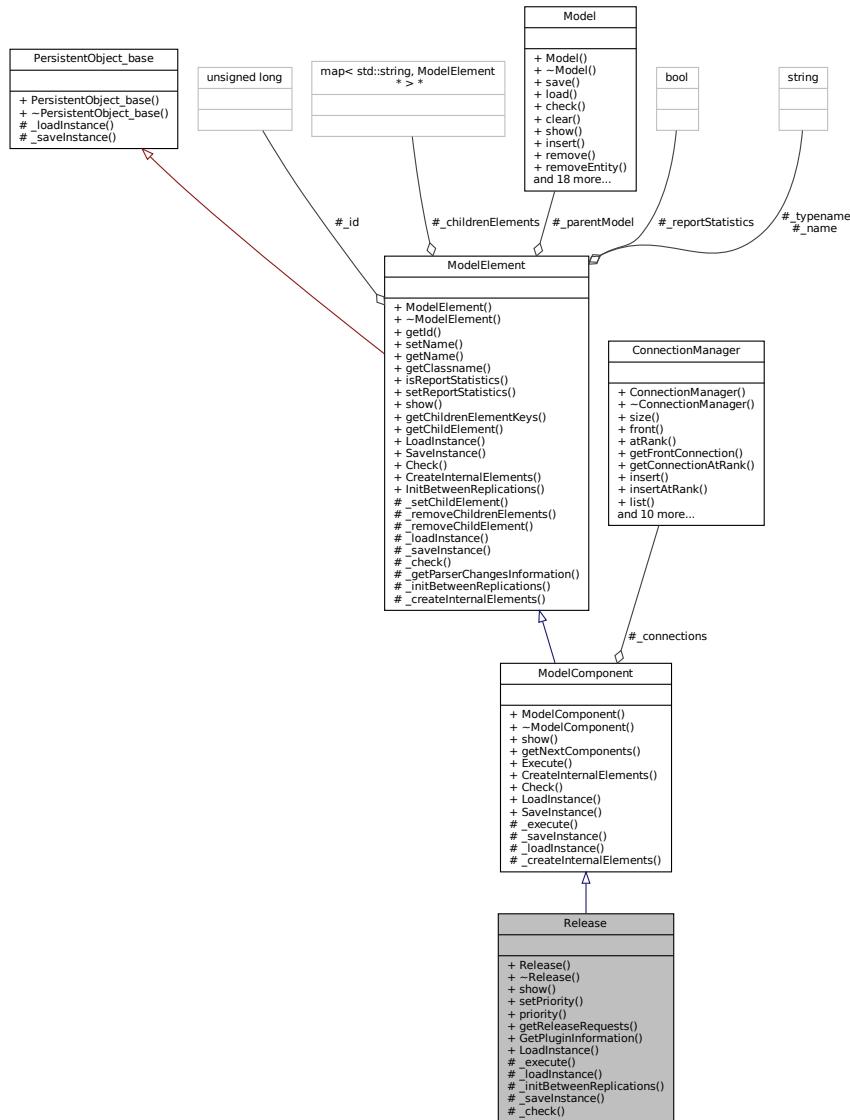
8.101 Release Class Reference

```
#include <Release.h>
```

Inheritance diagram for Release:



Collaboration diagram for Release:



Public Member Functions

- `Release (Model *model, std::string name="")`
- `virtual ~Release ()=default`
- `virtual std::string show ()`
- `void setPriority (unsigned short _priority)`
- `unsigned short priority () const`
- `List< SeizableItemRequest * > * getReleaseRequests () const`

Static Public Member Functions

- `static PluginInformation * GetPluginInformation ()`
- `static ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.101.1 Detailed Description

Release module DESCRIPTION The **Release** module is used to release units of a resource that an entity previously has seized. This module may be used to release individual resources or may be used to release resources within a set. For each resource to be released, the name and quantity to release are specified. When the entity enters the **Release** module, it gives up control of the specified resource(s). Any entities waiting in queues for those resources will gain control of the resources immediately. TYPICAL USES Finishing a customer order (release the operator) Completing a tax return (release the accountant) Leaving the hospital (release the doctor, nurse, hospital room) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Type of resource for releasing, either specifying a particular resource, or selecting from a pool of resources (that is, a resource set). The resource name may also be specified by an expression or attribute value. Resource Name Name of the resource that will be released. Set Name Name of the resource set from which a member will be released. Attribute Name Name of the attribute that specifies the resource name to be released. Expression Name of the expression that specifies the name of the resource to be released. Quantity Number of resources of a given name or from a given set that will be released. For sets, this value specifies only the number of a selected resource that will be released (based on the resource's capacity), not the number of members to be released within the set. Rule Method of determining which resource within a set to release. Last Member Seized and First Member Seized will release the last/first member from within the set that was seized. Specific member indicates that a member number or attribute (with a member number value) will be used to specify the member to release. Set Index Member index of the resource set that the entity will release.

Definition at line 64 of file [Release.h](#).

8.101.2 Constructor & Destructor Documentation

8.101.2.1 Release() `Release::Release (`
`Model * model,`
`std::string name = "")`

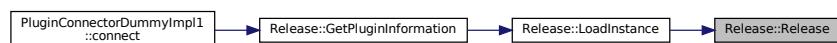
Definition at line 20 of file [Release.cpp](#).

```
00020 : ModelComponent (model, Util::TypeOf<Release>(), name) {  

00021 }
```

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

Here is the caller graph for this function:



8.101.2.2 ~Release() virtual Release::~Release () [virtual], [default]

8.101.3 Member Function Documentation

8.101.3.1 _check() bool Release::_check (std::string * errorMessage) [protected], [virtual]

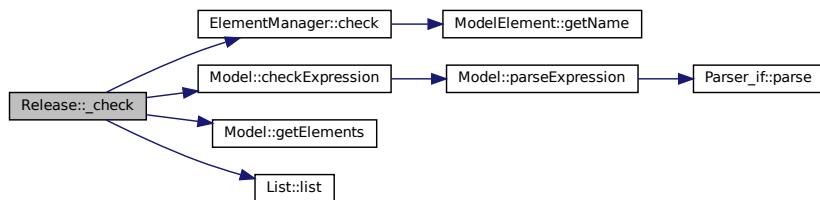
Reimplemented from [ModelElement](#).

Definition at line 113 of file [Release.cpp](#).

```
0013                                     {
0014     bool resultAll = true;
0015
0016     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
0017         resultAll &= _parentModel->checkExpression((*it)->getQuantityExpression(), "quantity", errorMessage);
0018         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Resource>(),
0019             (*it)->getResource(), "Resource", errorMessage);
0020         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Attribute>(),
0021             (*it)->getSaveAttribute(), "SaveAttribute", false, errorMessage);
0022     }
0023     return resultAll;
0024 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), [Model::checkExpression\(\)](#), [Model::getElements\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.101.3.2 _execute() void Release::_execute (Entity * entity) [protected], [virtual]

Implements [ModelComponent](#).

Definition at line 55 of file [Release.cpp](#).

```
00055                                     {
00056     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
00057         Resource* resource = (*it)->getResource();
00058         unsigned int quantity = _parentModel->parseExpression((*it)->getQuantityExpression());
00059         assert((*it)->getResource()->getNumberBusy() >= quantity);
00060         _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00061             entity, this, "Entity frees " + std::to_string(quantity) + " units of resource \""
00062             resource->getName() + "\" seized on time "
00063             std::to_string((*it)->getResource()->getLastTimeSeized()));
```

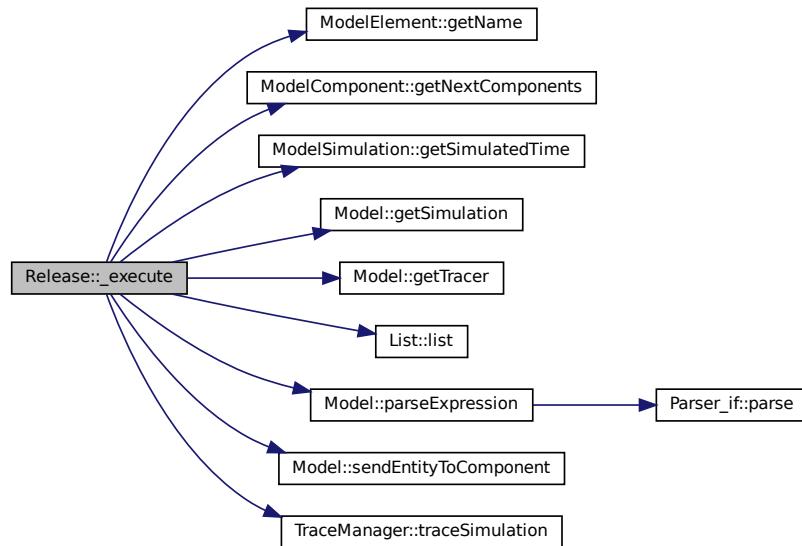
```

00061     (*it)->getResource()->release(quantity, _parentModel->getSimulation()->getSimulatedTime());
00062     //{releases and sets the 'LastTimeSeized'property}
00063     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00064 }

```

References [ModelElement::_parentModel](#), [ModelElement::getName\(\)](#), [ModelComponent::getNextComponents\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [List< T >::list\(\)](#), [Model::parseExpression\(\)](#), [Model::sendEntityToComponent\(\)](#), and [TraceManager::traceSimulation\(\)](#).

Here is the call graph for this function:



8.101.3.3 `_initBetweenReplications()`

```
void Release::_initBetweenReplications() [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 66 of file [Release.cpp](#).

```

00066 {
00067     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
00068         (*it)->getResource()->initBetweenReplications();
00069     }
00070 }

```

References [List< T >::list\(\)](#).

Here is the call graph for this function:



```
8.101.3.4 _loadInstance() bool Release::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

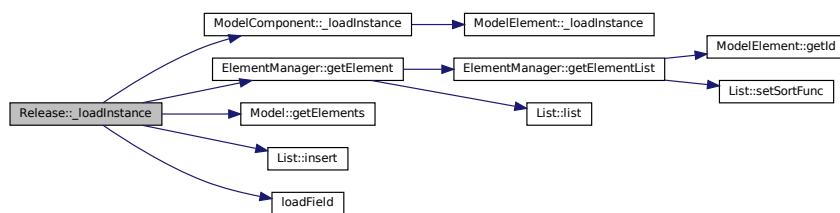
Definition at line 72 of file [Release.cpp](#).

```
00072
00073     bool res = ModelComponent::_loadInstance(fields);
00074     if (res) {
00075         this->_priority = std::stoi(loadField(fields, "priority", "0"));
00076         //Util::identification resourceId = std::stoi((*(fields->find("resourceId"))).second);
00077         /* Resource* res = dynamic_cast<Resource*>
00078             (_model->elements()->element(Util::TypeOf<Resource>(), resourceId));
00079
00080             unsigned short numRequests = std::stoi((*(fields->find("releaseResquestSize"))).second);
00081             for (unsigned short i = 0; i < numRequests; i++) {
00082                 /*std::string resRequest = ((*(fields->find("resourceItemRequest"))).second);
00083                 SeizableItemRequest::ResourceType resourceType =
00084                     static_cast<SeizableItemRequest::ResourceType>(std::stoi(loadField(fields, "resourceType" +
00085                         std::to_string(i), std::to_string(static_cast<int> (SeizableItemRequest::ResourceType::RESOURCE))));
```

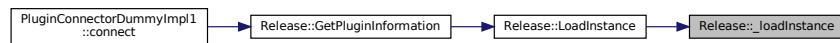
References [ModelComponent::_loadInstance\(\)](#), [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), [List< T >::insert\(\)](#), [SeizableItemRequest::LARGESTREMAININGCAPACITY](#), [loadField\(\)](#), and [SeizableItemRequest::RESOURCE](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.101.3.5 `_saveInstance()` `std::map< std::string, std::string > * Release::_saveInstance ()`
 [protected], [virtual]

Reimplemented from [ModelComponent](#).

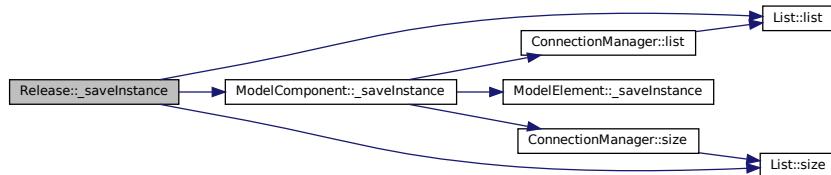
Definition at line 95 of file [Release.cpp](#).

```

00095
00096     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance ();
00097     //Util::TypeOf<Release>());
00098     if (_priority != 0) fields->emplace("priority", std::to_string(this->_priority));
00099     fields->emplace("releaseRequestSize", std::to_string(_releaseRequests->size()));
00100     unsigned short i = 0;
00101     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++, i++) {
00102         if ((*it)->getResourceType() != SeizableItemRequest::ResourceType::RESOURCE)
00103             fields->emplace("resourceType" + std::to_string(i), std::to_string(static_cast<int>((*it)->getResourceType())));
00104         //fields->emplace("resourceItemRequest" + std::to_string(i), "(" +
00105         map2str((*it)->_saveInstance()) + ")");
00106         fields->emplace("resourceId" + std::to_string(i),
00107                         std::to_string((*it)->getResource()->getId()));
00108         fields->emplace("resourceName" + std::to_string(i), ((*it)->getResource()->getName()));
00109         if ((*it)->getQuantityExpression() != "1") fields->emplace("quantity" + std::to_string(i),
00110                         (*it)->getQuantityExpression());
00111         if ((*it)->getSelectionRule() != SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)
00112             fields->emplace("selectionRule" + std::to_string(i), std::to_string(static_cast<int>((*it)->getSelectionRule())));
00113         if ((*it)->getSaveAttribute() != "") fields->emplace("saveAttribute" + std::to_string(i),
00114                         (*it)->getSaveAttribute());
00115         if ((*it)->getIndex() != 0) fields->emplace("index" + std::to_string(i),
00116                         std::to_string((*it)->getIndex()));
00117     }
00118     return fields;
00119 }
```

References [ModelComponent::_saveInstance\(\)](#), [SeizableItemRequest::LARGESTREMAININGCAPACITY](#), [List< T >::list\(\)](#), [SeizableItemRequest::RESOURCE](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.101.3.6 `GetPluginInformation()` `PluginInformation * Release::GetPluginInformation ()` [static]

Definition at line 137 of file [Release.cpp](#).

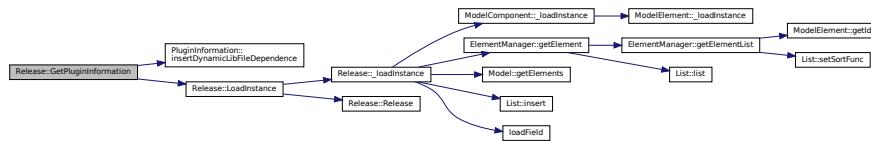
```

00137
00138     PluginInformation* info = new PluginInformation(Util::TypeOf<Release>(), &Release::LoadInstance);
00139     info->insertDynamicLibFileDependence("resource.so");
00140     return info;
00141 }
```

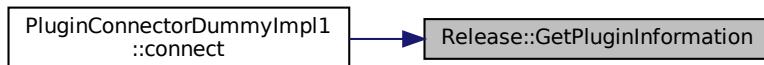
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), and [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

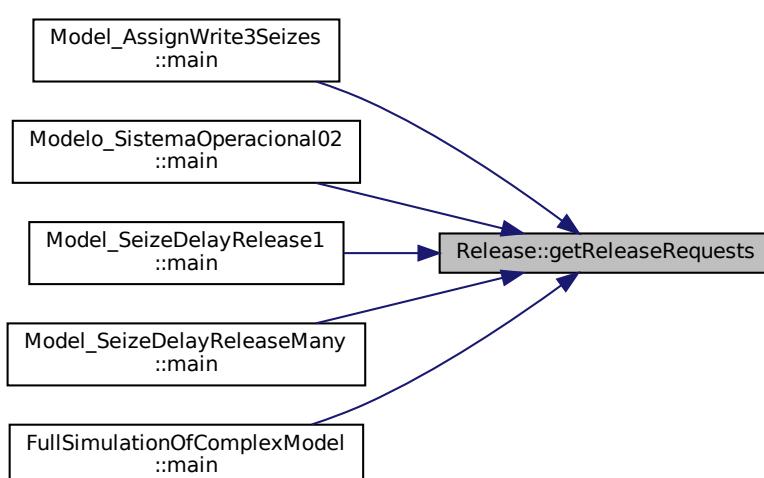


8.101.3.7 getReleaseRequests() List< SeizableItemRequest * > * Release::getReleaseRequests ()

Definition at line 43 of file [Release.cpp](#).

```
00043     return _releaseRequests;
00044 }
00045 }
```

Referenced by Model_AssignWrite3Seiz



```
8.101.3.8 LoadInstance() ModelComponent * Release::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

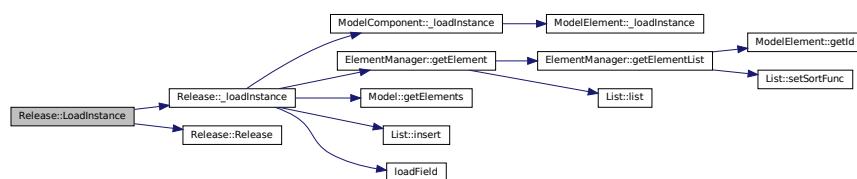
Definition at line 143 of file [Release.cpp](#).

```
00143
00144     Release* newComponent = new Release(model);
00145     try {
00146         newComponent->_loadInstance(fields);
00147     } catch (const std::exception& e) {
00148     }
00149 }
00150     return newComponent;
00151
00152 }
```

References [_loadInstance\(\)](#), and [Release\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.101.3.9 priority() unsigned short Release::priority () const
```

Definition at line 39 of file [Release.cpp](#).

```
00039
00040     return _priority;
00041 }
```

```
8.101.3.10 setPriority() void Release::setPriority (
    unsigned short _priority )
```

Definition at line 35 of file [Release.cpp](#).

```
00035
00036     this->_priority = _priority;
00037 }
```

8.101.3.11 `show()` `std::string Release::show() [virtual]`

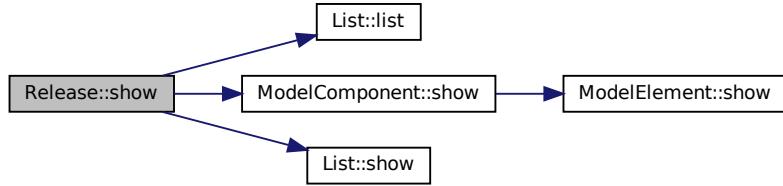
Reimplemented from [ModelComponent](#).

Definition at line 23 of file [Release.cpp](#).

```
00023     {
00024         std::string txt = ModelComponent::show\(\) +
00025             "priority=" + std::to_string(_priority) +
00026             "releaseRequests=[";
00027         unsigned short i = 0;
00028         for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list\(\)->begin(); it != _releaseRequests->list\(\)->end(); it++, i++) {
00029             txt += "request" + std::to_string(i) + "[" + _releaseRequests->show\(\) + "]";
00030         }
00031         txt = txt.substr(0, txt.length() - 1) + "}";
00032         return txt;
00033     }
```

References [List< T >::list\(\)](#), [ModelComponent::show\(\)](#), and [List< T >::show\(\)](#).

Here is the call graph for this function:



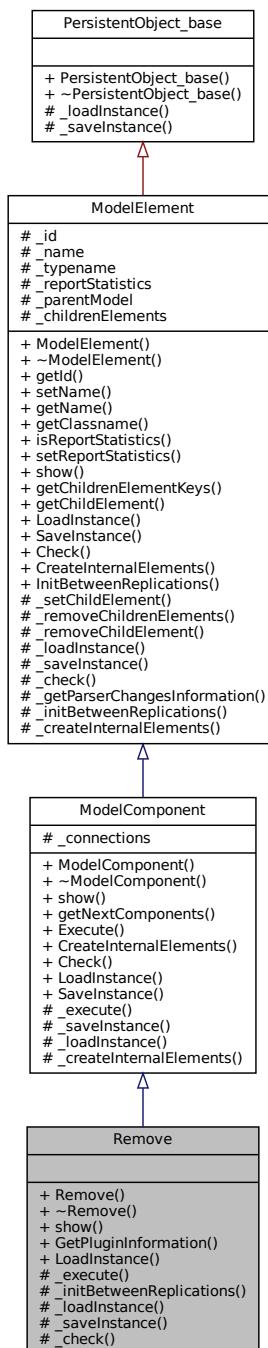
The documentation for this class was generated from the following files:

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

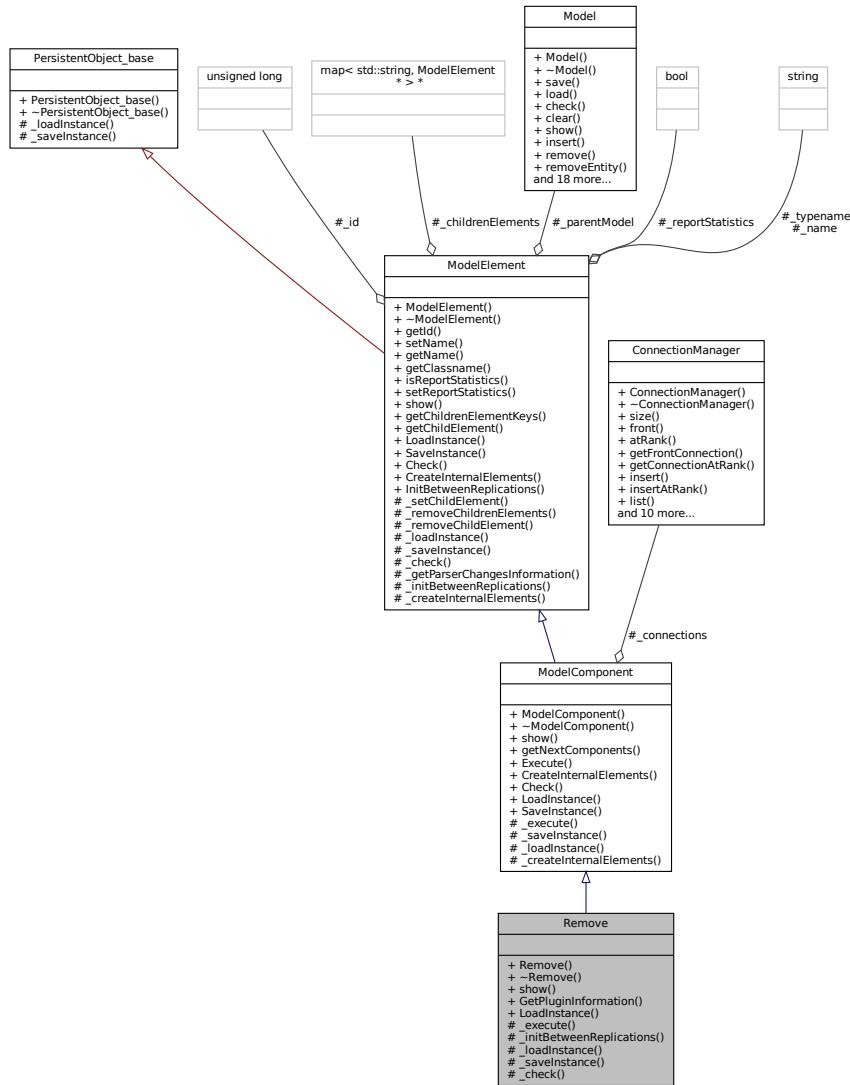
8.102 Remove Class Reference

```
#include <Remove.h>
```

Inheritance diagram for Remove:



Collaboration diagram for Remove:



Public Member Functions

- `Remove (Model *model, std::string name="")`
- virtual `~Remove ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.102.1 Detailed Description

Remove module DESCRIPTION The **Remove** module removes a single entity from a specified position in a queue and sends it to a designated module. When an entity arrives at a **Remove** module, it removes the entity from the specified queue and sends it to the connected module. The rank of the entity signifies the location of the entity within the queue. The entity that caused the removal proceeds to the next module specified and is processed before the removed entity. TYPICAL USES Removing an order from a queue that is due to be completed next Calling a patient from a waiting room for an examination Retrieving the next order to be processed from a pile of documents Prompt Description Name Unique module identifier displayed on the module shape. **Queue** Name Name of the queue from which the entity will be removed. **Entity** Rank of the entity to remove from within the queue.

Definition at line 37 of file [Remove.h](#).

8.102.2 Constructor & Destructor Documentation

8.102.2.1 Remove() Remove::Remove (

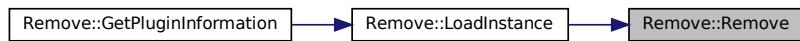
```
Model * model,
std::string name = "")
```

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

```
00017 : ModelComponent(model, Util::TypeOf<Remove>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.102.2.2 ~Remove() virtual Remove::~Remove () [virtual], [default]

8.102.3 Member Function Documentation

8.102.3.1 `_check()` `bool Remove::_check (std::string * errorMessage)` [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Remove.cpp](#).

```
00056
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
```

8.102.3.2 `_execute()` `void Remove::_execute (Entity * entity)` [protected], [virtual]

Implements [ModelComponent](#).

Definition at line 34 of file [Remove.cpp](#).

```
00034
00035     {
00036         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037         this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.102.3.3 `_initBetweenReplications()` `void Remove::_initBetweenReplications ()` [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Remove.cpp](#).

```
00047
00048 }
```

```
8.102.3.4 _loadInstance() bool Remove::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

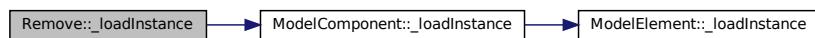
Definition at line 39 of file [Remove.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

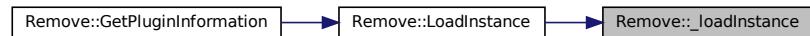
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.102.3.5 _saveInstance() std::map< std::string, std::string > * Remove::_saveInstance ( )
[protected], [virtual]
```

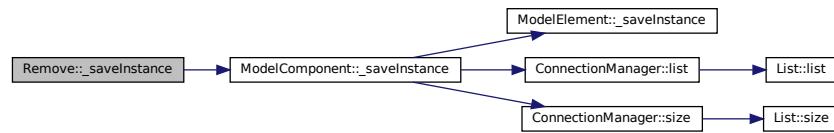
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [Remove.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



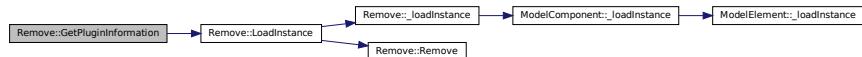
8.102.3.6 GetPluginInformation() `PluginInformation * Remove::GetPluginInformation () [static]`

Definition at line 62 of file [Remove.cpp](#).

```
00062                                         {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Remove>(), &Remove::LoadInstance);
00064     // ...
00065     return info;
00066 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.102.3.7 LoadInstance() `ModelComponent * Remove::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

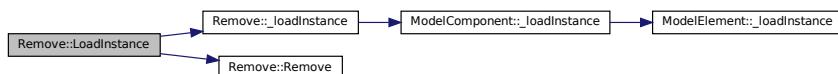
Definition at line 24 of file [Remove.cpp](#).

```
00024
00025     Remove* newComponent = new Remove(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
```

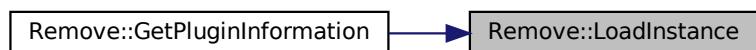
References [_loadInstance\(\)](#), and [Remove\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.102.3.8 show() std::string Remove::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Remove.cpp](#).

```
00020     {
00021     return ModelComponent::show() + "";
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



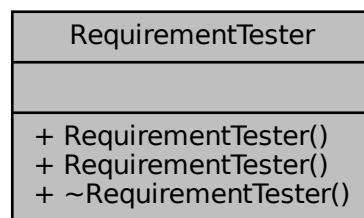
The documentation for this class was generated from the following files:

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

8.103 RequirementTester Class Reference

```
#include <RequirementTester.h>
```

Collaboration diagram for RequirementTester:

**Public Member Functions**

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

8.103.1 Detailed Description

Definition at line 17 of file [RequirementTester.h](#).

8.103.2 Constructor & Destructor Documentation

8.103.2.1 RequirementTester() [1/2] `RequirementTester::RequirementTester ()`

Definition at line 16 of file [RequirementTester.cpp](#).

```
00016 {  
00017 }
```

8.103.2.2 RequirementTester() [2/2] `RequirementTester::RequirementTester (const RequirementTester & orig)`

Definition at line 19 of file [RequirementTester.cpp](#).

```
00019 {  
00020 }
```

8.103.2.3 ~RequirementTester() `RequirementTester::~RequirementTester () [virtual]`

Definition at line 22 of file [RequirementTester.cpp](#).

```
00022 {  
00023 }
```

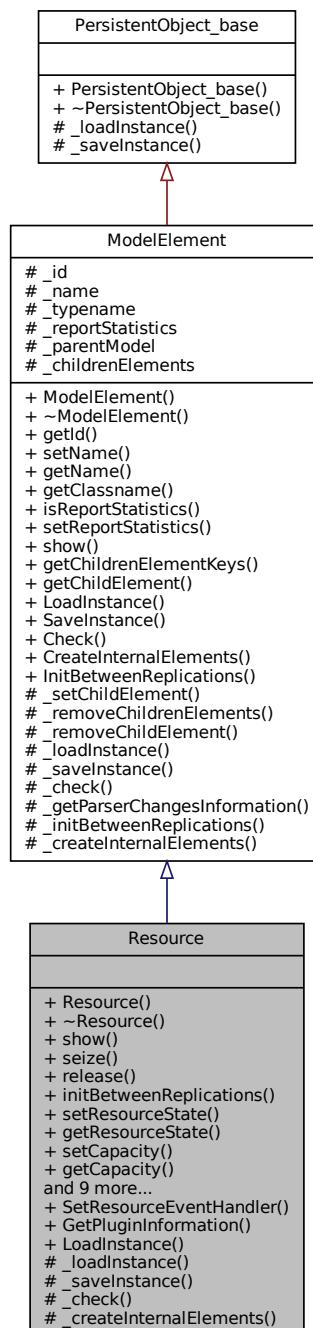
The documentation for this class was generated from the following files:

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

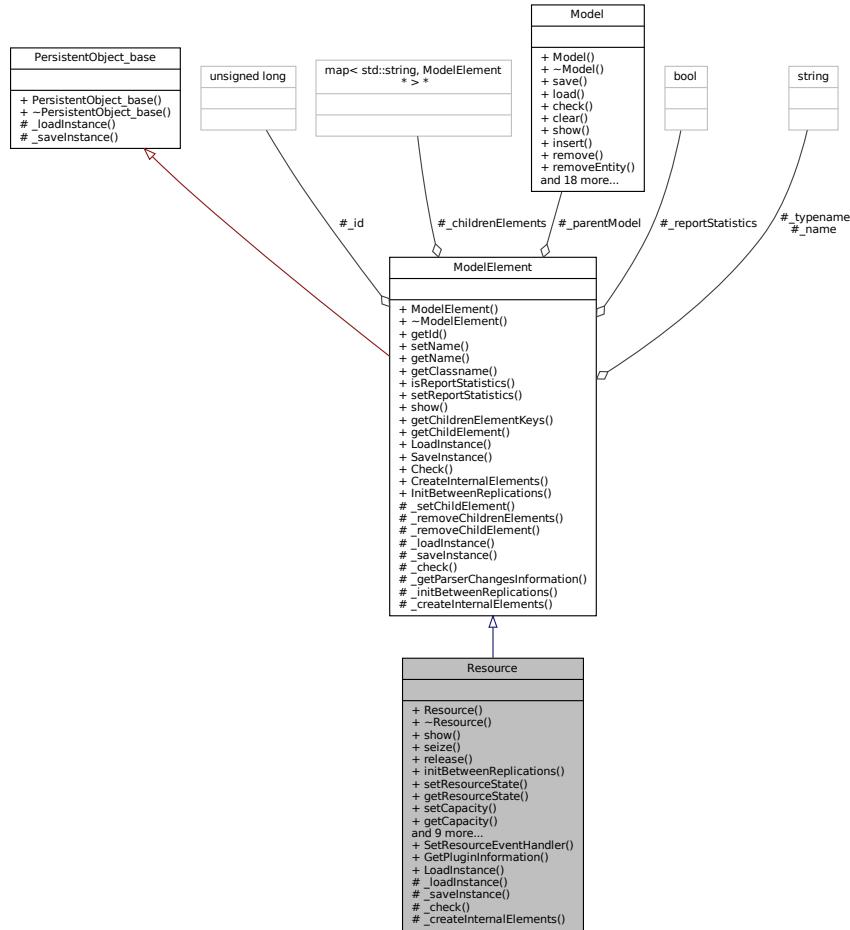
8.104 Resource Class Reference

```
#include <Resource.h>
```

Inheritance diagram for Resource:



Collaboration diagram for Resource:



Public Types

- enum ResourceState : int {
 ResourceState::IDLE = 1, ResourceState::BUSY = 2, ResourceState::FAILED = 3, ResourceState::INACTIVE
 = 4,
 ResourceState::OTHER = 5 }
 - typedef std::function< void(Resource *) > ResourceEventHandler

Public Member Functions

- **Resource** (`Model *model, std::string name=""`)
 - virtual `~Resource ()`
 - virtual `std::string show ()`
 - void `seize (unsigned int quantity, double tnow)`
 - void `release (unsigned int quantity, double tnow)`
 - void `initBetweenReplications ()`
 - 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`
- void `addReleaseResourceEventHandler (ResourceEventHandler eventHandler)`
- double `getLastTimeSeized () const`

Static Public Member Functions

- template<typename Class >
static `ResourceEventHandler SetResourceEventHandler (void(Class::*function)(Resource *), Class *object)`
- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Additional Inherited Members

8.104.1 Detailed Description

Resource module DESCRIPTION This data module defines the resources in the simulation system, including costing information and resource availability. Resources may have a fixed capacity that does not vary over the simulation run or may operate based on a schedule. **Resource** failures and states can also be specified in this module. TYPICAL USES Equipment (machinery, cash register, phone line) People (clerical, order processing, sales clerks, operators) PROMPTS Prompt Description Name The name of the resource whose characteristics are being defined. This name must be unique. Type Method for determining the capacity for a resource. Fixed Capacity will not change during the simulation run. Based on **Schedule** signifies that a **Schedule** module is used to specify the capacity and duration information for the resource. Capacity Number of resource units of a given name that are available to the system for processing. Applies only when Type is Fixed Capacity. **Schedule** Name Identifies the name of the schedule to be used by the resource. The schedule defines the capacity of a resource for a given period of time. Applies only when type is **Schedule**. **Schedule** Rule Dictates when the actual capacity change is to occur when a decrease in capacity is required for a busy resource unit. Applies only when Type is **Schedule**. Busy/Hour Cost per hour of a resource that is processing an entity. The resource becomes busy when it is originally allocated to an entity and becomes idle when it is released. During the time when it is busy, cost will accumulate based on the busy/hour cost. The busy cost per hour is automatically converted to the appropriate base time unit specified within the Replication Parameters page of the Run > Setup menu item. Idle/Hour Cost per hour of a resource that is idle. The resource is idle while it is not processing an entity. During the time when it is idle, cost will accumulate based on the idle/hour cost. The idle cost per hour is automatically converted to the appropriate base time unit specified within the Replication Parameters page of the Run > Setup menu item. Per Use Cost of a resource on

a usage basis, regardless of the time for which it is used. Each time the resource is allocated to an entity, it will incur a per-use cost. StateSet Name Name of states that the resource may be assigned during the simulation run. Initial State Initial state of a resource. If specified, the name must be defined within the repeat group of state names. This field is shown only when a StateSet Name is defined. Failures Lists all failures that will be associated with the resource. Failure Name—Name of the failure associated with the resource. Failure Rule—Behavior that should occur when a failure is to occur for a busy resource unit. Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this resource.

Definition at line 81 of file [Resource.h](#).

8.104.2 Member Typedef Documentation

8.104.2.1 ResourceEventHandler `typedef std::function<void(Resource*)> Resource::ResourceEventHandler`

Definition at line 83 of file [Resource.h](#).

8.104.3 Member Enumeration Documentation

8.104.3.1 ResourceState `enum Resource::ResourceState : int [strong]`

Enumerator

IDLE	
BUSY	
FAILED	
INACTIVE	
OTHER	

Definition at line 90 of file [Resource.h](#).

```
00090           : int {
00091     IDLE = 1, BUSY = 2, FAILED = 3, INACTIVE = 4, OTHER = 5
00092   };
```

8.104.4 Constructor & Destructor Documentation

8.104.4.1 Resource() `Resource::Resource (` `Model * model,` `std::string name = "")`

Definition at line 18 of file [Resource.cpp](#).

```
00018           : ModelElement(model, Util::TypeOf<Resource>(), name) {
00019     GetterMember getter = DefineGetterMember<Resource>(this, &Resource::getCapacity);
00020     SetterMember setter = DefineSetterMember<Resource>(this, &Resource::setCapacity);
```

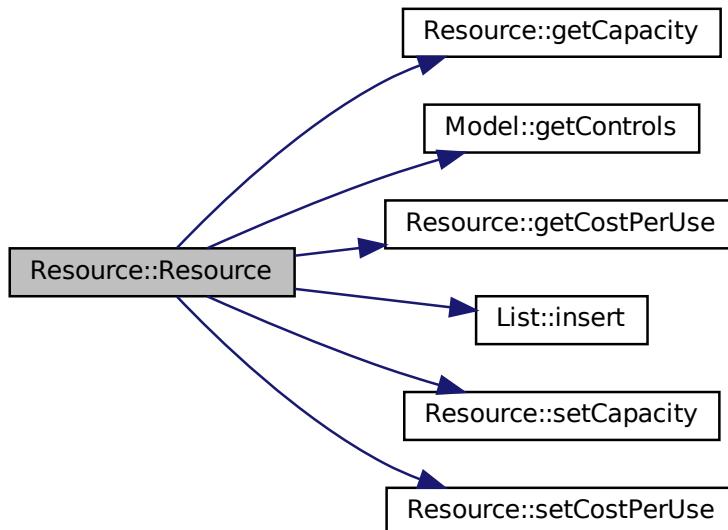
```

00021     model->getControls()->insert(new SimulationControl(Util::TypeOf<Resource>(), _name + ".Capacity",
00022                                         getter, setter));
00023     GetterMember getter2 = DefineGetterMember<Resource>(this, &Resource::getCostPerUse);
00024     SetterMember setter2 = DefineSetterMember<Resource>(this, &Resource::setCostPerUse);
00025     model->getControls()->insert(new SimulationControl(Util::TypeOf<Resource>(), _name +
00026                                         ".CostPerUse", getter2, setter2));
00027 // ...
00028 }
```

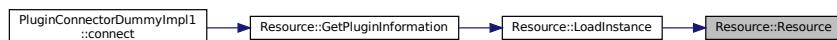
References [ModelElement::_name](#), [getCapacity\(\)](#), [Model::getControls\(\)](#), [getCostPerUse\(\)](#), [List< T >::insert\(\)](#), [setCapacity\(\)](#), and [setCostPerUse\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.104.4.2 ~Resource() Resource::~Resource () [virtual]

Definition at line 29 of file [Resource.cpp](#).

```

00029 {
00030 }
```

8.104.5 Member Function Documentation

8.104.5.1 `_check()` `bool Resource::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 172 of file [Resource.cpp](#).

```
00172
00173     return true;
00174 }
```

8.104.5.2 `_createInternalElements()` `void Resource::_createInternalElements () [protected], [virtual]`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

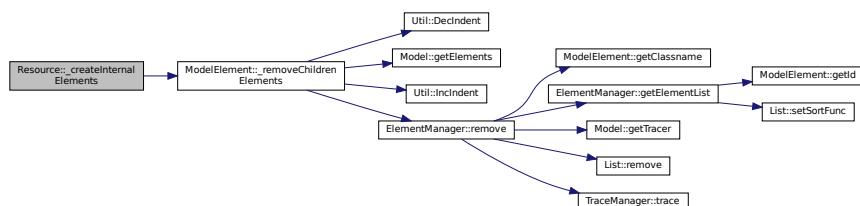
Reimplemented from [ModelElement](#).

Definition at line 176 of file [Resource.cpp](#).

```
00176
00177     if (_reportStatistics && _cstatTimeSeized == nullptr) {
00178         _cstatTimeSeized = new StatisticsCollector(_parentModel, _name + "." + "TimeSeized", this);
00179         _numSeizes = new Counter(_parentModel, _name + "." + "Seizes", this);
00180         _numReleases = new Counter(_parentModel, _name + "." + "Releases", this);
00181         _childrenElements->insert({"TimeSeized", _cstatTimeSeized});
00182         _childrenElements->insert({"Seizes", _numSeizes});
00183         _childrenElements->insert({"Releases", _numReleases});
00184     } else if (!_reportStatistics && _cstatTimeSeized != nullptr) {
00185         _removeChildrenElements();
00186     }
00187 }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



```
8.104.5.3 _loadInstance() bool Resource::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

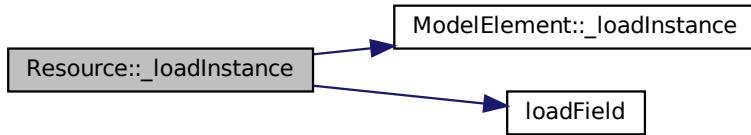
Definition at line 151 of file [Resource.cpp](#).

```
00151
00152     bool res = ModelElement::_loadInstance(fields);
00153     if (res) {
00154         //this->_capacity = (fields->find("capacity") != fields->end() ?
00155             std::stoi((*(fields->find("capacity"))).second) : 1);
00156         this->_capacity = std::stoi(loadField(fields, "capacity", "1"));
00157         this->_costBusyHour = std::stod(loadField(fields, "costBusyHour", "1.0"));
00158         //(*(fields->find("costBusyHour"))).second;
00159         this->_costIdleHour = std::stod(loadField(fields, "costIdleHour", "1.0"));
00160         //(*(fields->find("costIdleHour"))).second;
00161         this->_costPerUse = std::stod(loadField(fields, "costPerUse", "1.0"));
00162         //(*(fields->find("costPerUse"))).second;
00163     }
00164     return res;
00165 }
```

References [ModelElement::_loadInstance\(\)](#), and [loadField\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.104.5.4 _saveInstance() std::map< std::string, std::string > * Resource::_saveInstance ( )
[protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 163 of file [Resource.cpp](#).

```
00163
00164     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00165     //Util::TypeOf<Resource>());
00166     if (_capacity != 1) fields->emplace("capacity", std::to_string(this->_capacity));
00167     if (_costBusyHour != 1.0) fields->emplace("costBusyHour", std::to_string(this->_costBusyHour));
00168     if (_costIdleHour != 1.0) fields->emplace("costIdleHour", std::to_string(this->_costIdleHour));
00169     if (_costPerUse != 1.0) fields->emplace("costPerUse", std::to_string(this->_costPerUse));
```

```
00169     return fields;
00170 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.104.5.5 addReleaseResourceEventHandler()

```
void Resource::addReleaseResourceEventHandler (
    ResourceEventHandler eventHandler )
```

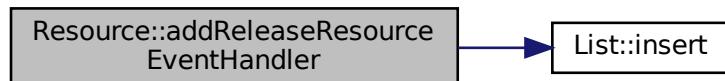
Definition at line 122 of file [Resource.cpp](#).

```
00122
00123     this->_resourceEventHandlers->insert(eventHandler); // \todo: priority should be registered as
                           well, so handlers are invoked ordered by priority
00124 }
```

References [List< T >::insert\(\)](#).

Referenced by [Seize::_loadInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.104.5.6 getCapacity() `unsigned int Resource::getCapacity () const`Definition at line 90 of file [Resource.cpp](#).

```
00090
00091     return _capacity;
00092 }
```

Referenced by [Seize::_execute\(\)](#), and [Resource\(\)](#).

Here is the caller graph for this function:

**8.104.5.7 getCostBusyHour()** `double Resource::getCostBusyHour () const`Definition at line 98 of file [Resource.cpp](#).

```
00098
00099     return _costBusyHour;
00100 }
```

8.104.5.8 getCostIdleHour() `double Resource::getCostIdleHour () const`Definition at line 106 of file [Resource.cpp](#).

```
00106
00107     return _costIdleHour;
00108 }
```

8.104.5.9 getCostPerUse() `double Resource::getCostPerUse () const`Definition at line 114 of file [Resource.cpp](#).

```
00114
00115     return _costPerUse;
00116 }
```

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

Here is the caller graph for this function:



8.104.5.10 `getLastTimeSeized()` double Resource::getLastTimeSeized () const

Definition at line 126 of file [Resource.cpp](#).

```
00126                               {
00127     return _lastTimeSeized;
00128 }
```

8.104.5.11 `getNumberBusy()` unsigned int Resource::getNumberBusy () const

Definition at line 118 of file [Resource.cpp](#).

```
00118                               {
00119     return _numberBusy;
00120 }
```

Referenced by [Seize::_execute\(\)](#).

Here is the caller graph for this function:



8.104.5.12 `GetPluginInformation()` PluginInformation * Resource::GetPluginInformation () [static]

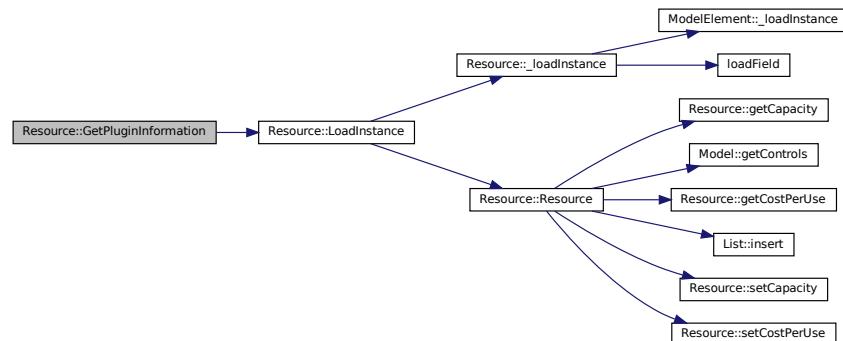
Definition at line 136 of file [Resource.cpp](#).

```
00136                               {
00137     PluginInformation* info = new PluginInformation(Util::TypeOf<Resource>(),
00138         &Resource::LoadInstance);
00139     return info;
00140 }
```

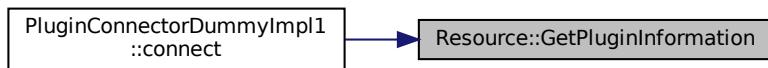
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.104.5.13 getResourceState() `Resource::ResourceState` `Resource::getResourceState() const`

Definition at line 82 of file [Resource.cpp](#).

```
00082
00083     return _resourceState;
00084 }
```

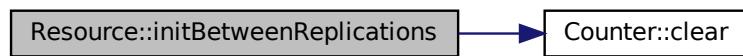
8.104.5.14 initBetweenReplications() `void Resource::initBetweenReplications()`

Definition at line 69 of file [Resource.cpp](#).

```
00069
00070     this->_lastTimeSeized = 0.0;
00071     this->_numberBusy = 0;
00072     if (_reportStatistics) {
00073         this->_numSeizes->clear();
00074         this->_numReleases->clear();
00075     }
00076 }
```

References [ModelElement::_reportStatistics](#), and [Counter::clear\(\)](#).

Here is the call graph for this function:



8.104.5.15 LoadInstance() `ModelElement * Resource::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

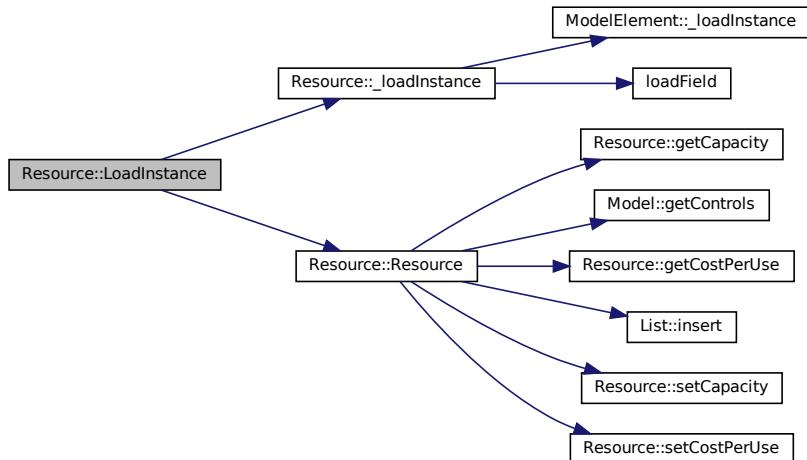
Definition at line 141 of file [Resource.cpp](#).

```
00141
00142     Resource* newElement = new Resource(model);
00143     try {
00144         newElement->_loadInstance(fields);
00145     } catch (const std::exception& e) {
00146     }
00147 }
00148     return newElement;
00149 }
```

References [_loadInstance\(\)](#), and [Resource\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.104.5.16 release() `void Resource::release (unsigned int quantity, double tnow)`

Definition at line 50 of file [Resource.cpp](#).

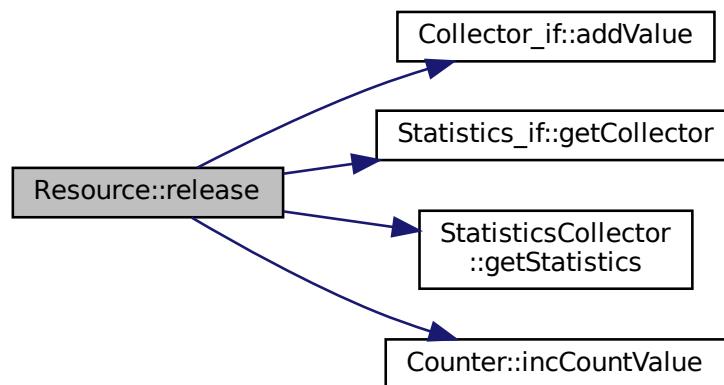
```
00050 {
```

```

00051     if (_numberBusy >= quantity) {
00052         _numberBusy -= quantity;
00053     } else {
00054         _numberBusy = 0;
00055     }
00056     if (_numberBusy == 0) {
00057         _resourceState = Resource::ResourceState::IDLE;
00058     }
00059     double timeSeized = tnow - _lastTimeSeized;
00060     if (_reportStatistics) {
00061         _numReleases->incCountValue(quantity);
00062         _cstatTimeSeized->getStatistics()->getCollector()->addValue(timeSeized);
00063     }
00064     /**
00065     _lastTimeSeized = timeSeized;
00066     _notifyReleaseEventHandlers();
00067 }
```

References [ModelElement::_reportStatistics](#), [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [IDLE](#), and [Counter::incCountValue\(\)](#).

Here is the call graph for this function:



8.104.5.17 seize() void Resource::seize (unsigned int quantity, double tnow)

Definition at line 41 of file [Resource.cpp](#).

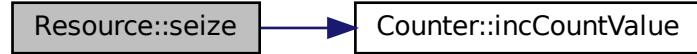
```

00041     _numberBusy += quantity;
00042     if (_reportStatistics)
00043         _numSeizes->incCountValue(quantity);
00044     _lastTimeSeized = tnow;
00045     _resourceState = Resource::ResourceState::BUSY;
00046     // \todo implement costs
00047
00048 }
```

References [ModelElement::_reportStatistics](#), [BUSY](#), and [Counter::incCountValue\(\)](#).

Referenced by [Seize::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.104.5.18 setCapacity() `void Resource::setCapacity (unsigned int _capacity)`

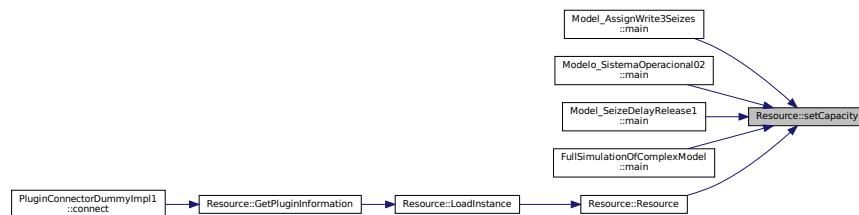
Definition at line 86 of file [Resource.cpp](#).

```

00086
00087     this->_capacity = _capacity;
00088 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [FullSimulationOfComplexModel::main\(\)](#), and [Resource\(\)](#).

Here is the caller graph for this function:



8.104.5.19 setCostBusyHour() void Resource::setCostBusyHour (double _costBusyHour)

Definition at line 94 of file [Resource.cpp](#).

```
00094
00095     this->_costBusyHour = _costBusyHour;
00096 }
```

8.104.5.20 setCostIdleHour() void Resource::setCostIdleHour (double _costIdleHour)

Definition at line 102 of file [Resource.cpp](#).

```
00102
00103     this->_costIdleHour = _costIdleHour;
00104 }
```

8.104.5.21 setCostPerUse() void Resource::setCostPerUse (double _costPerUse)

Definition at line 110 of file [Resource.cpp](#).

```
00110
00111     this->_costPerUse = _costPerUse;
00112 }
```

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

Here is the caller graph for this function:



8.104.5.22 SetResourceEventHandler() template<typename Class > static [ResourceEventHandler](#) Resource::SetResourceEventHandler (void(Class::*)([Resource](#) *) function, Class * object) [inline], [static]

Definition at line 86 of file [Resource.h](#).

```
00086
00087     {
00088         return std::bind(function, object, std::placeholders::_1);
00089     }
```

8.104.5.23 setResourceState() void Resource::setResourceState ([ResourceState](#) _resourceState)

Definition at line 78 of file [Resource.cpp](#).

```
00078
00079     this->_resourceState = _resourceState;
00080 }
```

8.104.5.24 `show()` `std::string Resource::show() [virtual]`

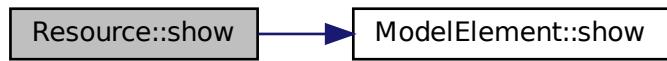
Reimplemented from [ModelElement](#).

Definition at line 32 of file [Resource.cpp](#).

```
00032     {
00033     return ModelElement::show() +
00034         ",capacity=" + std::to_string(_capacity) +
00035         ",costBusyByour=" + std::to_string(_costBusyHour) +
00036         ",costIdleByour=" + std::to_string(_costIdleHour) +
00037         ",costPerUse=" + std::to_string(_costPerUse) +
00038         ",state=" + std::to_string(static_cast<int> (_resourceState));
00039 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:

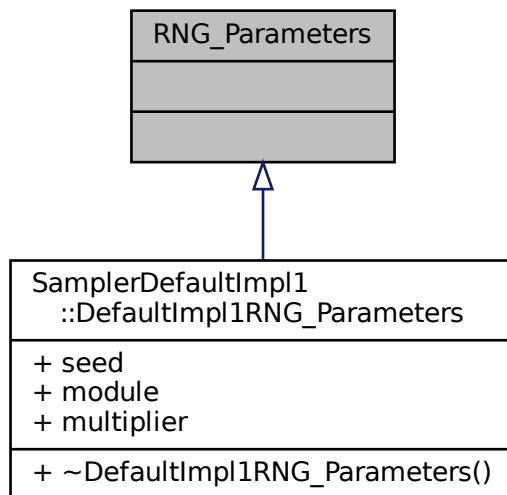


The documentation for this class was generated from the following files:

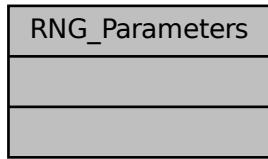
- [Resource.h](#)
- [Resource.cpp](#)

8.105 RNG_Parameters Class Reference

Inheritance diagram for RNG_Parameters:



Collaboration diagram for RNG_Parameters:



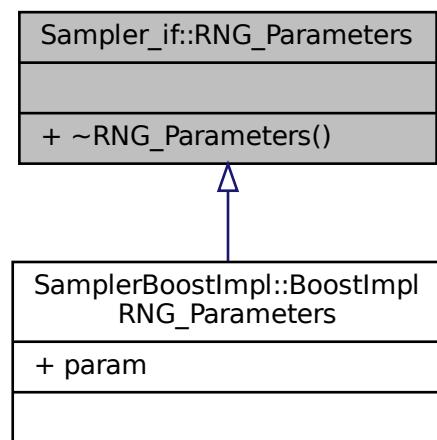
The documentation for this class was generated from the following file:

- [SamplerDefaultImpl1.h](#)

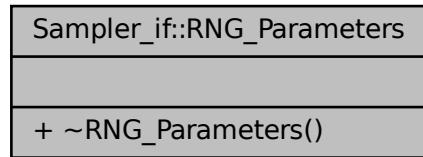
8.106 Sampler_if::RNG_Parameters Struct Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for Sampler_if::RNG_Parameters:



Collaboration diagram for Sampler_if::RNG_Parameters:



Public Member Functions

- virtual [~RNG_Parameters](#) ()=default

8.106.1 Detailed Description

class that encapsulates attributes required to generate random numbers, which depends on the generation method used.

Definition at line [26](#) of file [Sampler_if.h](#).

8.106.2 Constructor & Destructor Documentation

8.106.2.1 ~RNG_Parameters() virtual Sampler_if::RNG_Parameters::~RNG_Parameters () [virtual],
[default]

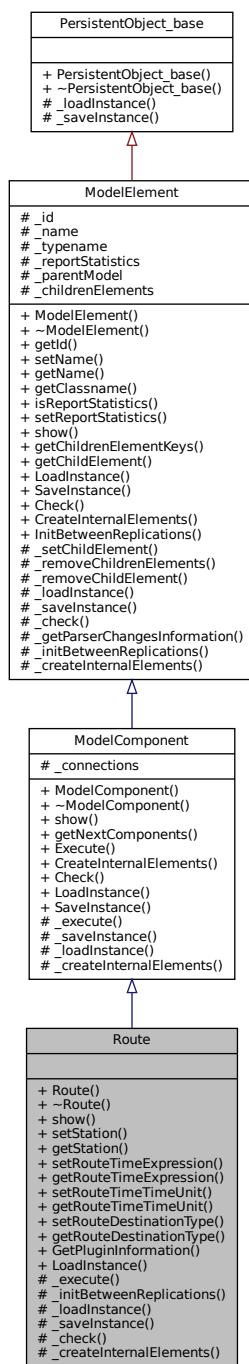
The documentation for this struct was generated from the following file:

- [Sampler_if.h](#)

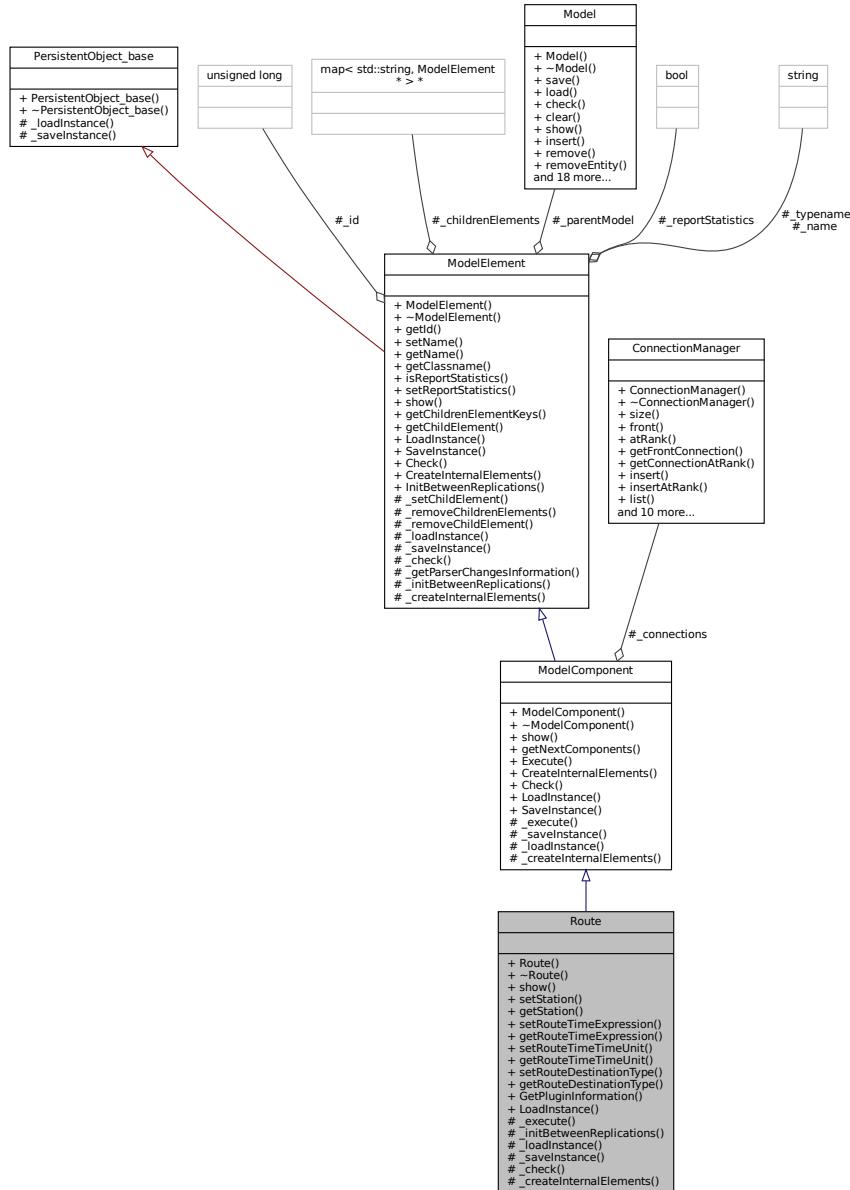
8.107 Route Class Reference

```
#include <Route.h>
```

Inheritance diagram for Route:



Collaboration diagram for Route:



Public Types

- enum `DestinationType` : int { `DestinationType::Station` = 0, `DestinationType::BySequence` = 1 }

Public Member Functions

- `Route (Model *model, std::string name="")`
- `virtual ~Route ()=default`
- `virtual std::string show ()`
- `void setStation (Station *_station)`
- `Station * getStation () const`

- void `setRouteTimeExpression` (std::string _routeTimeExpression)
- std::string `getRouteTimeExpression` () const
- void `setRouteTimeTimeUnit` (Util::TimeUnit _routeTimeTimeUnit)
- Util::TimeUnit `getRouteTimeTimeUnit` () const
- void `setRouteDestinationType` (DestinationType _routeDestinationType)
- DestinationType `getRouteDestinationType` () const

Static Public Member Functions

- static PluginInformation * `GetPluginInformation` ()
- static ModelComponent * `LoadInstance` (Model *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void `_execute` (Entity *entity)
- virtual void `_initBetweenReplications` ()
- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)
- virtual void `_createInternalElements` ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

8.107.1 Detailed Description

Route module DESCRIPTION The **Route** module transfers an entity to a specified station or the next station in the station visitation sequence defined for the entity. A delay time to transfer to the next station may be defined. When an entity enters the **Route** module, its **Station** attribute (Entity.Station) is set to the destination station. The entity is then sent to the destination station, using the route time specified. If the station destination is entered as By **Sequence**, the next station is determined by the entity's sequence and step within the set (defined by special-purpose attributes Entity.Sequence and Entity.Jobstep, respectively). TYPICAL USES Send a part to its next processing station based on its routing slip Send an account balance call to an account agent Send restaurant customers to a specific table PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. **Route** Time Travel time from the entity's current location to the destination station. Units Time units for route-time parameters. Destination Type Method for determining the entity destination location. Selection of By **Sequence** requires that the entity has been assigned a sequence name and that the sequence itself has been defined. **Station** Name Name of the individual destination station. **Attribute** Name Name of the attribute that stores the station name to which entities will route. Expression Expression that is evaluated to the station name where entities will route.

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

8.107.2 Member Enumeration Documentation

8.107.2.1 DestinationType enum `Route::DestinationType` : int [strong]

Enumerator

Station	
BySequence	

Definition at line 56 of file [Route.h](#).

```
00056     : int {
00057         Station = 0, BySequence = 1
00058     };
```

8.107.3 Constructor & Destructor Documentation

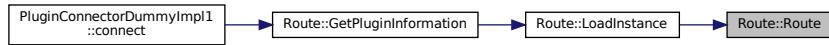
8.107.3.1 Route() `Route::Route (`
 `Model * model,`
 `std::string name = "")`

Definition at line 21 of file [Route.cpp](#).

```
00021 : ModelComponent(model, Util::TypeOf<Route>(), name) {
00022 }
```

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

Here is the caller graph for this function:



8.107.3.2 ~Route() `virtual Route::~Route () [virtual], [default]`

8.107.4 Member Function Documentation

```
8.107.4.1 _check() bool Route::_check (
    std::string * errorMessage ) [protected], [virtual]
```

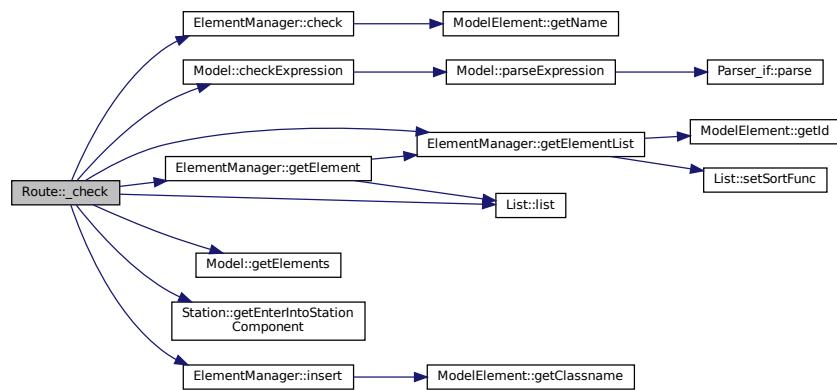
Reimplemented from [ModelElement](#).

Definition at line 139 of file [Route.cpp](#).

```
00139
00140     //include attributes needed
00141     ElementManager* elements = _parentModel->getElements();
00142     std::vector<std::string> neededNames = {"Entity.TotalTransferTime", "Entity.Station"};
00143     std::string neededName;
00144     for (unsigned int i = 0; i < neededNames.size(); i++) {
00145         neededName = neededNames[i];
00146         if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00147             Attribute* attr = new Attribute(_parentModel, neededName);
00148             elements->insert(attr);
00149         }
00150     }
00151     // include StatisticsCollector needed in EntityType
00152     std::list<ModelElement*>* enttypes = elements->getElementTypeList(Util::TypeOf<EntityType>()->list());
00153     for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00154         if ((*it)->isReportStatistics())
00155             static_cast<EntityType*> ((*it))->addGetStatisticsCollector((*it)->getName() +
00156 ".TransferTime"); // force create this CStat before simulation starts
00157     }
00158     bool resultAll = true;
00159     resultAll &= _parentModel->checkExpression(_routeTimeExpression, "Route time expression",
00160     errorMessage);
00161     if (this->_routeDestinationType == Route::DestinationType::Station) {
00162         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",
00163     errorMessage);
00164         if (resultAll) {
00165             resultAll &= _station->getEnterIntoStationComponent() != nullptr;
00166             if (!resultAll) {
00167                 errorMessage->append("Station has no component to enter into it");
00168             }
00169         }
00170     }
00171     //__model->getParent ()->getPluginManager ()-
00172     return resultAll;
00173 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), [Model::checkExpression\(\)](#), [ElementManager::getElement\(\)](#), [ElementManager::getElementTypeList\(\)](#), [Model::getElements\(\)](#), [Station::getEnterIntoStationComponent\(\)](#), [ElementManager::insert\(\)](#), [List< T >::list\(\)](#), and [Station](#).

Here is the call graph for this function:



8.107.4.2 `_createInternalElements()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

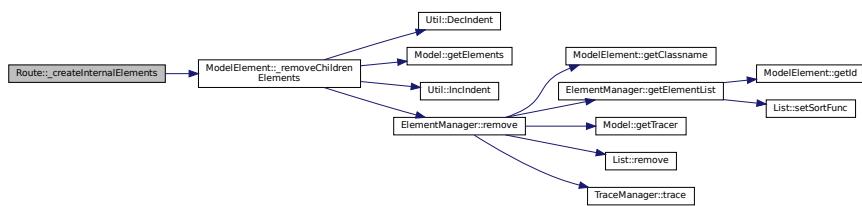
Reimplemented from [ModelComponent](#).

Definition at line 179 of file [Route.cpp](#).

```
00179         if (_reportStatistics) {
00180             if (_numberIn == nullptr) {
00181                 _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00182                 _childrenElements->insert({"CountNumberIn", _numberIn});
00183             }
00184         } else
00185             if (_numberIn != nullptr) {
00186                 _removeChildrenElements();
00187             }
00188     }
00189 }
```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#) and [ModelElement::_reportStatistics](#).

Here is the call graph for this function:



8.107.4.3 `_execute()`

`Entity * entity)` [protected], [virtual]

Implements [ModelComponent](#).

Definition at line 75 of file [Route.cpp](#).

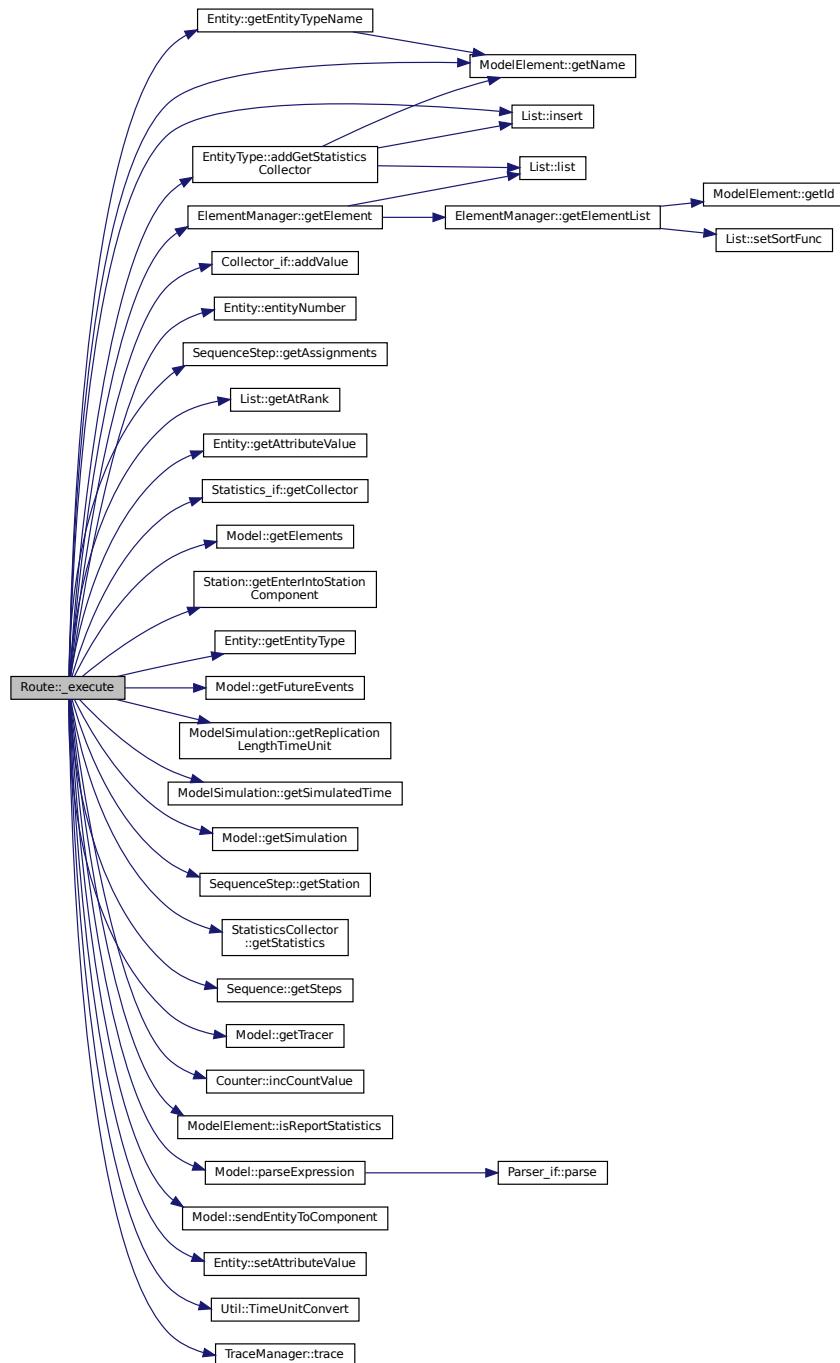
```
00075     Station* destinyStation = _station;
00076     if (_routeDestinationType == Route::DestinationType::BySequence) {
00077         unsigned int sequenceId = static_cast<unsigned int>(
00078             entity->getAttributeValue("Entity.Sequence"));
00079         unsigned int step = static_cast<unsigned int>
00080             (entity->getAttributeValue("Entity.SequenceStep"));
00081         Sequence* sequence = static_cast<Sequence*>
00082             (_parentModel->getElements())->getElement(Util::TypeOf<Sequence>(),
00083                 sequenceId);
00083         SequenceStep* seqStep = sequence->getSteps()->getAtRank(step);
00084         if (seqStep == nullptr) {
00085             step = 0;
00086             seqStep = sequence->getSteps()->getAtRank(step);
00087             assert(seqStep != nullptr);
00088         }
00089         destinyStation = seqStep->getStation();
00090         for (std::list<std::string>::iterator it = seqStep->getAssignments()->begin(); it !=
00091             seqStep->getAssignments()->end(); it++) {
00092             _parentModel->parseExpression((*it));
00093         }
00094         entity->setAttributeValue("Entity.SequenceStep", step + 1.0);
00095     }
```

```

00093     if (_reportStatistics)
00094         _numberIn->incCountValue();
00095     // adds the route time to the TransferTime statistics / attribute related to the Entity
00096     double routeTime = _parentModel->parseExpression(_routeTimeExpression) *
00097         Util::TimeUnitConvert(_routeTimeTimeUnit,
00098         _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00099     if (entity->getEntityType()->isReportStatistics())
00100         entity->getEntityType()->addGetStatisticsCollector(entity->getEntityTypeName() +
00101             ".TransferTime")->getStatistics()->getCollector()->addValue(routeTime);
00102     entity->setAttributeValue("Entity.TotalTransferTime",
00103         entity->getAttributeValue("Entity.TotalTransferTime") + routeTime);
00104     if (routeTime > 0.0) {
00105         // calculates when this Entity will reach the end of this route and schedule this Event
00106         double routeEndTime = _parentModel->getSimulation()->getSimulatedTime() + routeTime;
00107         Event* newEvent = new Event(routeEndTime, entity,
00108             destinyStation->getEnterIntoStationComponent());
00109         _parentModel->getFutureEvents()->insert(newEvent);
00110         _parentModel->getTracer()->trace("End of route of entity " +
00111             std::to_string(entity->entityNumber()) + " to the component \"\"
00112             destinyStation->getEnterIntoStationComponent()->getName() + "\" was scheduled to time " +
00113             std::to_string(routeEndTime));
00114     } else {
00115         // send without delay
00116         _parentModel->sendEntityToComponent(entity, destinyStation->getEnterIntoStationComponent(),
00117             0.0);
00118     }
00119 }
```

References ModelElement::_parentModel, ModelElement::_reportStatistics, EntityType::addGetStatisticsCollector(), Collector_if::addValue(), BySequence, Entity::entityNumber(), SequenceStep::getAssignments(), List< T >::getAtRank(), Entity::getAttributeValue(), Statistics_if::getCollector(), ElementManager::getElement(), Model::getElements(), Station::getEnterIntoStationComponent(), Entity::getEntityType(), Entity::getEntityTypeName(), Model::getFutureEvents(), ModelElement::getName(), ModelSimulation::getReplicationLengthTimeUnit(), ModelSimulation::getSimulatedTime(), Model::getSimulation(), SequenceStep::getStation(), StatisticsCollector::getStatistics(), Sequence::getSteps(), Model::getTracer(), Counter::incCountValue(), List< T >::insert(), ModelElement::isReportStatistics(), Model::parseExpression(), Model::sendEntityToComponent(), Entity::setAttributeValue(), Util::TimeUnitConvert(), and TraceManager::trace().

Here is the call graph for this function:



8.107.4.4 `_initBetweenReplications()` void `Route::_initBetweenReplications() [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 125 of file [Route.cpp](#).

```
00125
00126     _numberIn->clear();
00127 }
```

References [Counter::clear\(\)](#).

Here is the call graph for this function:



8.107.4.5 [_loadInstance\(\)](#)

```
bool Route::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

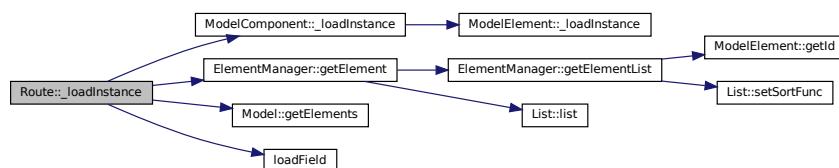
Definition at line 112 of file [Route.cpp](#).

```
00112
00113     bool res = ModelComponent::_loadInstance(fields);
00114     if (res) {
00115         this->_routeTimeExpression = loadField(fields, "routeTimeExpression", "0.0");
00116         this->_routeTimeTimeUnit = static_cast<Util::TimeUnit>
00117             (std::stoi((*fields->find("routeTimeTimeUnit")).second));
00118         this->_routeDestinationType = static_cast<Route::DestinationType> (std::stoi(loadField(fields,
00119             "routeDestinationType", std::to_string(static_cast<int> (DestinationType::Station))));
00120         std::string stationName = ((*fields->find("stationName")).second);
00121         Station* station = dynamic_cast<Station*>
00122             (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00123         this->_station = station;
00124     }
00125     return res;
00126 }
```

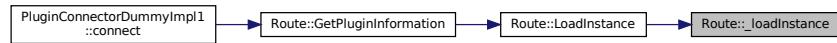
References [ModelComponent::_loadInstance\(\)](#), [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), [loadField\(\)](#), and [Station](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.107.4.6 _saveInstance() `std::map< std::string, std::string > * Route::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelComponent](#).

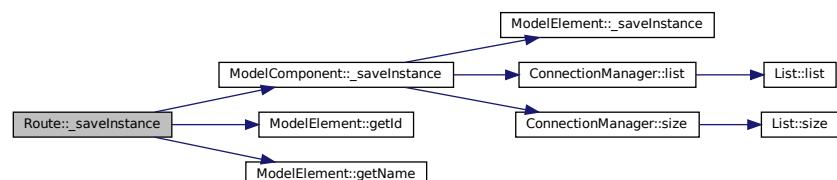
Definition at line 129 of file [Route.cpp](#).

```

00129
00130     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00131     fields->emplace("stationId", std::to_string(this->_station->getId()));
00132     fields->emplace("stationName", "\"" + (this->_station->getName()) + "\"");
00133     if (_routeTimeExpression != "0.0") fields->emplace("routeTimeExpression", "\"" +
00134         this->_routeTimeExpression + "\"");
00135     if (_routeTimeTimeUnit != Util::TimeUnit::second) fields->emplace("routeTimeTimeUnit",
00136         std::to_string(static_cast<int> (this->_routeTimeTimeUnit)));
00137     if (_routeDestinationType != DestinationType::Station) fields->emplace("routeDestinationType",
00138         std::to_string(static_cast<int> (this->_routeDestinationType)));
00139     return fields;
00140 }
```

References [ModelComponent::_saveInstance\(\)](#), [ModelElement::getId\(\)](#), [ModelElement::getName\(\)](#), [Util::second](#), and [Station](#).

Here is the call graph for this function:



8.107.4.7 GetPluginInformation() `PluginInformation * Route::GetPluginInformation () [static]`

Definition at line 172 of file [Route.cpp](#).

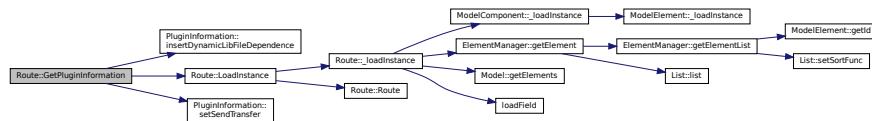
```

00172
00173     PluginInformation* info = new PluginInformation(Util::TypeOf<Route>(), &Route::LoadInstance);
00174     info->setSendTransfer(true);
00175     info->insertDynamicLibFileDependence("station.so");
00176     return info;
00177 }
```

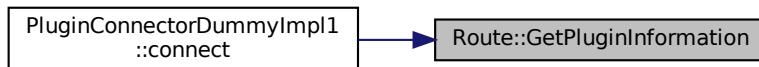
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), [LoadInstance\(\)](#), and [PluginInformation::setSendTransfer\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.107.4.8 getRouteDestinationType() `Route::DestinationType` `Route::getRouteDestinationType () const`

Definition at line 71 of file `Route.cpp`.

```
00071
00072     return _routeDestinationType;
00073 }
```

8.107.4.9 getRouteTimeExpression() `std::string` `Route::getRouteTimeExpression () const`

Definition at line 55 of file `Route.cpp`.

```
00055
00056     return _routeTimeExpression;
00057 }
```

8.107.4.10 getRouteTimeTimeUnit() `Util::TimeUnit` `Route::getRouteTimeTimeUnit () const`

Definition at line 63 of file `Route.cpp`.

```
00063
00064     return _routeTimeTimeUnit;
00065 }
```

8.107.4.11 getStation() `Station *` `Route::getStation () const`

Definition at line 47 of file `Route.cpp`.

```
00047
00048     return _station;
00049 }
```

```
8.107.4.12 LoadInstance() ModelComponent * Route::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

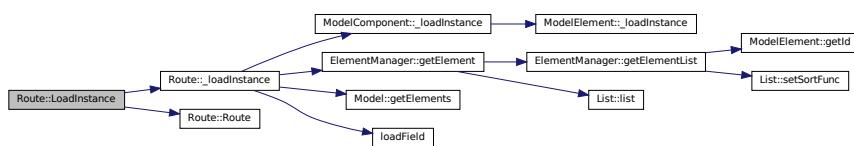
Definition at line 33 of file [Route.cpp](#).

```
00033
00034     Route* newComponent = new Route(model);
00035     try {
00036         newComponent->_loadInstance(fields);
00037     } catch (const std::exception& e) {
00038     }
00039 }
00040 return newComponent;
00041 }
```

References [_loadInstance\(\)](#), and [Route\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.107.4.13 setRouteDestinationType() void Route::setRouteDestinationType (
    DestinationType _routeDestinationType )
```

Definition at line 67 of file [Route.cpp](#).

```
00067
00068     this->_routeDestinationType = _routeDestinationType;
00069 }
```

Referenced by [Model_StatationRouteSequence::main\(\)](#).

Here is the caller graph for this function:



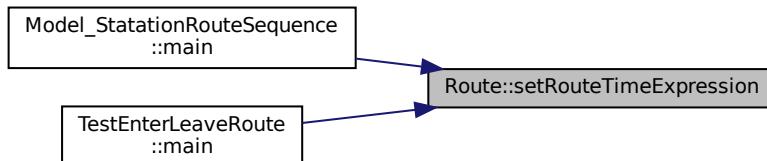
8.107.4.14 setRouteTimeExpression() void Route::setRouteTimeExpression (std::string _routeTimeExpression)

Definition at line 51 of file [Route.cpp](#).

```
00051
00052     this->_routeTimeExpression = _routeTimeExpression;
00053 }
```

Referenced by [Model_StatationRouteSequence::main\(\)](#), and [TestEnterLeaveRoute::main\(\)](#).

Here is the caller graph for this function:



8.107.4.15 setRouteTimeTimeUnit() void Route::setRouteTimeTimeUnit (Util::TimeUnit _routeTimeTimeUnit)

Definition at line 59 of file [Route.cpp](#).

```
00059
00060     this->_routeTimeTimeUnit = _routeTimeTimeUnit;
00061 }
```

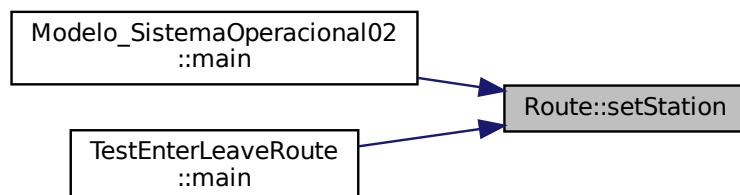
8.107.4.16 setStation() void Route::setStation (Station * _station)

Definition at line 43 of file [Route.cpp](#).

```
00043
00044     this->_station = _station;
00045 }
```

Referenced by [Modelo_SistemaOperacional02::main\(\)](#), and [TestEnterLeaveRoute::main\(\)](#).

Here is the caller graph for this function:



8.107.4.17 `show()` `std::string Route::show() [virtual]`

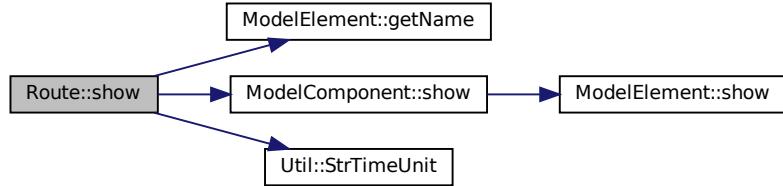
Reimplemented from [ModelComponent](#).

Definition at line 24 of file [Route.cpp](#).

```
00024     {
00025         std::string msg = ModelComponent::show() +
00026             ",destinationType=" + std::to_string(static_cast<int> (this->_routeDestinationType)) +
00027             ",timeExpression=" + this->_routeTimeExpression + " " +
00028             Util::StrTimeUnit(this->_routeTimeTimeUnit);
00029         if (_station != nullptr)
00030             msg += ",station=" + this->_station->getName();
00031     }
00031 }
```

References [ModelElement::getName\(\)](#), [ModelComponent::show\(\)](#), and [Util::StrTimeUnit\(\)](#).

Here is the call graph for this function:



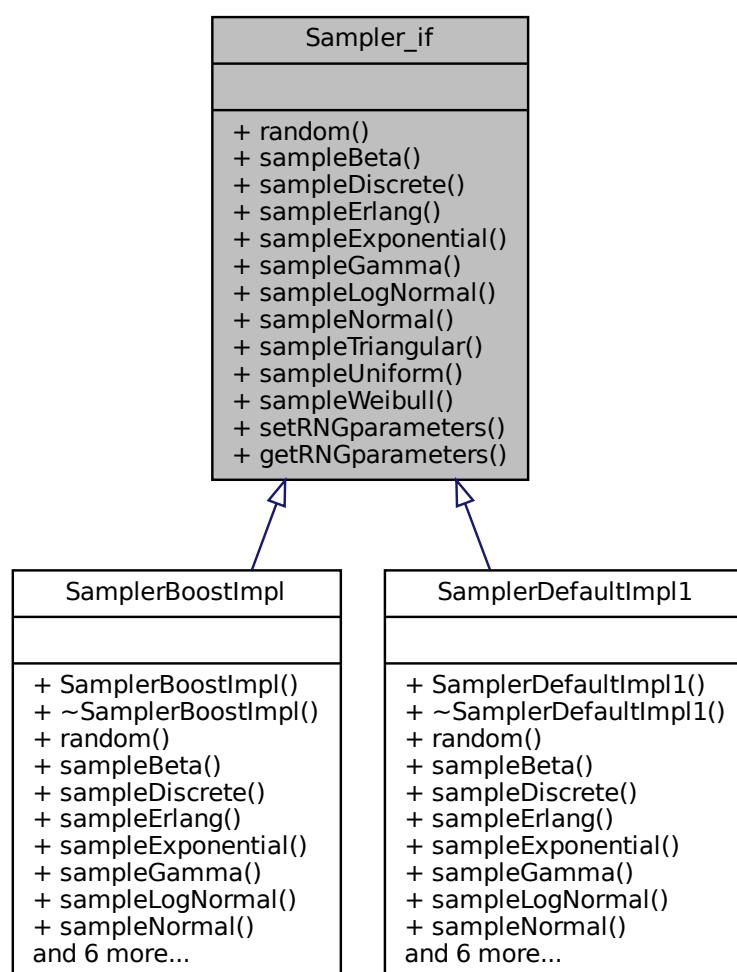
The documentation for this class was generated from the following files:

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

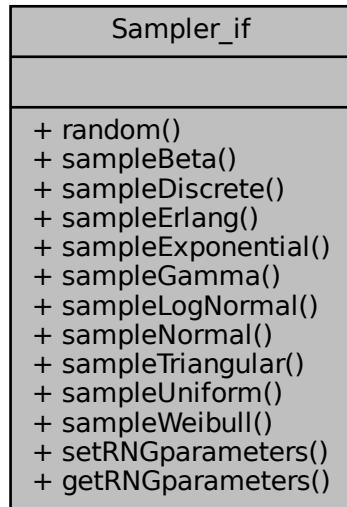
8.108 Sampler_if Class Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for Sampler_if:



Collaboration diagram for Sampler_if:



Classes

- struct [RNG_Parameters](#)

Public Member Functions

- virtual double [random \(\)=0](#)
- virtual double [sampleBeta \(double alpha, double beta, double infLimit, double supLimit\)=0](#)
- virtual double [sampleDiscrete \(double acumProb, double value,...\)=0](#)
- virtual double [sampleErlang \(double mean, int M\)=0](#)
- virtual double [sampleExponential \(double mean\)=0](#)
- virtual double [sampleGamma \(double mean, double alpha\)=0](#)
- virtual double [sampleLogNormal \(double mean, double stddev\)=0](#)
- virtual double [sampleNormal \(double mean, double stddev\)=0](#)
- virtual double [sampleTriangular \(double min, double mode, double max\)=0](#)
- virtual double [sampleUniform \(double min, double max\)=0](#)
- virtual double [sampleWeibull \(double alpha, double scale\)=0](#)
- virtual void [setRNGparameters \(RNG_Parameters *param\)=0](#)
- virtual [RNG_Parameters * getRNGparameters \(\) const =0](#)

8.108.1 Detailed Description

Interface that describes the methods to be implemented by classes that generate random values that follow a specific probability distribution.

Definition at line 20 of file [Sampler_if.h](#).

8.108.2 Member Function Documentation

8.108.2.1 getRNGparameters() virtual `RNG_Parameters*` Sampler_if::getRNGparameters () const [pure virtual]

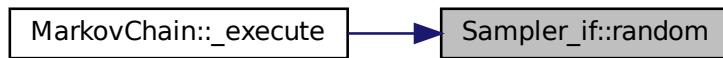
Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

8.108.2.2 random() virtual double Sampler_if::random () [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Referenced by [MarkovChain::_execute\(\)](#).

Here is the caller graph for this function:



8.108.2.3 sampleBeta() virtual double Sampler_if::sampleBeta (double *alpha*, double *beta*, double *infLimit*, double *supLimit*) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

8.108.2.4 sampleDiscrete() virtual double Sampler_if::sampleDiscrete (double *acumProb*, double *value*, ...) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.5 sampleErlang() virtual double Sampler_if::sampleErlang (
    double mean,
    int M) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.6 sampleExponential() virtual double Sampler_if::sampleExponential (
    double mean) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.7 sampleGamma() virtual double Sampler_if::sampleGamma (
    double mean,
    double alpha) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.8 sampleLogNormal() virtual double Sampler_if::sampleLogNormal (
    double mean,
    double stddev) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.9 sampleNormal() virtual double Sampler_if::sampleNormal (
    double mean,
    double stddev) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.10 sampleTriangular() virtual double Sampler_if::sampleTriangular (
    double min,
    double mode,
    double max) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.11 sampleUniform() virtual double Sampler_if::sampleUniform (
    double min,
    double max ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.12 sampleWeibull() virtual double Sampler_if::sampleWeibull (
    double alpha,
    double scale ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

```
8.108.2.13 setRNGparameters() virtual void Sampler_if::setRNGparameters (
    RNG_Parameters * param ) [pure virtual]
```

Implemented in [SamplerBoostImpl](#), and [SamplerDefaultImpl1](#).

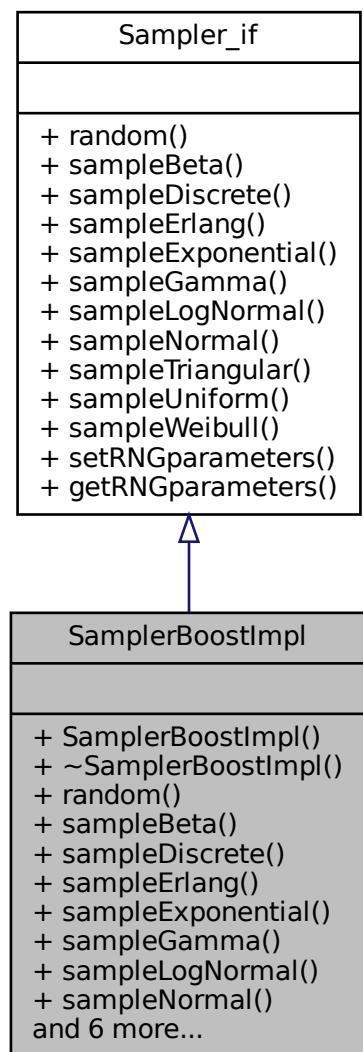
The documentation for this class was generated from the following file:

- [Sampler_if.h](#)

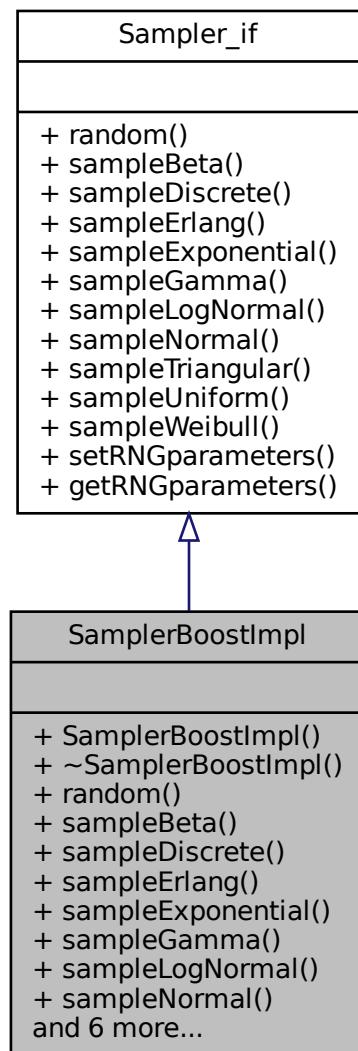
8.109 SamplerBoostImpl Class Reference

```
#include <SamplerBoostImpl.h>
```

Inheritance diagram for SamplerBoostImpl:



Collaboration diagram for SamplerBoostImpl:



Classes

- struct [BoostImplRNG_Parameters](#)

Public Member Functions

- [SamplerBoostImpl \(\)](#)
- virtual [~SamplerBoostImpl \(\)=default](#)
- virtual double [random \(\)](#)
- virtual double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)
- virtual double [sampleDiscrete](#) (double acumProb, double value,...)
- virtual double [sampleErlang](#) (double mean, int M)

- virtual double `sampleExponential` (double mean)
- virtual double `sampleGamma` (double mean, double alpha)
- virtual double `sampleLogNormal` (double mean, double stddev)
- virtual double `sampleNormal` (double mean, double stddev)
- virtual double `sampleTriangular` (double min, double mode, double max)
- virtual double `sampleUniform` (double min, double max)
- virtual double `sampleWeibull` (double alpha, double scale)
- void `reset` ()
reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.
- virtual void `setRNGparameters` (`Sampler_if::RNG_Parameters` *param)
- virtual `RNG_Parameters` * `getRNGparameters` () const

8.109.1 Detailed Description

Definition at line 20 of file `SamplerBoostImpl.h`.

8.109.2 Constructor & Destructor Documentation

8.109.2.1 `SamplerBoostImpl()` `SamplerBoostImpl::SamplerBoostImpl` ()

Definition at line 18 of file `SamplerBoostImpl.cpp`.

```
00018 {  
00019 }
```

8.109.2.2 `~SamplerBoostImpl()` `virtual SamplerBoostImpl::~SamplerBoostImpl` () [virtual], [default]

8.109.3 Member Function Documentation

8.109.3.1 `getRNGparameters()` `Sampler_if::RNG_Parameters` * `SamplerBoostImpl::getRNGparameters` () const [virtual]

Implements `Sampler_if`.

Definition at line 71 of file `SamplerBoostImpl.cpp`.

```
00071 {  
00072 // \todo: toimplement  
00073 }
```

8.109.3.2 random() double SamplerBoostImpl::random () [virtual]

Implements [Sampler_if](#).

Definition at line 21 of file [SamplerBoostImpl.cpp](#).

```
00021      {
00022          //uniform_int_distribution<> dist(0.0, 1.0);
00023          return 0.0; // dist(_gen);
00024 }
```

8.109.3.3 reset() void SamplerBoostImpl::reset ()

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

8.109.3.4 sampleBeta() double SamplerBoostImpl::sampleBeta (

```
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 26 of file [SamplerBoostImpl.cpp](#).

```
00026
00027     return 0.0; //dummy
00028 }
```

8.109.3.5 sampleDiscrete() double SamplerBoostImpl::sampleDiscrete (

```
    double acumProb,
    double value,
    ... ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 30 of file [SamplerBoostImpl.cpp](#).

```
00030
00031     return 0.0; //dummy
00032 }
```

8.109.3.6 sampleErlang() double SamplerBoostImpl::sampleErlang (

```
    double mean,
    int M ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 34 of file [SamplerBoostImpl.cpp](#).

```
00034
00035     return 0.0; //dummy
00036 }
```

8.109.3.7 sampleExponential() double SamplerBoostImpl::sampleExponential (double mean) [virtual]

Implements [Sampler_if](#).

Definition at line 38 of file [SamplerBoostImpl.cpp](#).

```
00038     return 0.0; //dummy
00039
00040 }
```

8.109.3.8 sampleGamma() double SamplerBoostImpl::sampleGamma (double mean, double alpha) [virtual]

Implements [Sampler_if](#).

Definition at line 42 of file [SamplerBoostImpl.cpp](#).

```
00042
00043     return 0.0; //dummy
00044 }
```

8.109.3.9 sampleLogNormal() double SamplerBoostImpl::sampleLogNormal (double mean, double stddev) [virtual]

Implements [Sampler_if](#).

Definition at line 46 of file [SamplerBoostImpl.cpp](#).

```
00046
00047     return 0.0; //dummy
00048 }
```

8.109.3.10 sampleNormal() double SamplerBoostImpl::sampleNormal (double mean, double stddev) [virtual]

Implements [Sampler_if](#).

Definition at line 50 of file [SamplerBoostImpl.cpp](#).

```
00050
00051     return 0.0; //dummy
00052 }
```

8.109.3.11 sampleTriangular() double SamplerBoostImpl::sampleTriangular (double min, double mode, double max) [virtual]

Implements [Sampler_if](#).

Definition at line 54 of file [SamplerBoostImpl.cpp](#).

```
00054
00055     return 0.0; //dummy
00056 }
```

```
8.109.3.12 sampleUniform() double SamplerBoostImpl::sampleUniform (
    double min,
    double max ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 58 of file [SamplerBoostImpl.cpp](#).

```
00058
00059     //uniform_int_distribution<> dist(min, max);
00060     return 0.0; //dist(_gen);
00061 }
```

```
8.109.3.13 sampleWeibull() double SamplerBoostImpl::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 63 of file [SamplerBoostImpl.cpp](#).

```
00063
00064     return 0.0; //dummy
00065 }
```

```
8.109.3.14 setRNGparameters() void SamplerBoostImpl::setRNGparameters (
    Sampler_if::RNG_Parameters * param ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 67 of file [SamplerBoostImpl.cpp](#).

```
00067
00068
00069 }
```

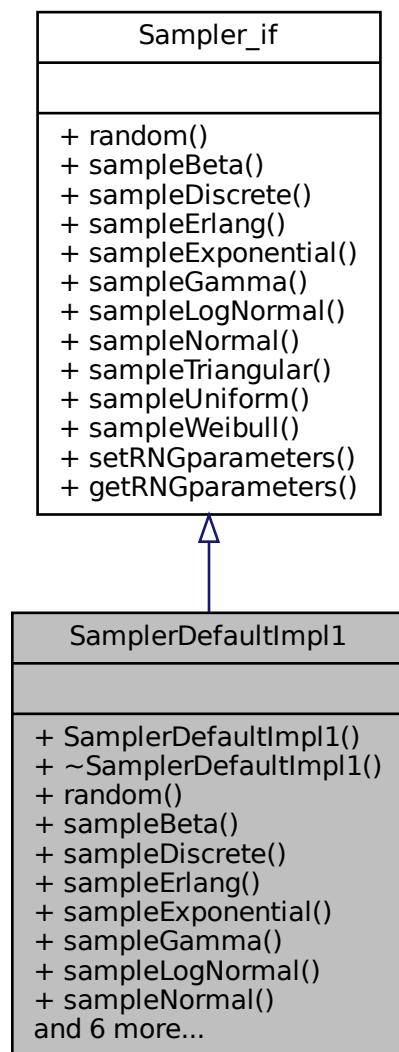
The documentation for this class was generated from the following files:

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

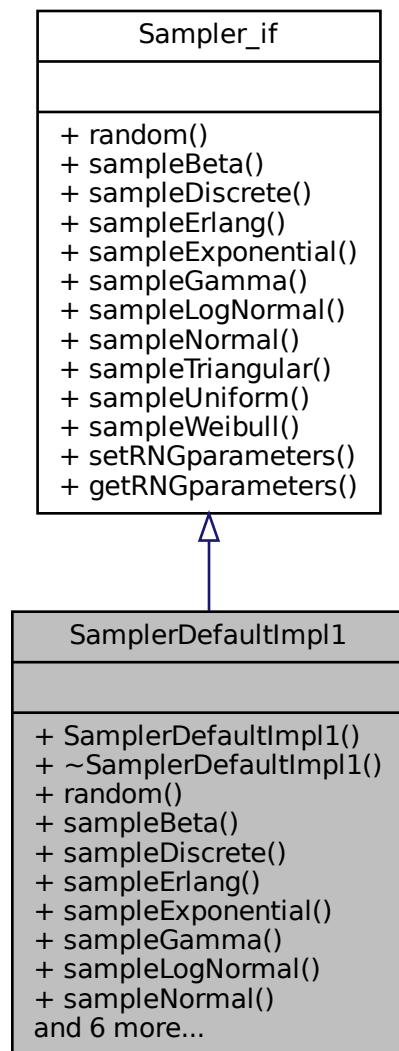
8.110 SamplerDefaultImpl1 Class Reference

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

Inheritance diagram for SamplerDefaultImpl1:



Collaboration diagram for SamplerDefaultImpl1:



Classes

- struct [DefaultImpl1RNG_Parameters](#)

Public Member Functions

- [SamplerDefaultImpl1 \(\)](#)
- virtual [~SamplerDefaultImpl1 \(\)=default](#)
- virtual double [random \(\)](#)
- virtual double [sampleBeta \(double alpha, double beta, double infLimit, double supLimit\)](#)
- virtual double [sampleDiscrete \(double acumProb, double value,...\)](#)
- virtual double [sampleErlang \(double mean, int M\)](#)

- virtual double `sampleExponential` (double mean)
- virtual double `sampleGamma` (double mean, double alpha)
- virtual double `sampleLogNormal` (double mean, double stddev)
- virtual double `sampleNormal` (double mean, double stddev)
- virtual double `sampleTriangular` (double min, double mode, double max)
- virtual double `sampleUniform` (double min, double max)
- virtual double `sampleWeibull` (double alpha, double scale)
- void `reset` ()
reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.
- virtual void `setRNGparameters` (`RNG_Parameters` *param)
- virtual `RNG_Parameters` * `getRNGparameters` () const

8.110.1 Detailed Description

Definition at line 20 of file `SamplerDefaultImpl1.h`.

8.110.2 Constructor & Destructor Documentation

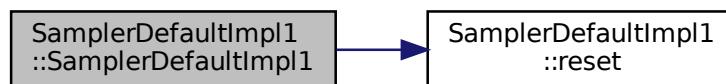
8.110.2.1 `SamplerDefaultImpl1()` `SamplerDefaultImpl1::SamplerDefaultImpl1 ()`

Definition at line 23 of file `SamplerDefaultImpl1.cpp`.

```
00023 {  
00024     reset ();  
00025 }
```

References `reset()`.

Here is the call graph for this function:



8.110.2.2 `~SamplerDefaultImpl1()` `virtual SamplerDefaultImpl1::~SamplerDefaultImpl1 () [virtual], [default]`

8.110.3 Member Function Documentation

8.110.3.1 getRNGparameters() `Sampler_if::RNG_Parameters * SamplerDefaultImpl1::getRNGparameters() const [virtual]`

Implements [Sampler_if](#).

Definition at line 160 of file [SamplerDefaultImpl1.cpp](#).

```
00160
00161     return _param;
00162 }
```

8.110.3.2 random() `double SamplerDefaultImpl1::random() [virtual]`

Implements [Sampler_if](#).

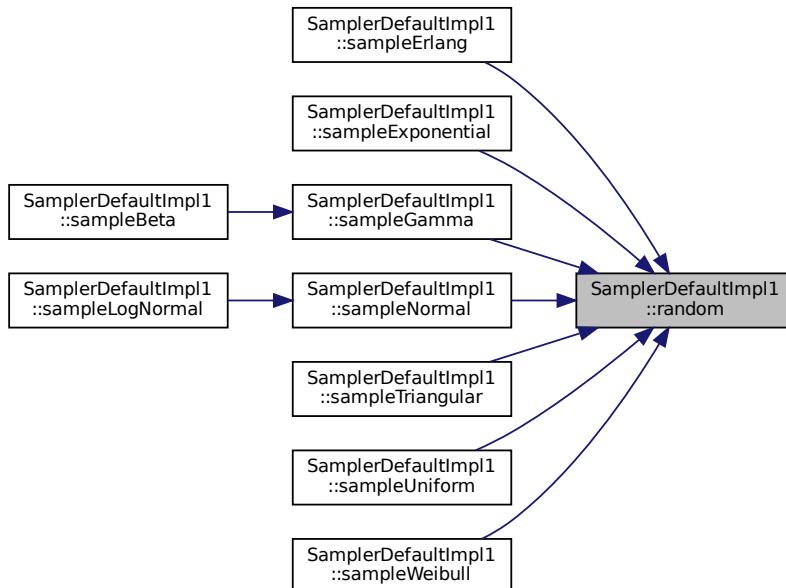
Definition at line 32 of file [SamplerDefaultImpl1.cpp](#).

```
00032
00033     double module = (double) static_cast<DefaultImpl1RNG_Parameters*>(_param)->module;
00034     _xi *= static_cast<DefaultImpl1RNG_Parameters*>(_param)->multiplier;
00035     _xi -= std::trunc((double) _xi / module) * module;
00036     return (double) _xi / (double) static_cast<DefaultImpl1RNG_Parameters*>(_param)->module;
00037 }
```

References [SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::module](#).

Referenced by [sampleErlang\(\)](#), [sampleExponential\(\)](#), [sampleGamma\(\)](#), [sampleNormal\(\)](#), [sampleTriangular\(\)](#), [sampleUniform\(\)](#), and [sampleWeibull\(\)](#).

Here is the caller graph for this function:



8.110.3.3 `reset()` void SamplerDefaultImpl1::reset ()

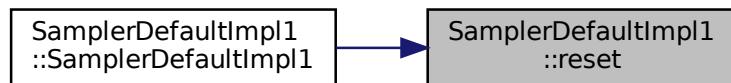
reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

Definition at line 27 of file `SamplerDefaultImpl1.cpp`.

```
00027      {
00028          _xi = static_cast<DefaultImpl1RNG_Parameters*> (_param)->seed;
00029          //__normalflag = true;
00030      }
```

Referenced by [SamplerDefaultImpl1\(\)](#).

Here is the caller graph for this function:



8.110.3.4 `sampleBeta()` double SamplerDefaultImpl1::sampleBeta (

```
double alpha,
double beta,
double infLimit,
double supLimit ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 113 of file `SamplerDefaultImpl1.cpp`.

```
00113
00114     double X, Y1, Y2;
00115     assert(!((alpha <= 0.0) || (beta <= 0.0) || (infLimit > supLimit) || (infLimit < 0) || (supLimit <
0)));
00116     do {
00117         Y1 = sampleGamma(alpha, alpha);
00118         Y2 = sampleGamma(beta, beta);
00119         X = Y1 / (Y1 + Y2);
00120     } while (!((X >= 0) && (X <= 1.0)));
00121     return infLimit + (supLimit - infLimit) * X;
00122 }
```

References [sampleGamma\(\)](#).

Here is the call graph for this function:



8.110.3.5 sampleDiscrete() double SamplerDefaultImpl1::sampleDiscrete (double acumProb,
double value,
...) [virtual]

Implements [Sampler_if](#).

Definition at line 150 of file [SamplerDefaultImpl1.cpp](#).

```
00150
00151     // \todo: to implement
00152     return 0.0;
00153 }
```

8.110.3.6 sampleErlang() double SamplerDefaultImpl1::sampleErlang (

```
double mean,
int M ) [virtual]
```

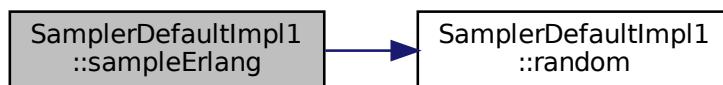
Implements [Sampler_if](#).

Definition at line 47 of file [SamplerDefaultImpl1.cpp](#).

```
00047
00048     int i;
00049     double P;
00050     assert((mean >= 0.0) && (M > 0));
00051     P = 1;
00052     for (i = 1; i <= M; i++) {
00053         P *= random();
00054     }
00055     return (mean / M) * (-log(P));
00056 }
```

References [random\(\)](#).

Here is the call graph for this function:



8.110.3.7 sampleExponential() double SamplerDefaultImpl1::sampleExponential (double mean) [virtual]

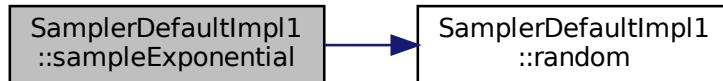
Implements [Sampler_if](#).

Definition at line 43 of file [SamplerDefaultImpl1.cpp](#).

```
00043
00044     return mean * (-std::log(random()));
00045 }
```

References [random\(\)](#).

Here is the call graph for this function:



8.110.3.8 sampleGamma() double SamplerDefaultImpl1::sampleGamma (double mean, double alpha) [virtual]

Implements [Sampler_if](#).

Definition at line 86 of file [SamplerDefaultImpl1.cpp](#).

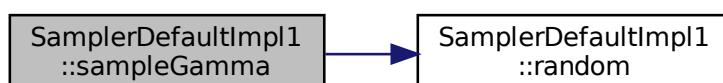
```

00086
00087     int i;
00088     double P;
00089     int IntAlpha;
00090     double OstAlpha;
00091     assert(!((mean <= 0.0) || (alpha <= 0.0)));
00092     if (alpha < 1.0)
00093         return (mean / alpha) * _gammaJong(alpha);
00094     else {
00095         if (alpha == 1.0)
00096             return mean * (-log(random()));
00097         else {
00098             IntAlpha = round(alpha);
00099             OstAlpha = alpha - IntAlpha;
00100             do {
00101                 P = 1;
00102                 for (i = 1; i <= IntAlpha; i++)
00103                     P *= random();
00104                 } while (!(P > 0));
00105                 if (OstAlpha > 0)
00106                     return (mean / alpha)*((-log(P)) + _gammaJong(OstAlpha));
00107                 else
00108                     return (mean / alpha)*(-log(P));
00109             };
00110     };
00111 }
```

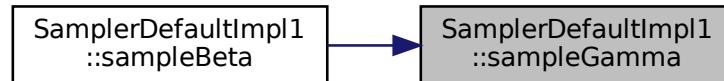
References [random\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.110.3.9 sampleLogNormal() double SamplerDefaultImpl1::sampleLogNormal (double mean, double stddev) [virtual]

Implements [Sampler_if](#).

Definition at line 129 of file [SamplerDefaultImpl1.cpp](#).

```

00129
00130     double meanNorm, DispNorm;
00131     assert(!((mean <= 0.0) || (stddev <= 0.0)));
00132     DispNorm = log((stddev * stddev) / (mean * mean) + 1.0);
00133     meanNorm = log(mean) - 0.5 * DispNorm;
00134     return exp(sampleNormal(meanNorm, sqrt(DispNorm)));
00135 }
```

References [sampleNormal\(\)](#).

Here is the call graph for this function:



8.110.3.10 sampleNormal() double SamplerDefaultImpl1::sampleNormal (double mean, double stddev) [virtual]

Implements [Sampler_if](#).

Definition at line 58 of file [SamplerDefaultImpl1.cpp](#).

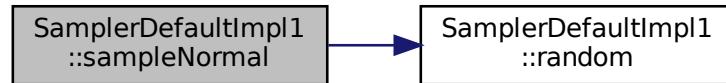
```

00058
00059     double z = std::sqrt(-2 * std::log(random())) * std::cos(2 * M_PI * random());
00060     return mean + stddev*z;
00061 }
```

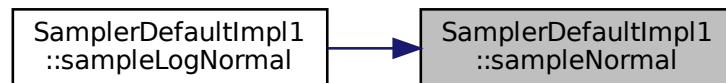
References [random\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.110.3.11 sampleTriangular() double SamplerDefaultImpl1::sampleTriangular (double *min*, double *mode*, double *max*) [virtual]

Implements [Sampler_if](#).

Definition at line 137 of file [SamplerDefaultImpl1.cpp](#).

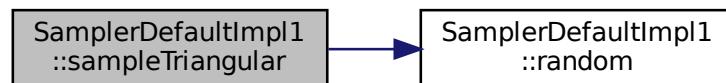
```

00137
00138     double Part1, Part2, Full, R;
00139     assert(!((min > mode) || (max < mode) || (min > max)));
00140     Part1 = mode - min;
00141     Part2 = max - mode;
00142     Full = max - min;
00143     R = random();
00144     if (R <= Part1 / Full)
00145         return min + sqrt(Part1 * Full * R);
00146     else
00147         return max - sqrt(Part2 * Full * (1.0 - R));
00148 }

```

References [random\(\)](#).

Here is the call graph for this function:



```
8.110.3.12 sampleUniform() double SamplerDefaultImpl1::sampleUniform (
    double min,
    double max ) [virtual]
```

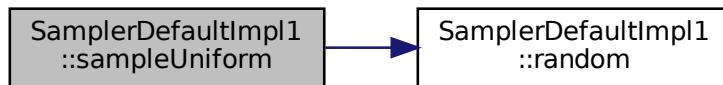
Implements [Sampler_if](#).

Definition at line 39 of file [SamplerDefaultImpl1.cpp](#).

```
00039
00040     return min + (max - min) * random();
00041 }
```

References [random\(\)](#).

Here is the call graph for this function:



```
8.110.3.13 sampleWeibull() double SamplerDefaultImpl1::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

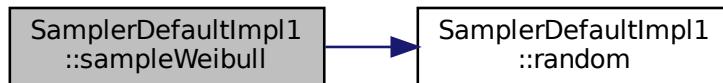
Implements [Sampler_if](#).

Definition at line 124 of file [SamplerDefaultImpl1.cpp](#).

```
00124
00125     assert(!((alpha <= 0.0) || (scale <= 0.0)));
00126     return exp(log(scale * (-log(random())))) / alpha;
00127 }
```

References [random\(\)](#).

Here is the call graph for this function:



8.110.3.14 setRNGparameters() void SamplerDefaultImpl1::setRNGparameters (Sampler_if::RNG_Parameters * param) [virtual]

Implements [Sampler_if](#).

Definition at line 155 of file [SamplerDefaultImpl1.cpp](#).

```
00155
00156
00157     _param = param; // there is a better solution for this...
00158 }
```

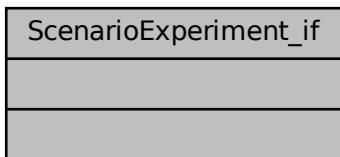
The documentation for this class was generated from the following files:

- [SamplerDefaultImpl1.h](#)
- [SamplerDefaultImpl1.cpp](#)

8.111 ScenarioExperiment_if Class Reference

```
#include <ScenarioExperiment_if.h>
```

Collaboration diagram for ScenarioExperiment_if:



8.111.1 Detailed Description

Definition at line 17 of file [ScenarioExperiment_if.h](#).

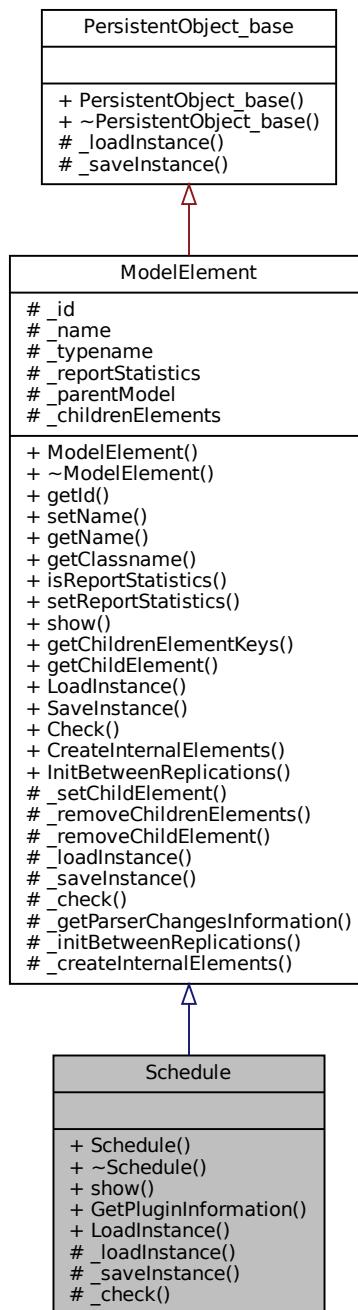
The documentation for this class was generated from the following file:

- [ScenarioExperiment_if.h](#)

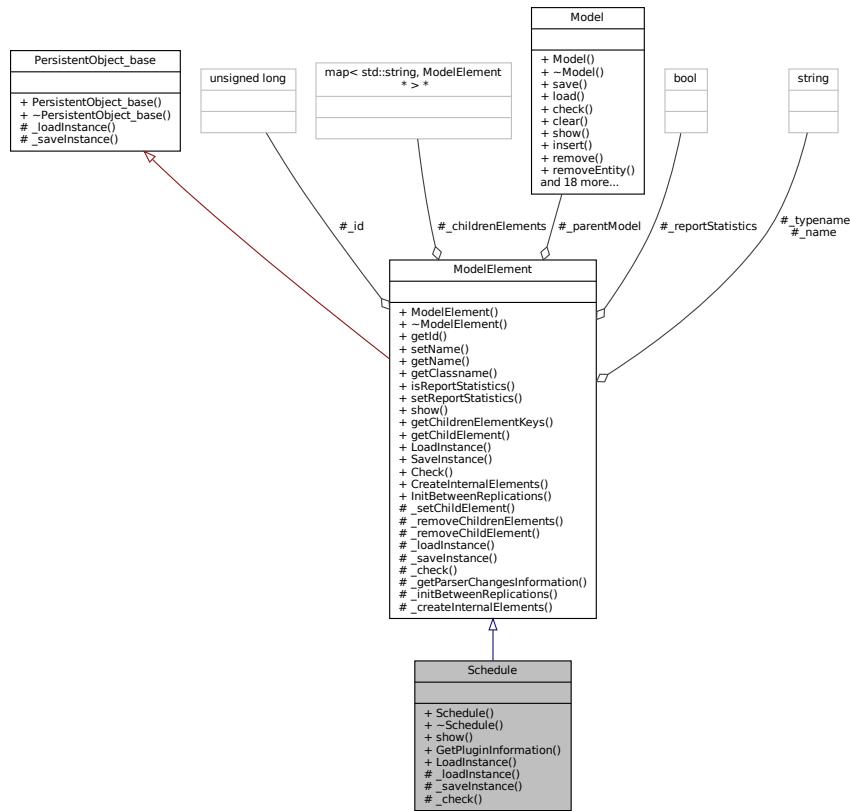
8.112 Schedule Class Reference

```
#include <Schedule.h>
```

Inheritance diagram for Schedule:



Collaboration diagram for Schedule:



Public Member Functions

- `Schedule (Model *model, std::string name="")`
- virtual `~Schedule ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.112.1 Detailed Description

Schedule module DESCRIPTION This data module may be used in conjunction with the [Resource](#) module to define an operating schedule for a resource or with the [Create](#) module to define an arrival schedule. Additionally, a schedule may be used and referenced to factor time delays based on the simulation time. TYPICAL USES Work schedule for staff, including breaks Breakdown patterns for equipment Volume of customers arriving at a store Learning-curve factors for new workers PROMPTS [File](#) Read Time Specifies when to read the values from the file into the variable. If you select PreCheck, the values for the variable are read while the model is still in Edit mode (prior to the model being checked and compiled). If you select BeginSimulation, values are read when the model is compiled, prior to the first replication. If you select BeginReplication, values are read prior to each replication. Initial Values Lists the initial value or values of the variable. You can assign new values to the variable at different stages of the model by using the [Assign](#) module. Initial Value [Variable](#) value at the start of the simulation. Prompt Description Name The name of the schedule being defined. This name must be unique. Type Type of schedule being defined. This may be Capacity-related (for resource schedules), Arrival-related (for the [Create](#) module), or Other (miscellaneous time delays or factors) Time Units Time units used for the time-duration information. Scale Factor Method of scaling the schedule for increases or decreases in Arrival/Other values. The specified Value fields will be multiplied by the scale factor to determine the new values. Not available for Capacity-type schedules. Durations Lists the value and duration pairs for the schedule. Values can be capacity, arrival, or other type values, while the duration is specified in time units. [Schedule](#) pairs will repeat after all durations have been completed, unless the last duration is left blank (infinite). [Schedule](#) data can be entered graphically using the graphical schedule editor or manually using the Value/ Duration fields. Value Represents either the capacity of a resource (if Type is Capacity), arrival rate (if Type is Arrival), or some other value (if Type is Other). Examples of Other may be a factor that is used in a delay expression to scale a delay time during various parts of the day. Duration Time duration for which a specified Value will be valid.

Definition at line 72 of file [Schedule.h](#).

8.112.2 Constructor & Destructor Documentation

```
8.112.2.1 Schedule() Schedule::Schedule (
    Model * model,
    std::string name = "" )
```

Definition at line 16 of file [Schedule.cpp](#).

```
00016 : ModelElement(model, Util::TypeOf<Schedule>(), name) {
00017 }
```

```
8.112.2.2 ~Schedule() virtual Schedule::~Schedule ( ) [virtual], [default]
```

8.112.3 Member Function Documentation

```
8.112.3.1 _check() bool Schedule::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 34 of file [Schedule.cpp](#).

```
00034
00035 }
```

```
8.112.3.2 _loadInstance() bool Schedule::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

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

```
00028
00029 }
```

```
8.112.3.3 _saveInstance() std::map< std::string, std::string > * Schedule::_saveInstance ( )
[protected], [virtual]
```

Reimplemented from [ModelElement](#).

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

```
00031
00032 }
```

```
8.112.3.4 GetPluginInformation() PluginInformation * Schedule::GetPluginInformation ( ) [static]
```

Definition at line 22 of file [Schedule.cpp](#).

```
00022
00023 }
```

```
8.112.3.5 LoadInstance() ModelElement * Schedule::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Schedule.cpp](#).

```
00025
00026 }
```

8.112.3.6 show() std::string Schedule::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 19 of file [Schedule.cpp](#).

```
00019 {  
00020 }
```

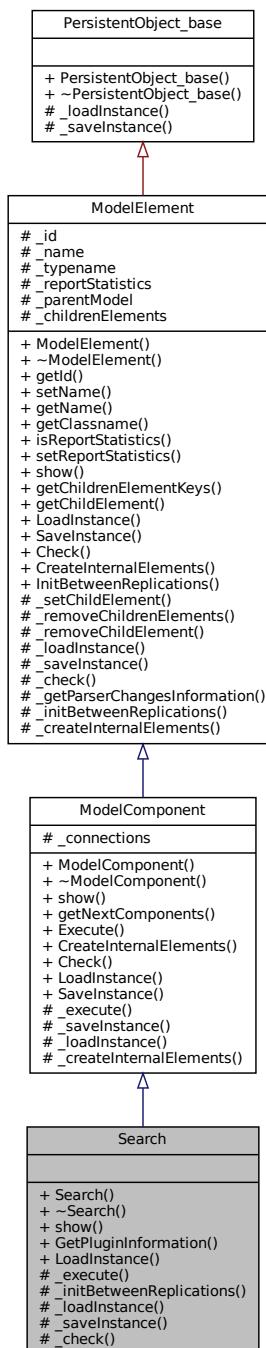
The documentation for this class was generated from the following files:

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

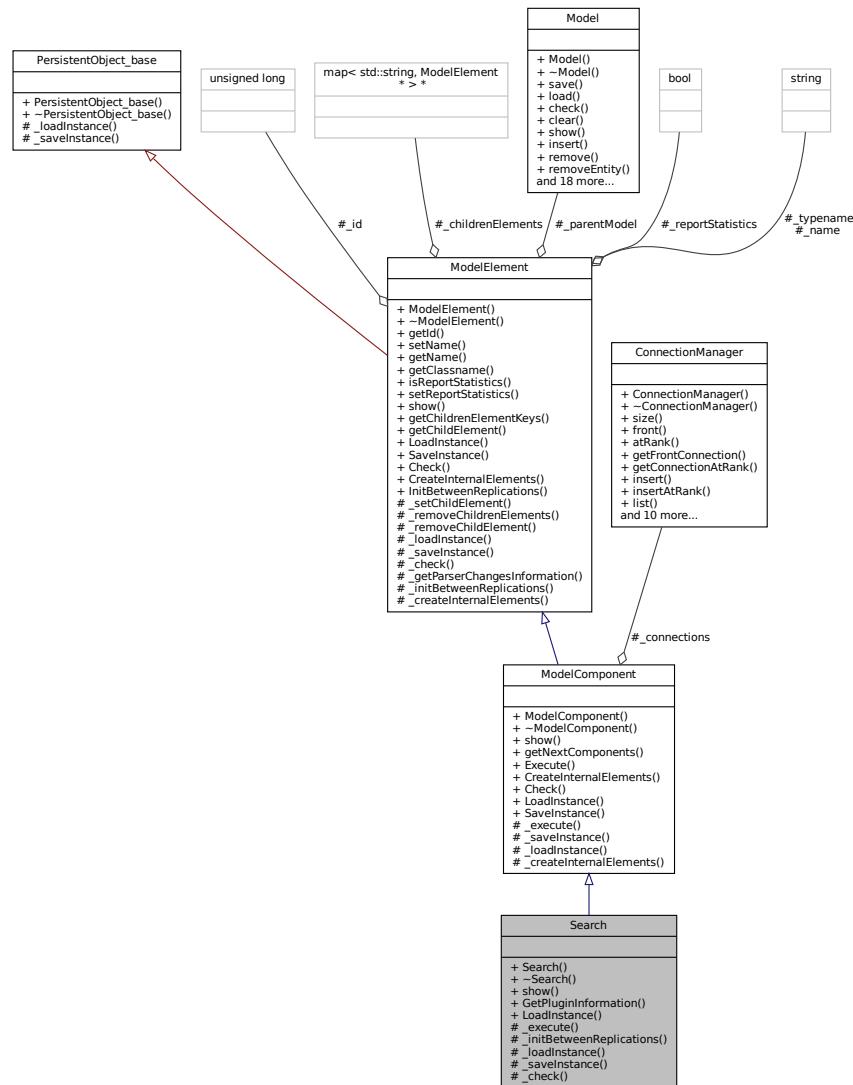
8.113 Search Class Reference

```
#include <Search.h>
```

Inheritance diagram for Search:



Collaboration diagram for Search:



Public Member Functions

- `Search (Model *model, std::string name="")`
- virtual `~Search ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.113.1 Detailed Description

Search module DESCRIPTION The **Search** module searches a queue, a group (batch), or an expression to find the entity rank (for entities in a queue or group) or the value of the global variable J that satisfies the specified search condition. When searching a queue or group, the value of the global system variable J is set to the rank of the first entity that satisfies **Search** Condition, or to 0 if **Search** Condition is not satisfied. When searching an expression, the global system variable J is set to the value of the first index value that satisfies the search condition or to zero if no value of J in the specified range satisfies the search condition. When an entity arrives at a **Search** module, the index J is set to the starting index and the search condition is then checked. If the search condition is satisfied, the search ends and the current value of J is retained. Otherwise, the value of J is increased or decreased and the condition is rechecked. This process repeats until the search condition is satisfied or the ending value is reached. If the condition is not met or there are no entities in the queue or group, J is set equal to 0. TYPICAL USES Looking for a particular order number in a queue Searching a group for a certain part type Determining which process to enter based on availability of resources (search an expression) Prompt Description Name Unique module identifier displayed on the module shape. Type Determination of what will be searched. **Search** options include entities in a queue, entities within a group (batch) or some expression(s). **Queue** Name Name of the queue that will be searched. Applies only when the Type is **Search** a **Queue**. Starting Value Starting rank in the queue or group or starting value for J in an expression. Ending Value Ending rank in the queue or group or ending value for J in an expression. **Search** Condition Condition containing the index J for searching expressions or containing an attribute name(s) for searching queues or batches.

Definition at line 55 of file [Search.h](#).

8.113.2 Constructor & Destructor Documentation

8.113.2.1 Search() `Search::Search (`
 `Model * model,`
 `std::string name = "")`

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

```
00017 : ModelComponent(model, Util::TypeOf<Search>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.113.2.2 ~Search() `virtual Search::~Search () [virtual], [default]`

8.113.3 Member Function Documentation

8.113.3.1 `_check()` `bool Search::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Search.cpp](#).

```
00056     bool resultAll = true;
00057     //...
00058     return resultAll;
00059 }
00060 }
```

8.113.3.2 `_execute()` `void Search::_execute (Entity * entity) [protected], [virtual]`

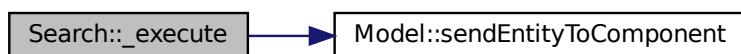
Implements [ModelComponent](#).

Definition at line 34 of file [Search.cpp](#).

```
00034     {
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.113.3.3 `_initBetweenReplications()` `void Search::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Search.cpp](#).

```
00047     {
00048 }
```

```
8.113.3.4 _loadInstance() bool Search::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

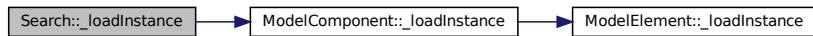
Definition at line 39 of file [Search.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

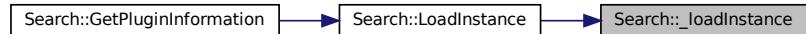
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.113.3.5 _saveInstance() std::map< std::string, std::string > * Search::_saveInstance ( )
[protected], [virtual]
```

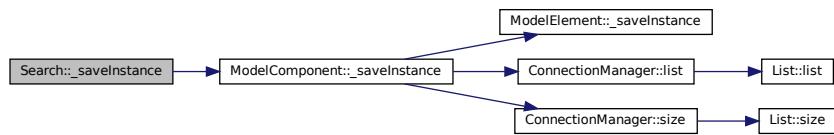
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [Search.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.113.3.6 GetPluginInformation() `PluginInformation * Search::GetPluginInformation () [static]`

Definition at line 62 of file [Search.cpp](#).

```
00062 {  
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Search>(), &Search::LoadInstance);  
00064 // ...  
00065     return info;  
00066 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:

**8.113.3.7 LoadInstance()** `ModelComponent * Search::LoadInstance (`

```
    Model * model,  
    std::map< std::string, std::string > * fields ) [static]
```

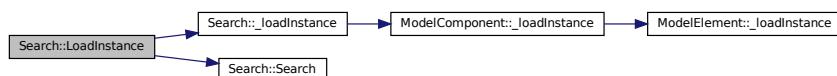
Definition at line 24 of file [Search.cpp](#).

```
00024 {  
00025     Search* newComponent = new Search(model);  
00026     try {  
00027         newComponent->_loadInstance(fields);  
00028     } catch (const std::exception& e) {  
00029     }  
00030     return newComponent;  
00031 }  
00032 }
```

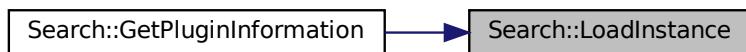
References [_loadInstance\(\)](#), and [Search\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.113.3.8 `show()` `std::string Search::show() [virtual]`

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Search.cpp](#).

```
00020         {
00021     return ModelComponent::show() + "";
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



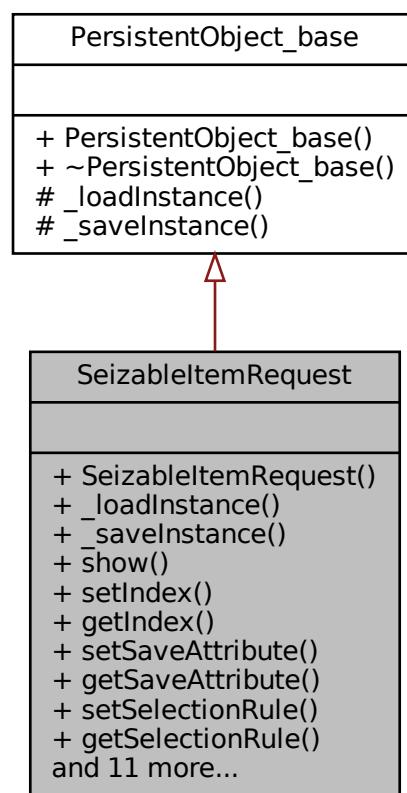
The documentation for this class was generated from the following files:

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

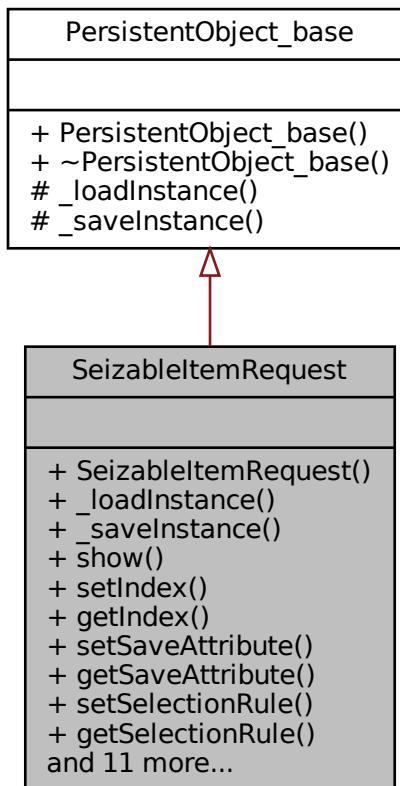
8.114 SeizableItemRequest Class Reference

```
#include <SeizableItemRequest.h>
```

Inheritance diagram for SeizableItemRequest:



Collaboration diagram for SeizableItemRequest:



Public Types

- enum `SelectionRule` : int {
`SelectionRule::CYCLICAL` = 1, `SelectionRule::RANDOM` = 2, `SelectionRule::SPECIFICMEMBER` = 3,
`SelectionRule::LARGESTREMAININGCAPACITY` = 4,
`SelectionRule::SMALLESTNUMBERBUSY` = 5 }
- enum `ResourceType` : int { `ResourceType::RESOURCE` = 1, `ResourceType::SET` = 2 }

Public Member Functions

- `SeizableItemRequest (ModelElement *resourceOrSet, std::string quantityExpression="1", SeizableItemRequest::ResourceType resourceType=SeizableItemRequest::ResourceType::RESOURCE, SeizableItemRequest::SelectionRule selectionRule=SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY, std::string saveAttribute="", unsigned int index=0)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- `std::string show ()`
- `void setIndex (unsigned int index)`
- `unsigned int getIndex () const`
- `void setSaveAttribute (std::string saveAttribute)`

- std::string `getSaveAttribute () const`
- void `setSelectionRule (SelectionRule selectionRule)`
- `SelectionRule getSelectionRule () const`
- void `setQuantityExpression (std::string quantityExpression)`
- std::string `getQuantityExpression () const`
- std::string `getResourceName () const`
- void `setResource (Resource *resource)`
- `Resource * getResource () const`
- void `setSet (Set *set)`
- `Set * getSet () const`
- void `setResourceType (ResourceType resourceType)`
- `ResourceType getResourceType () const`
- void `setLastMemberSeized (unsigned int lastMemberSeized)`
- unsigned int `getLastMemberSeized () const`

8.114.1 Detailed Description

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

8.114.2 Member Enumeration Documentation

8.114.2.1 ResourceType enum `SeizableItemRequest::ResourceType : int [strong]`

Enumerator

RESOURCE	
SET	

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

```
00028      : int {
00029      RESOURCE = 1, SET = 2
00030  };
```

8.114.2.2 SelectionRule enum `SeizableItemRequest::SelectionRule : int [strong]`

Enumerator

CYCLICAL	
RANDOM	
SPECIFICMEMBER	
LARGESTREMAININGCAPACITY	
SMALLESTNUMBERBUSY	

Definition at line 24 of file `SeizableItemRequest.h`.

```
00024      : int {
```

```
00025     CYCLICAL = 1, RANDOM = 2, SPECIFICMEMBER = 3, LARGESTREMAININGCAPACITY = 4, SMALLESTNUMBERBUSY
00026     = 5
00027 };
```

8.114.3 Constructor & Destructor Documentation

8.114.3.1 SeizableItemRequest() SeizableItemRequest::SeizableItemRequest (

```
    ModelElement * resourceOrSet,
    std::string quantityExpression = "1",
    SeizableItemRequest::ResourceType resourceType = SeizableItemRequest::ResourceType::RESOURCE,
    SeizableItemRequest::SelectionRule selectionRule = SeizableItemRequest::SelectionRule::LARGESTREM
    std::string saveAttribute = "",
    unsigned int index = 0 )
```

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

```
00017
00018     {
00019         _resourceType = resourceType;
00020         _resourceOrSet = resourceOrSet;
00021         _quantityExpression = quantityExpression;
00022         _selectionRule = selectionRule;
00023         _saveAttribute = saveAttribute;
00024     }
```

8.114.4 Member Function Documentation

8.114.4.1 _loadInstance() bool SeizableItemRequest::_loadInstance (

```
    std::map< std::string, std::string * > * fields ) [virtual]
```

Implements [PersistentObject_base](#).

Definition at line 26 of file [SeizableItemRequest.cpp](#).

```
00026
00027     bool res = true;
00028     try {
00029         _resourceType = static_cast<SeizableItemRequest::ResourceType>
00030             (std::stoi((*(fields->find("resourceType"))).second));
00031         _resourceName = ((*(fields->find("resourceName"))).second);
00032         //Resource* resource = dynamic_cast<Resource*>
00033         (_parentModel->getElements()->getElement(Util::TypeOf<Resource>(), resourceName));
00034         _quantityExpression = ((*(fields->find("quantity"))).second);
00035         _selectionRule = static_cast<SeizableItemRequest::SelectionRule>
00036             (std::stoi((*(fields->find("rule"))).second));
00037         _saveAttribute = ((*(fields->find("saveAttribute"))).second);
00038         _index = std::stoi((*(fields->find("index"))).second);
00039     } catch (const std::exception& e) {
00040         res = false;
00041     }
00042     return res;
00043 }
```

8.114.4.2 `_saveInstance()` std::map< std::string, std::string > * SeizableItemRequest::_saveInstance () [virtual]

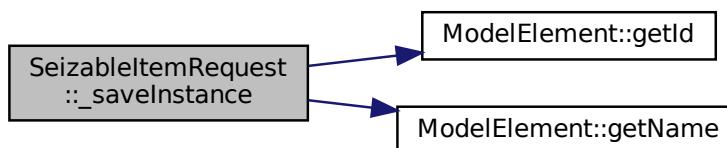
Implements [PersistentObject_base](#).

Definition at line 42 of file [SeizableItemRequest.cpp](#).

```
00042     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00043     fields->emplace("resourceType", std::to_string(static_cast<int> (_resourceType)));
00044     fields->emplace("resourceId", std::to_string(_resourceOrSet->getId()));
00045     fields->emplace("resourceName", _resourceOrSet->getName());
00046     fields->emplace("quantityExpression", _quantityExpression);
00047     fields->emplace("selectionRule", std::to_string(static_cast<int> (_selectionRule)));
00048     fields->emplace("saveAttribute", _saveAttribute);
00049     fields->emplace("index", std::to_string(_index));
00050
00051     return fields;
00052 }
```

References [ModelElement::getId\(\)](#), and [ModelElement::getName\(\)](#).

Here is the call graph for this function:



8.114.4.3 `getIndex()` unsigned int SeizableItemRequest::getIndex () const

Definition at line 62 of file [SeizableItemRequest.cpp](#).

```
00062
00063     return _index;
00064 }
```

8.114.4.4 `getLastMemberSeized()` unsigned int SeizableItemRequest::getLastMemberSeized () const

Definition at line 122 of file [SeizableItemRequest.cpp](#).

```
00122
00123     return _lastMemberSeized;
00124 }
```

8.114.4.5 `getQuantityExpression()` std::string SeizableItemRequest::getQuantityExpression () const

Definition at line 86 of file [SeizableItemRequest.cpp](#).

```
00086
00087     return _quantityExpression;
00088 }
```

8.114.4.6 getResource() `Resource * SeizableItemRequest::getResource () const`

Definition at line 98 of file `SeizableItemRequest.cpp`.

```
00098          {  
00099      return static_cast<Resource*> (_resourceOrSet);  
00100 }
```

8.114.4.7 getResourceName() `std::string SeizableItemRequest::getResourceName () const`

Definition at line 90 of file `SeizableItemRequest.cpp`.

```
00090          {  
00091      return _resourceName;  
00092 }
```

8.114.4.8 getResourceType() `SeizableItemRequest::ResourceType SeizableItemRequest::getResourceType () const`

Definition at line 114 of file `SeizableItemRequest.cpp`.

```
00114          {  
00115      return _resourceType;  
00116 }
```

8.114.4.9 getSaveAttribute() `std::string SeizableItemRequest::getSaveAttribute () const`

Definition at line 70 of file `SeizableItemRequest.cpp`.

```
00070          {  
00071      return _saveAttribute;  
00072 }
```

8.114.4.10 getSelectionRule() `SeizableItemRequest::SelectionRule SeizableItemRequest::getSelectionRule () const`

Definition at line 78 of file `SeizableItemRequest.cpp`.

```
00078          {  
00079      return _selectionRule;  
00080 }
```

8.114.4.11 getSet() `Set * SeizableItemRequest::getSet () const`

Definition at line 106 of file `SeizableItemRequest.cpp`.

```
00106          {  
00107      return static_cast<Set*> (_resourceOrSet);  
00108 }
```

8.114.4.12 setIndex() void SeizableItemRequest::setIndex (unsigned int *index*)

Definition at line 58 of file [SeizableItemRequest.cpp](#).

```
00058     this->_index = _index;
00059 }
00060 }
```

8.114.4.13 setLastMemberSeized() void SeizableItemRequest::setLastMemberSeized (unsigned int *lastMemberSeized*)

Definition at line 118 of file [SeizableItemRequest.cpp](#).

```
00118     this->_lastMemberSeized = lastMemberSeized;
00119 }
00120 }
```

8.114.4.14 setQuantityExpression() void SeizableItemRequest::setQuantityExpression (std::string *quantityExpression*)

Definition at line 82 of file [SeizableItemRequest.cpp](#).

```
00082     this->_quantityExpression = _quantityExpression;
00083 }
00084 }
```

8.114.4.15 setResource() void SeizableItemRequest::setResource (Resource * *resource*)

Definition at line 94 of file [SeizableItemRequest.cpp](#).

```
00094     this->_resourceOrSet = resource;
00095 }
00096 }
```

8.114.4.16 setResourceType() void SeizableItemRequest::setResourceType (SeizableItemRequest::ResourceType *resourceType*)

Definition at line 110 of file [SeizableItemRequest.cpp](#).

```
00110     this->_resourceType = _resourceType;
00111 }
00112 }
```

8.114.4.17 setSaveAttribute() void SeizableItemRequest::setSaveAttribute (std::string *saveAttribute*)

Definition at line 66 of file [SeizableItemRequest.cpp](#).

```
00066     this->_saveAttribute = _saveAttribute;
00067 }
00068 }
```

8.114.4.18 setSelectionRule() void SeizableItemRequest::setSelectionRule (SeizableItemRequest::SelectionRule selectionRule)

Definition at line 74 of file [SeizableItemRequest.cpp](#).

```
00074
00075     this->_selectionRule = _selectionRule;
00076 }
```

8.114.4.19 setSet() void SeizableItemRequest::setSet (Set * set)

Definition at line 102 of file [SeizableItemRequest.cpp](#).

```
00102
00103     this->_resourceOrSet = set;
00104 }
```

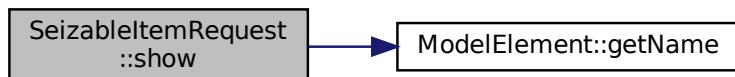
8.114.4.20 show() std::string SeizableItemRequest::show ()

Definition at line 54 of file [SeizableItemRequest.cpp](#).

```
00054
00055     {
00056         return "resourceType=" + std::to_string(static_cast<int> (_resourceType)) + ",resource=" + _resourceOrSet->getName() + "\",quantityExpression=\"" + _quantityExpression + "\", selectionRule=" + std::to_string(static_cast<int> (_selectionRule)) + ", _saveAttribute=\"" + _saveAttribute + "\",index=" + std::to_string(_index);
00056 }
```

References [ModelElement::getName\(\)](#).

Here is the call graph for this function:



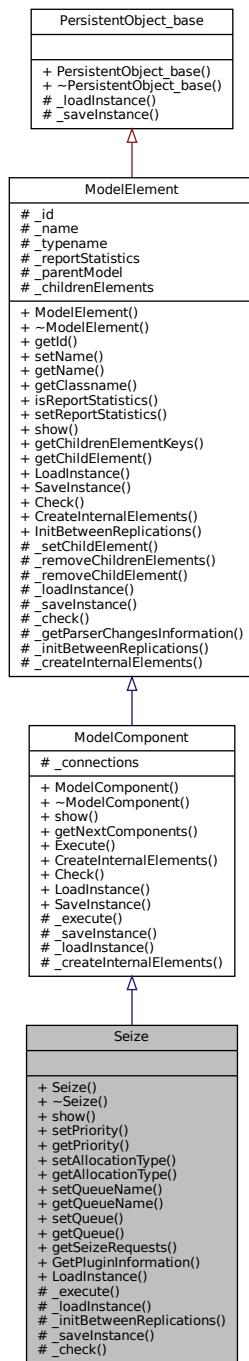
The documentation for this class was generated from the following files:

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

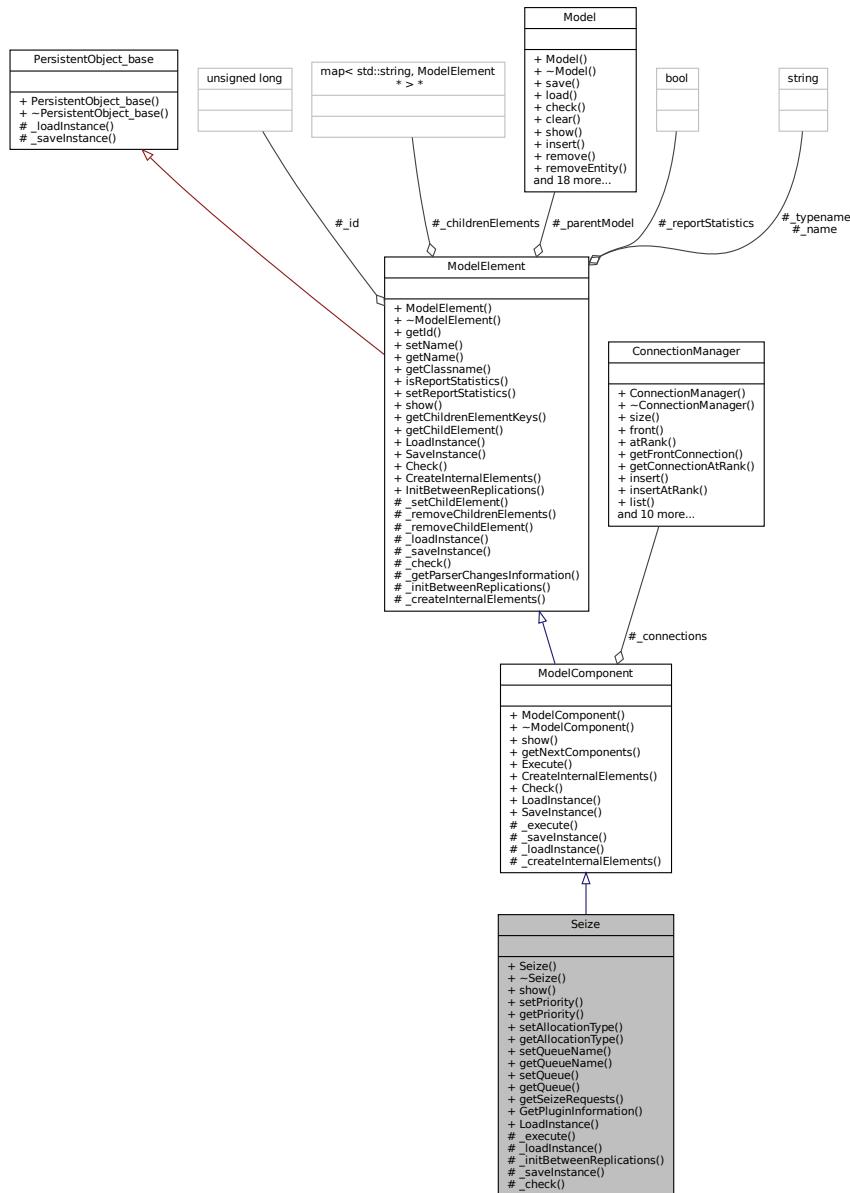
8.115 Seize Class Reference

```
#include <Seize.h>
```

Inheritance diagram for Seize:



Collaboration diagram for Seize:



Public Member Functions

- **Seize (Model *model, std::string name="")**
- virtual **~Seize ()=default**
- virtual std::string **show ()**
- void **setPriority (unsigned short _priority)**
- unsigned short **getPriority () const**
- void **setAllocationType (unsigned int _allocationType)**
- unsigned int **getAllocationType () const**
- void **setQueueName (std::string queueName) throw ()**
- std::string **getQueueName () const**
- void **setQueue (Queue *queue)**
- Queue * **getQueue () const**
- List< SeizableItemRequest * > * **getSeizeRequests () const**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.115.1 Detailed Description

Seize module DESCRIPTION The **Seize** module allocates units of one or more resources to an entity. The **Seize** module may be used to seize units of a particular resource, a member of a resource set, or a resource as defined by an alternative method, such as an attribute or expression. When an entity enters this module, it waits in a queue (if specified) until all specified resources are available simultaneously. Allocation type for resource usage is also specified. TYPICAL USES Beginning a customer order (seize the operator) Starting a tax return (seize the accountant) Being admitted to hospital (seize the hospital room, nurse, doctor) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Allocation Determines to which category the resource usage cost will be allocated for an entity going through the **Seize** module. Priority Priority value of the entity waiting at this module for the resource(s) specified if one or more entities from other modules are waiting for the same resource(s). Type Type of resource for seizing, either specifying a particular resource, or selecting from a pool of resources (that is, a resource set). The name of the resource may also be specified as an attribute value or within an expression. Resource Name Name of the resource that will be seized. Set Name Name of the resource set from which a member will be seized. Attribute Name Name of the attribute that stores the resource name to be seized. Expression Expression that evaluates to a resource name to be seized. Quantity Number of resources of a given name or from a given set that will be seized. For sets, this value specifies only the number of a selected resource that will be seized (based on the resource's capacity), not the number of members to be seized within the set. Selection Rule Method of selecting among available resources in a set. Cyclical will cycle through available members (for example, 1-2-3-1-2- 3). Random will randomly select a member. Preferred Order will always select the first available member (for example, 1, if available; then 2, if available; then 3). Specific Member requires an input attribute value to specify which member of the set (previously saved in the Save **Attribute** field). Largest Remaining Capacity and Smallest Number Busy are used for resources with multiple capacity. Save **Attribute** Attribute name used to store the index number into the set of the member that is chosen. This attribute can later be referenced with the Specific Member selection rule. Set Index Index value into the set that identifies the number into the set of the member requested. If an attribute name is used, the entity must have a value for the attribute before utilizing this option. Resource State State of the resource that will be assigned after the resource is seized. The resource state must be defined with the **Resource** module. Queue Type Determines the type of queue used to hold the entities while waiting to seize the resource(s). If Queue is selected, the queue name is specified. If Set is selected, the queue set and member in the set are specified. If Internal is selected, an internal queue is used to hold all waiting entities. Attribute and Expression are additional methods for defining the queue to be used. Queue Name This field is visible only if Queue Type is Queue, and it defines the symbol name of the queue. Set Name This field is visible only if Queue Type is Set, and it defines the queue set that contains the queue being referenced. Set Index This field is visible only if Queue Type is Set, and it defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entries for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute This field is visible only if Queue Type is Attribute. The attribute entered in this field will be evaluated to indicate which queue is to be used. Expression This field is visible only if Queue Type is Expression. The expression entered in this field will be evaluated to indicate which queue is to be used.

Definition at line 125 of file **Seize.h**.

8.115.2 Constructor & Destructor Documentation

8.115.2.1 Seize() Seize::Seize (

```
    Model * model,
    std::string name = "" )
```

Definition at line 19 of file [Seize.cpp](#).

```
00019 : ModelComponent(model, Util::TypeOf<Seize>(), name) {
00020 }
```

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

Here is the caller graph for this function:



8.115.2.2 ~Seize() virtual Seize::~Seize () [virtual], [default]

8.115.3 Member Function Documentation

8.115.3.1 _check() bool Seize::_check (

```
    std::string * errorMessage ) [protected], [virtual]
```

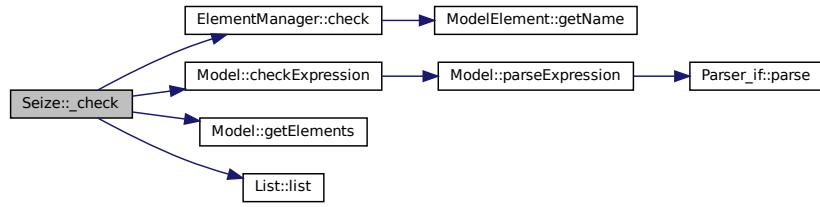
Reimplemented from [ModelElement](#).

Definition at line 196 of file [Seize.cpp](#).

```
00196                                     {
00197     bool resultAll = true;
00198     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++) {
00199         resultAll &= _parentModel->checkExpression((*it)->getQuantityExpression(), "quantity",
00200             errorMessage);
00201         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Resource>(),
00202             (*it)->getResource(), "Resource", errorMessage);
00203     }
00204     resultAll &= _parentModel->getElements()->check(Util::TypeOf<Queue>(),
00205         _queue, "Queue",
00206         errorMessage);
00207     // \todo implement check of each ResourceItemRequest
00208     //resultAll &= _parentModel->getElements()->check(Util::TypeOf<Attribute>(),
00209     //    _saveAttribute,
00210     //    "SaveAttribute", false, errorMessage);
00211     return resultAll;
00212 }
```

References [ModelElement::_parentModel](#), [ElementManager::check\(\)](#), [Model::checkExpression\(\)](#), [Model::getElements\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.115.3.2 `_execute()` void Seize::_execute (
 Entity * entity) [protected], [virtual]

Implements [ModelComponent](#).

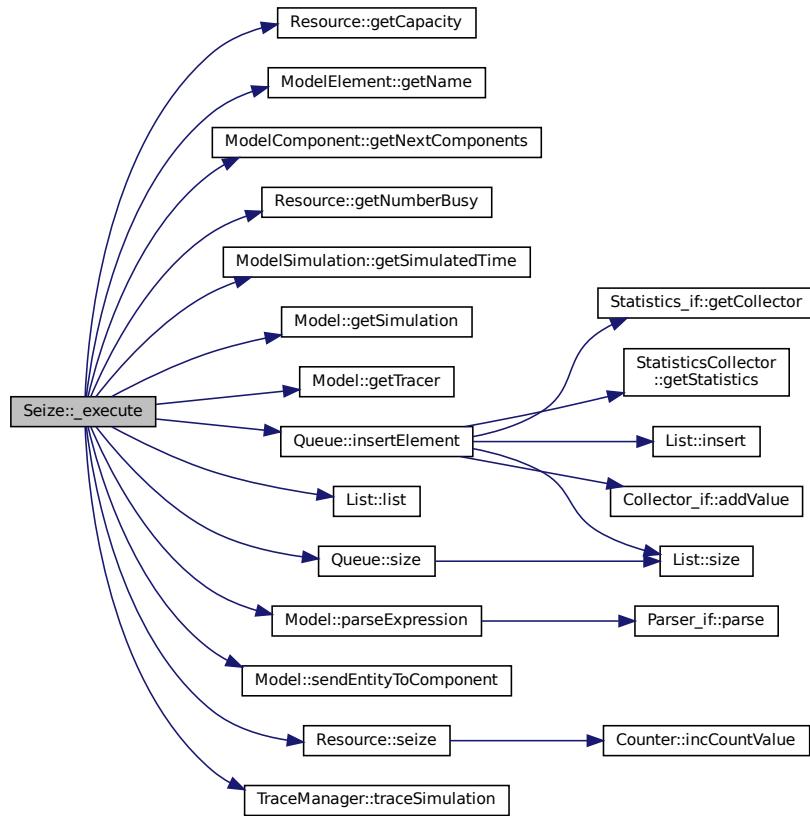
Definition at line 116 of file [Seize.cpp](#).

```

0016      {
0017      for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != 
0018      _seizeRequests->list()->end(); it++) {
0019          Resource* resource = (*it)->getResource();
0020          unsigned int quantity = _parentModel->parseExpression((*it)->getQuantityExpression());
0021          if (resource->getCapacity() - resource->getNumberBusy() < quantity) { // not enough free
0022              quantity to allocate. Entity goes to the queue
0023              WaitingResource* waitingRec = new WaitingResource(entity, this,
0024              _parentModel->getSimulation()->getSimulatedTime(), quantity);
0025              this->_queue->insertElement(waitingRec); // ->list()->insert(waitingRec);
0026
0027              _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(), entity,
0028              this, "Entity starts to wait for resource in queue \'" + _queue->getName() + "\' with " +
0029              std::to_string(_queue->size()) + " elements";
0030              return;
0031          } else { // alocate the resource
0032              resource->seize(quantity, _parentModel->getSimulation()->getSimulatedTime());
0033
0034              _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(), entity,
0035              this, "Entity seizes " + std::to_string(quantity) + " elements of resource \'" + resource->getName() +
0036              "\' (capacity:" + std::to_string(resource->getCapacity()) + ", numberbusy:" +
0037              std::to_string(resource->getNumberBusy()) + ")");
0038          }
0039      }
0040      _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
0041  
```

References [ModelElement::_parentModel](#), [Resource::getCapacity\(\)](#), [ModelElement::getName\(\)](#), [ModelComponent::getNextComponent\(\)](#), [Resource::getNumberBusy\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [Queue::insertElement\(\)](#), [List< T >::list\(\)](#), [Model::parseExpression\(\)](#), [Resource::seize\(\)](#), [Model::sendEntityToComponent\(\)](#), [Queue::size\(\)](#), and [TraceManager::traceSimulation\(\)](#).

Here is the call graph for this function:



8.115.3.3 `_initBetweenReplications()` void Seize::_initBetweenReplications () [protected], [virtual]

Reimplemented from [ModelElement](#).

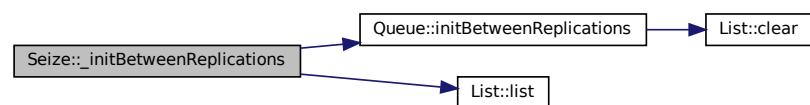
Definition at line 133 of file [Seize.cpp](#).

```

00133     {
00134         this->_queue->initBetweenReplications\(\);
00135         for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list\(\)->begin\(\); it != _seizeRequests->list\(\)->end\(\); it++) {
00136             (*it)->setLastMemberSeized(0);
00137             (*it)->getResource()->initBetweenReplications\(\);
00138         }
00139     }
  
```

References [Queue::initBetweenReplications\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



```
8.115.3.4 _loadInstance() bool Seize::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 141 of file [Seize.cpp](#).

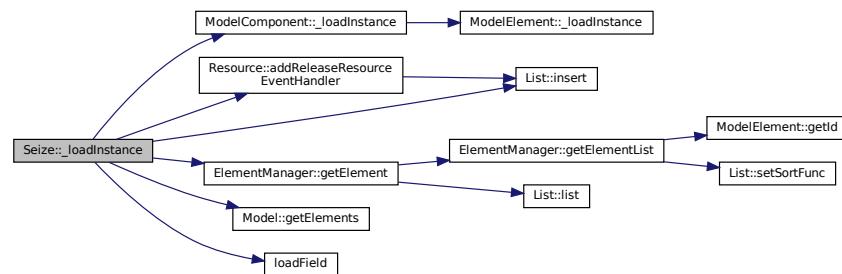
```
00141
00142     bool res = ModelComponent::_loadInstance(fields);
00143     if (res) {
00144         this->_allocationType = std::stoi(loadField(fields, "allocationType", "0"));
00145         this->_priority = std::stoi(loadField(fields, "priority", "0"));
00146         //Util::identitifcation queueId = std::stoi((*(fields->find("queueId"))).second);
00147         //Queue* queue = dynamic_cast<Queue*> (_model->elements()->element(Util::TypeOf<Queue>(),
00148             queueId));
00149         std::string queueName = ((*(fields->find("queueName"))).second);
00150         Queue* queue = dynamic_cast<Queue*>
00151             (_parentModel->getElements()->getElement(Util::TypeOf<Queue>(), queueName));
00152         this->_queue = queue;
00153         //Util::identitifcation resourceId = std::stoi((*(fields->find("resourceId"))).second);
00154         //Resource* resource = dynamic_cast<Resource*>
00155             (_model->elements()->element(Util::TypeOf<Resource>(), resourceId));
00156
00157         // \todo: next fields form a pair, and it should be a list of them
00158         unsigned short numRequests = std::stoi((*(fields->find("seizeResquestSize"))).second);
00159         for (unsigned short i = 0; i < numRequests; i++) {
00160             //std::string resRequest = ((*(fields->find("resourceItemRequest"))).second);
00161             SeizableItemRequest::ResourceType resourceType =
00162                 static_cast<SeizableItemRequest::ResourceType> (std::stoi(loadField(fields, "resourceType" +
00163                     std::to_string(i), std::to_string(static_cast<int> (SeizableItemRequest::ResourceType::RESOURCE))));
```

~~00164 std::string resourceName = ((*(fields->find("resourceName" + std::to_string(i)))).second);~~
~~00165 Resource* resource = dynamic_cast<Resource*>~~
~~00166 (_parentModel->getElements()->getElement(Util::TypeOf<Resource>(), resourceName));~~
~~00167 std::string quantityExpression = loadField(fields, "quantity" + std::to_string(i), "1");~~
~~00168 SeizableItemRequest::SelectionRule rule = static_cast<SeizableItemRequest::SelectionRule>~~
~~00169 (std::stoi(loadField(fields, "selectionRule" + std::to_string(i), std::to_string(static_cast<int> (SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY))));~~
~~00170 std::string saveAttribute = loadField(fields, "saveAttribute" + std::to_string(i), "");~~
~~00171 unsigned int index = std::stoi(loadField(fields, "index" + std::to_string(i), "0"));~~
~~00172 this->_seizeRequests->insert(new SeizableItemRequest(resource, quantityExpression,~~
~~resourceType, rule, saveAttribute, index));~~
~~00173 resource->addReleaseResourceEventHandler(Resource::SetResourceEventHandler<Seize>(&Seize::_handlerForResourceEvent,~~
~~this));~~
~~00174 }~~
~~00175 return res;~~
00176 }

References [ModelComponent::_loadInstance\(\)](#), [ModelElement::_parentModel](#), [Resource::addReleaseResourceEventHandler\(\)](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), [List< T >::insert\(\)](#), [SeizableItemRequest::LARGESTREMAININGCAPACITY](#), [loadField\(\)](#), and [SeizableItemRequest::RESOURCE](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.115.3.5 `_saveInstance()` `std::map< std::string, std::string > * Seize::_saveInstance ()`
[protected], [virtual]

Reimplemented from [ModelComponent](#).

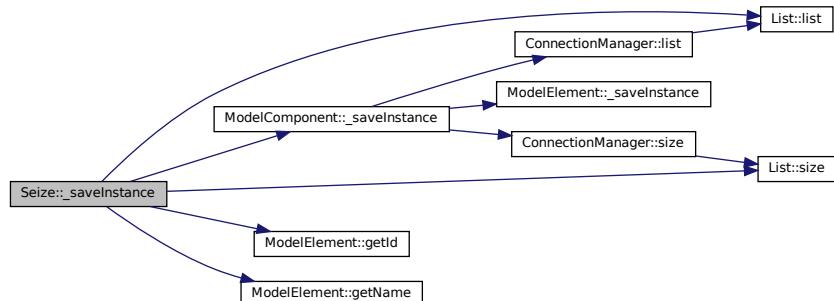
Definition at line 173 of file [Seize.cpp](#).

```

00173
00174     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00175     //Util::TypeOf<Seize>());
00176     if (_allocationType != 0) fields->emplace("allocationType",
00177         std::to_string(this->_allocationType));
00178     if (_priority != 0) fields->emplace("priority=", std::to_string(this->_priority));
00179     fields->emplace("queueId", std::to_string(this->_queue->getId()));
00180     fields->emplace("queueName", "\" " + (this->_queue->getName()) + "\"");
00181     // \todo: put together as ResourceItemRequest ans it should be a list of them
00182     //
00183     fields->emplace("seizeResquestSize", std::to_string(_seizeRequests->size()));
00184     unsigned short i = 0;
00185     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++, i++) {
00186         //fields->emplace("resourceItemRequest" + std::to_string(i), "{"
00187         map2str((*it)->_saveInstance()) + "}");
00188         if ((*it)->getResourceType() != SeizableItemRequest::ResourceType::RESOURCE)
00189             fields->emplace("resourceType" + std::to_string(i), std::to_string(static_cast<int>((*it)->getResourceType())));
00190         fields->emplace("resourceId" + std::to_string(i),
00191             std::to_string((*it)->getResource()->getId()));
00192         fields->emplace("resourceName" + std::to_string(i), ((*it)->getResource()->getName()));
00193         if ((*it)->getQuantityExpression() != "1") fields->emplace("quantity" + std::to_string(i),
00194             "\" " + (*it)->getQuantityExpression() + "\"");
00195         if ((*it)->getSelectionRule() != SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)
00196             fields->emplace("selectionRule" + std::to_string(i), std::to_string(static_cast<int>((*it)->getSelectionRule())));
00197             if ((*it)->getSaveAttribute() != "") fields->emplace("saveAttribute" + std::to_string(i), "\" "
00198             + (*it)->getSaveAttribute() + "\"");
00199             fields->emplace("index" + std::to_string(i), std::to_string((*it)->getIndex()));
00200     }
00201     return fields;
00202 }
```

References [ModelComponent::_saveInstance\(\)](#), [ModelElement::getId\(\)](#), [ModelElement::getName\(\)](#), [SeizableItemRequest::LARGESTREMAININGCAPACITY](#), [List< T >::list\(\)](#), [SeizableItemRequest::RESOURCE](#), and [List< T >::size\(\)](#).

Here is the call graph for this function:



8.115.3.6 getAllocationType() `unsigned int Seize::getAllocationType () const`

Definition at line 46 of file [Seize.cpp](#).

```
00046             {
00047     return _allocationType;
00048 }
```

8.115.3.7 GetPluginInformation() `PluginInformation * Seize::GetPluginInformation () [static]`

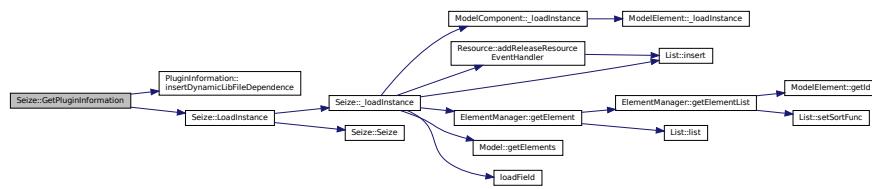
Definition at line 208 of file [Seize.cpp](#).

```
00208     {
00209     PluginInformation* info = new PluginInformation(Util::TypeOf<Seize>(), &Seize::LoadInstance);
00210     info->insertDynamicLibFileDependence("queue.so");
00211     info->insertDynamicLibFileDependence("resource.so");
00212     return info;
00213 }
```

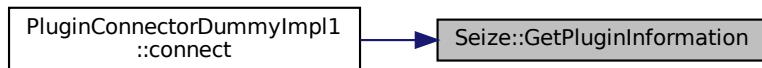
References [PluginInformation::insertDynamicLibFileDependence\(\)](#), and [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.115.3.8 getPriority()** `unsigned short Seize::getPriority () const`

Definition at line 38 of file [Seize.cpp](#).

```
00038             {
00039     return _priority;
00040 }
```

8.115.3.9 getQueue() Queue * Seize::getQueue () const

Definition at line 108 of file [Seize.cpp](#).

```
00108     {  
00109         return _queue;  
00110     }
```

8.115.3.10 getQueueName() std::string Seize::getQueueName () const

Definition at line 91 of file [Seize.cpp](#).

```
00091     {  
00092         return _queue->getName ();  
00093     }
```

References [ModelElement::getName\(\)](#).

Here is the call graph for this function:

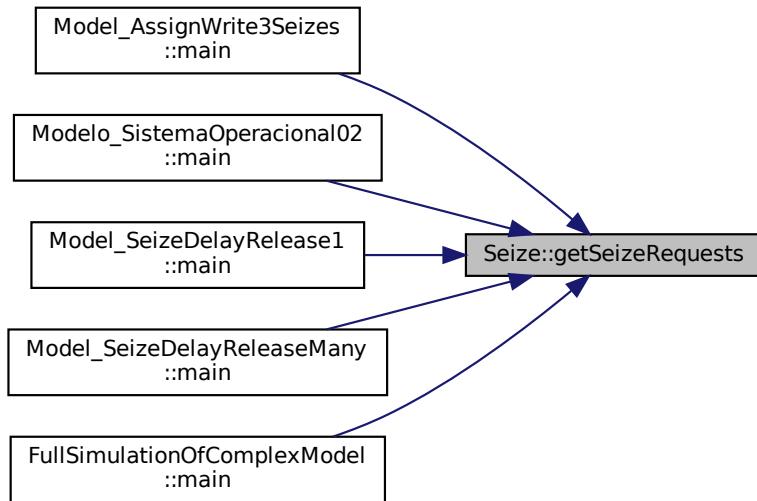
**8.115.3.11 getSeizeRequests()** List< SeizableItemRequest * > * Seize::getSeizeRequests () const

Definition at line 112 of file [Seize.cpp](#).

```
00112     {  
00113         return _seizeRequests;  
00114     }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.115.3.12 LoadInstance() ModelComponent * Seize::LoadInstance (

```

    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 215 of file [Seize.cpp](#).

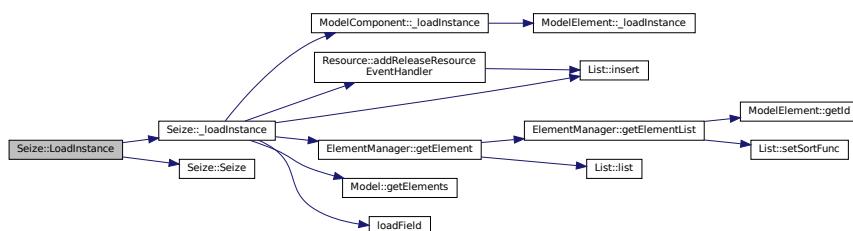
```

00215
00216     Seize* newComponent = new Seize(model);
00217     try {
00218         newComponent->loadInstance(fields);
00219     } catch (const std::exception& e) {
00220
00221     }
00222     return newComponent;
00223
00224 }
```

References [_loadInstance\(\)](#), and [Seize\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.115.3.13 setAllocationType() void Seize::setAllocationType (unsigned int _allocationType)

Definition at line 42 of file [Seize.cpp](#).

```
00042 {  
00043     this->_allocationType = _allocationType;  
00044 }
```

8.115.3.14 setPriority() void Seize::setPriority (unsigned short _priority)

Definition at line 34 of file [Seize.cpp](#).

```
00034 {  
00035     this->_priority = _priority;  
00036 }
```

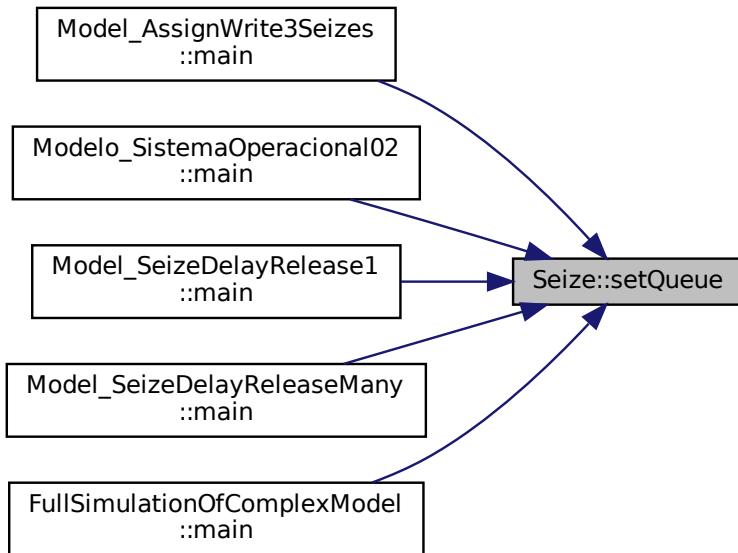
8.115.3.15 setQueue() void Seize::setQueue (Queue * queue)

Definition at line 104 of file [Seize.cpp](#).

```
00104 {  
00105     this->_queue = queue;  
00106 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.115.3.16 setQueueName() void Seize::setQueueName (std::string queueName) throw ()

Definition at line 50 of file [Seize.cpp](#).

```

00050     Queue* queue = dynamic_cast<Queue*>
00051         (_parentModel->getElements()->getElement(Util::TypeOf<Queue>(), queueName));
00052     if (queue != nullptr) {
00053         _queue = queue;
00054     } else {
00055         throw std::invalid_argument("Queue does not exist");
00056     }
00057 }
```

8.115.3.17 show() std::string Seize::show () [virtual]

Reimplemented from [ModelComponent](#).

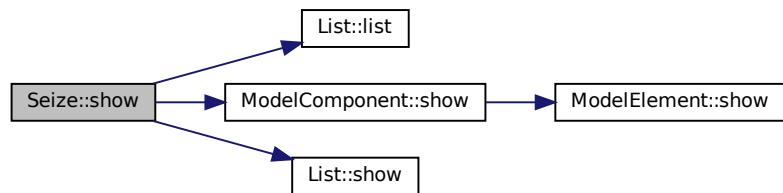
Definition at line 22 of file [Seize.cpp](#).

```

00022     {
00023         std::string txt = ModelComponent::show() +
00024             "priority=" + std::to_string(_priority) +
00025             "seizeRequests=\"";
00026         unsigned short i = 0;
00027         for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++, i++) {
00028             txt += "request" + std::to_string(i) + "[" + _seizeRequests->show() + "]";
00029         }
00030         txt = txt.substr(0, txt.length() - 1) + "}";
00031         return txt;
00032     }
```

References [List< T >::list\(\)](#), [ModelComponent::show\(\)](#), and [List< T >::show\(\)](#).

Here is the call graph for this function:



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

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

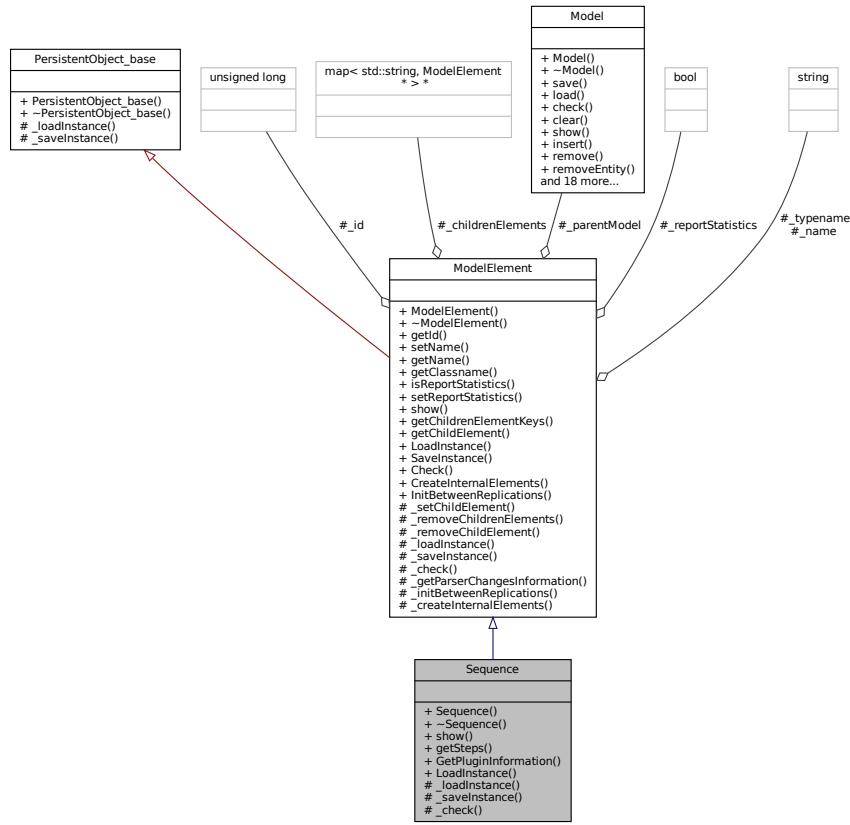
8.116 Sequence Class Reference

```
#include <Sequence.h>
```

Inheritance diagram for Sequence:



Collaboration diagram for Sequence:



Public Member Functions

- **Sequence (Model *model, std::string name="")**
- virtual **~Sequence ()=default**
- virtual std::string **show ()**
- **List< SequenceStep * > * getSteps () const**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.116.1 Detailed Description

Sequence module DESCRIPTION The **Sequence** module is used to define a sequence for entity flow through the model. A sequence consists of an ordered list of stations that an entity will visit. For each station in the visitation sequence, attributes and variables may be assigned values. Each station in the visitation sequence is referred to as a step (or jobstep) in the sequence. Three special-purpose attributes are provided for all entities. The **Sequence** attribute (`Entity.Sequence`) defines the sequence that an entity is to follow; a value of 0 indicates that the entity is not following any sequence. In order for an entity to follow a sequence, its **Sequence** attribute must be assigned a value (for example, in the **Assign** module). The **Jobstep** attribute (`Entity.Jobstep`) stores the entity's current step number in the sequence. This value is updated automatically each time an entity is transferred. You typically do not need to assign explicitly a value to **Jobstep** in the model. The **PlannedStation** attribute (`Entity.PlannedStation`) stores the number of the station associated with the next jobstep in the sequence. This attribute is not user assignable. It is automatically updated whenever `Entity.Sequence` or `Entity.JobStep` changes, or whenever the entity enters a station. Jobstep names must be globally unique. TYPICAL USES Define a routing path for part processing Define a sequence of steps patients must take upon arrival at an emergency room

Definition at line 74 of file [Sequence.h](#).

8.116.2 Constructor & Destructor Documentation

```
8.116.2.1 Sequence() Sequence::Sequence (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Sequence.cpp](#).

```
00018 : ModelElement(model, Util::TypeOf<Sequence>(), name) {  
00019 }
```

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

Here is the caller graph for this function:



```
8.116.2.2 ~Sequence() virtual Sequence::~Sequence ( ) [virtual], [default]
```

8.116.3 Member Function Documentation

```
8.116.3.1 _check() bool Sequence::_check (
    std::string * errorMessage ) [protected], [virtual]
```

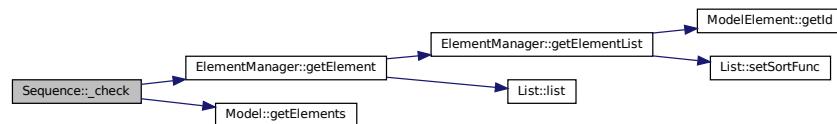
Reimplemented from [ModelElement](#).

Definition at line 60 of file [Sequence.cpp](#).

```
00060                                     {
00061     /* include attributes needed */
00062     std::vector<std::string> neededNames = {"Entity.Sequence", "Entity.SequenceStep"};
00063     std::string neededName;
00064     for (unsigned int i = 0; i < neededNames.size(); i++) {
00065         neededName = neededNames[i];
00066         if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr)
00067         {
00068             Attribute* attr1 = new Attribute(_parentModel, neededName);
00069             // _parentModel->insert(attr1);
00070         }
00071     }
00072     //
00073     return true;
00074 }
```

References [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



```
8.116.3.2 _loadInstance() bool Sequence::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 45 of file [Sequence.cpp](#).

```
00045                                     {
00046     bool res = ModelElement::_loadInstance(fields);
00047     if (res) {
00048         try {
00049             } catch (...) {
00050             }
00051     }
00052     return res;
00053 }
```

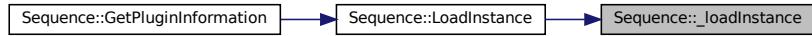
References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.116.3.3 `_saveInstance()` `std::map< std::string, std::string > * Sequence::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

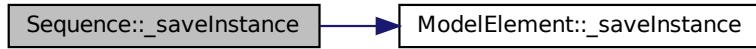
Definition at line 55 of file [Sequence.cpp](#).

```

00055
00056     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00057     //Util::TypeOf<Sequence>());
00058     return fields;
  
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.116.3.4 `GetPluginInformation()` `PluginInformation * Sequence::GetPluginInformation () [static]`

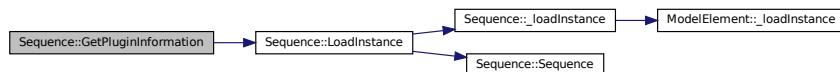
Definition at line 26 of file [Sequence.cpp](#).

```

00026
00027     PluginInformation* info = new PluginInformation(Util::TypeOf<Sequence>(),
00028     &Sequence::LoadInstance);
00029     return info;
  
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



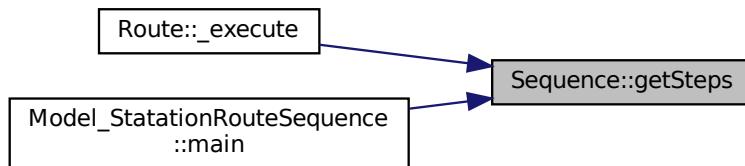
8.116.3.5 `getSteps()` `List< SequenceStep * > * Sequence::getSteps() const`

Definition at line 41 of file [Sequence.cpp](#).

```
00041
00042     return _steps;
00043 }
```

Referenced by [Route::_execute\(\)](#), and [Model_StationRouteSequence::main\(\)](#).

Here is the caller graph for this function:



8.116.3.6 `LoadInstance()` `ModelElement * Sequence::LoadInstance(`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

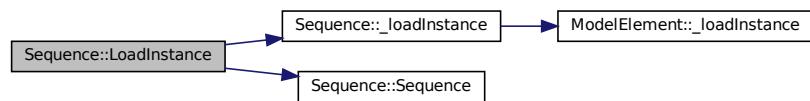
Definition at line 31 of file [Sequence.cpp](#).

```
00031
00032     Sequence* newElement = new Sequence(model);
00033     try {
00034         newElement->_loadInstance(fields);
00035     } catch (const std::exception& e) {
00036     }
00037 }
00038     return newElement;
00039 }
```

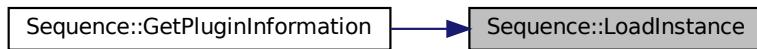
References [_loadInstance\(\)](#), and [Sequence\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.116.3.7 show() std::string Sequence::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 21 of file [Sequence.cpp](#).

```
00021     {
00022         std::string msg = ModelElement::show();
00023         return msg;
00024     }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



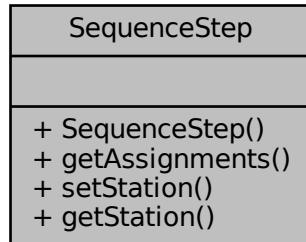
The documentation for this class was generated from the following files:

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

8.117 SequenceStep Class Reference

```
#include <Sequence.h>
```

Collaboration diagram for SequenceStep:



Public Member Functions

- `SequenceStep (Station *station, std::list< std::string > *assignments=nullptr)`
- `std::list< std::string > * getAssignments () const`
- `void setStation (Station *_station)`
- `Station * getStation () const`

8.117.1 Detailed Description

Definition at line 22 of file [Sequence.h](#).

8.117.2 Constructor & Destructor Documentation

8.117.2.1 SequenceStep()

```
SequenceStep::SequenceStep (
    Station * station,
    std::list< std::string > * assignments = nullptr ) [inline]
```

Definition at line 25 of file [Sequence.h](#).

```
00025
00026     _station = station;
00027     if (assignments == nullptr)
00028         _assignments = new std::list<std::string>();
00029     else
00030         _assignments = assignments;
00031 }
```

8.117.3 Member Function Documentation

8.117.3.1 getAssignments() `std::list<std::string>* SequenceStep::getAssignments () const [inline]`

Definition at line 33 of file [Sequence.h](#).

```
00033
00034     return _assignments;
00035 }
```

Referenced by [Route::_execute\(\)](#).

Here is the caller graph for this function:



8.117.3.2 getStation() `Station* SequenceStep::getStation () const [inline]`

Definition at line 41 of file [Sequence.h](#).

```
00041
00042     return _station;
00043 }
```

Referenced by [Route::_execute\(\)](#).

Here is the caller graph for this function:



8.117.3.3 setStation() `void SequenceStep::setStation (Station * _station) [inline]`

Definition at line 37 of file [Sequence.h](#).

```
00037
00038     this->_station = _station;
00039 }
```

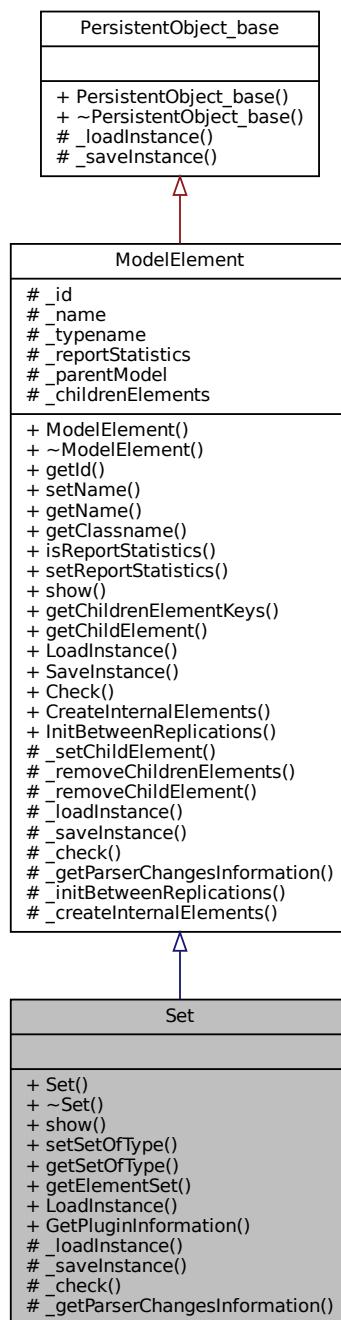
The documentation for this class was generated from the following file:

- [Sequence.h](#)

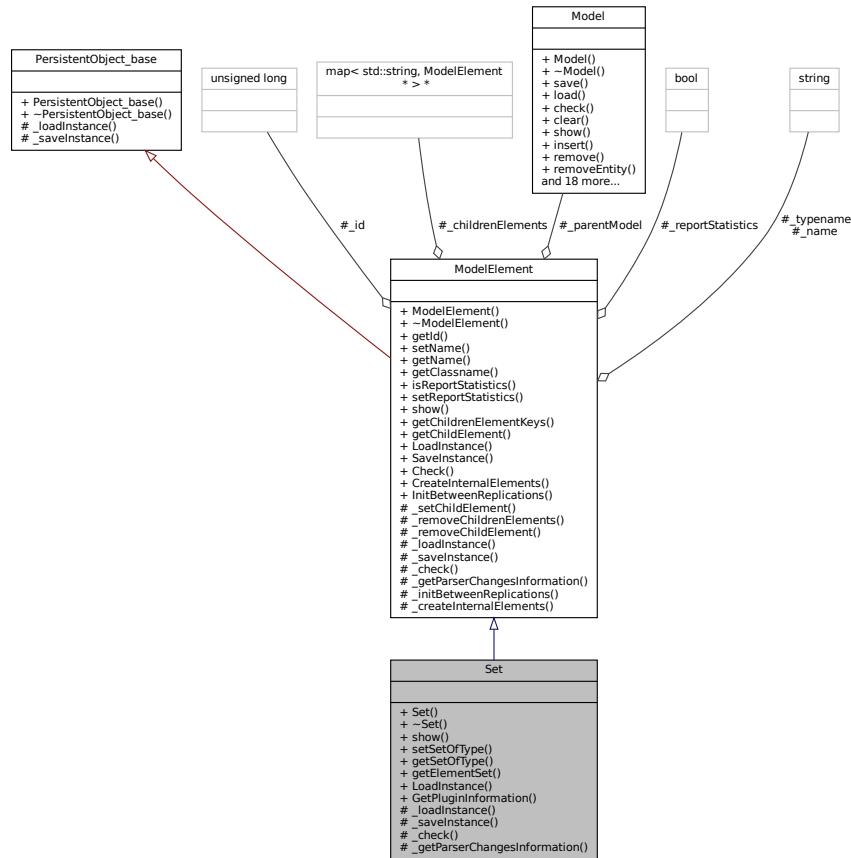
8.118 Set Class Reference

```
#include <Set.h>
```

Inheritance diagram for Set:



Collaboration diagram for Set:



Public Member Functions

- `Set (Model *model, std::string name="")`
- virtual `~Set ()=default`
- virtual `std::string show ()`
- void `setSetOfType (std::string _setOfType)`
- `std::string getSetOfType () const`
- `List< ModelElement * > * getElementSet () const`

Static Public Member Functions

- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`
- static `PluginInformation * GetPluginInformation ()`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual `ParserChangesInformation * _getParserChangesInformation ()`

Additional Inherited Members

8.118.1 Detailed Description

Set module DESCRIPTION This data module defines various types of sets, including resource, counter, tally, entity type, and entity picture. **Resource** sets can be used in the Process modules (and **Seize**, **Release**, **Enter**, and **Leave** of the Advanced Process and Advanced Transfer panels). **Counter** and **Tally** sets can be used in the **Record** module. **Queue** sets can be used with the **Seize**, **Hold**, **Access**, **Request**, **Leave**, and **Allocate** modules of the Advanced Process and Advanced Transfer panels.

TYPICAL USES Machines that can perform the same operations in a manufacturing facility Supervisors, check-out clerks in a store Shipping clerks, receptionists in an office **Set** of pictures corresponding to a set of entity types **PROMPTS** Prompt Description Name The unique name of the set being defined. **Type** Type of set being defined. **Members** Repeat group that specifies the resource members with the set. The order of listing the members within the repeat group is important when using selection rules such as Preferred Order and Cyclical. **Resource** Name Name of the resource to include in the resource set. Applies only when Type is **Resource**. **Tally** Name Name of the tally within the tally set. Applies only when Type is **Tally**. **Counter** Name Name of the counter within the counter set. Applies only when Type is **Counter**. **Entity** Type Name of the entity type within the entity type set. Applies only when Type is **Entity**. **Picture** Name Name of the picture within the picture set. Applies only when Type is **Entity** Picture.

Definition at line 55 of file [Set.h](#).

8.118.2 Constructor & Destructor Documentation

```
8.118.2.1 Set() Set::Set (
    Model * model,
    std::string name = "")
```

Definition at line 16 of file [Set.cpp](#).

```
00016 : ModelElement(model, Util::TypeOf<Set>(), name) {  
00017 }
```

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

Here is the caller graph for this function:



```
8.118.2.2 ~Set() virtual Set::~Set ( ) [virtual], [default]
```

8.118.3 Member Function Documentation

```
8.118.3.1 _check() bool Set::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 70 of file [Set.cpp](#).

```
00070
00071     bool resultAll = true;
00072     // resultAll |= ...
00073     return resultAll;
00074 }
```

```
8.118.3.2 _getParserChangesInformation() ParserChangesInformation * Set::_getParserChangesInformation () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 76 of file [Set.cpp](#).

```
00076
00077     ParserChangesInformation* changes = new ParserChangesInformation();
00078     //changes->getProductionToAdd()->insert(...);
00079     //changes->getTokensToAdd()->insert(...);
00080     return changes;
00081 }
```

```
8.118.3.3 _loadInstance() bool Set::_loadInstance (
```

```
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 51 of file [Set.cpp](#).

```
00051
00052     bool res = ModelElement::_loadInstance(fields);
00053     if (res) {
00054         try {
00055             //this->_attributeName = (*fields->find("attributeName")).second;
00056             //this->_orderRule = static_cast<OrderRule>
00057             (std::stoi((*fields->find("orderRule")).second));
00058         } catch (...) {
00059     }
00060     return res;
00061 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.118.3.4 `_saveInstance()` `std::map< std::string, std::string > * Set::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 63 of file [Set.cpp](#).

```
00063
00064     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00065     //Util::TypeOf<Set>());
00066     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00067     //fields->emplace("attributeName", "\"" +this->_attributeName+"\"");
00068 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.118.3.5 `getElementSet()` `List< ModelElement * > * Set::getElementSet () const`

Definition at line 32 of file [Set.cpp](#).

```
00032
00033     return _elementSet;
00034 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#).

Here is the caller graph for this function:



8.118.3.6 GetPluginInformation() `PluginInformation * Set::GetPluginInformation () [static]`

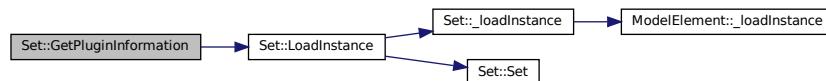
Definition at line 36 of file [Set.cpp](#).

```
00036                                     {
00037     PluginInformation* info = new PluginInformation(Util::TypeOf<Set>(), &Set::LoadInstance);
00038     return info;
00039 }
```

References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.118.3.7 getSetOfType()** `std::string Set::getSetOfType () const`

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

```
00028                                     {
00029     return _setOfType;
00030 }
```

8.118.3.8 LoadInstance() `ModelElement * Set::LoadInstance (`

```
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

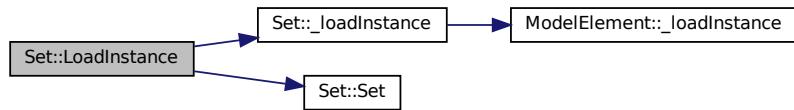
Definition at line 41 of file [Set.cpp](#).

```
00041                                     {
00042     Set* newElement = new Set(model);
00043     try {
00044         newElement->_loadInstance(fields);
00045     } catch (const std::exception& e) {
00046     }
00047     return newElement;
00048 }
00049 }
```

References [_loadInstance\(\)](#), and [Set\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



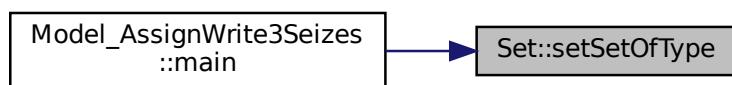
8.118.3.9 setSetOfType() `void Set::setSetOfType (std::string _setOfType)`

Definition at line 24 of file [Set.cpp](#).

```
00024     {  
00025         this->_setOfType = _setOfType;  
00026     }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#).

Here is the caller graph for this function:



8.118.3.10 show() std::string Set::show() [virtual]

Reimplemented from [ModelElement](#).

Definition at line 19 of file [Set.cpp](#).

```
00019     {
00020     return ModelElement::show() +
00021           "";
00022 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



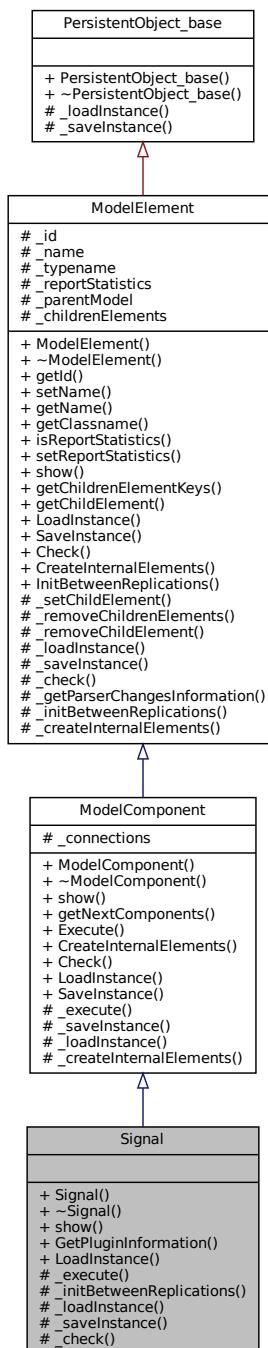
The documentation for this class was generated from the following files:

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

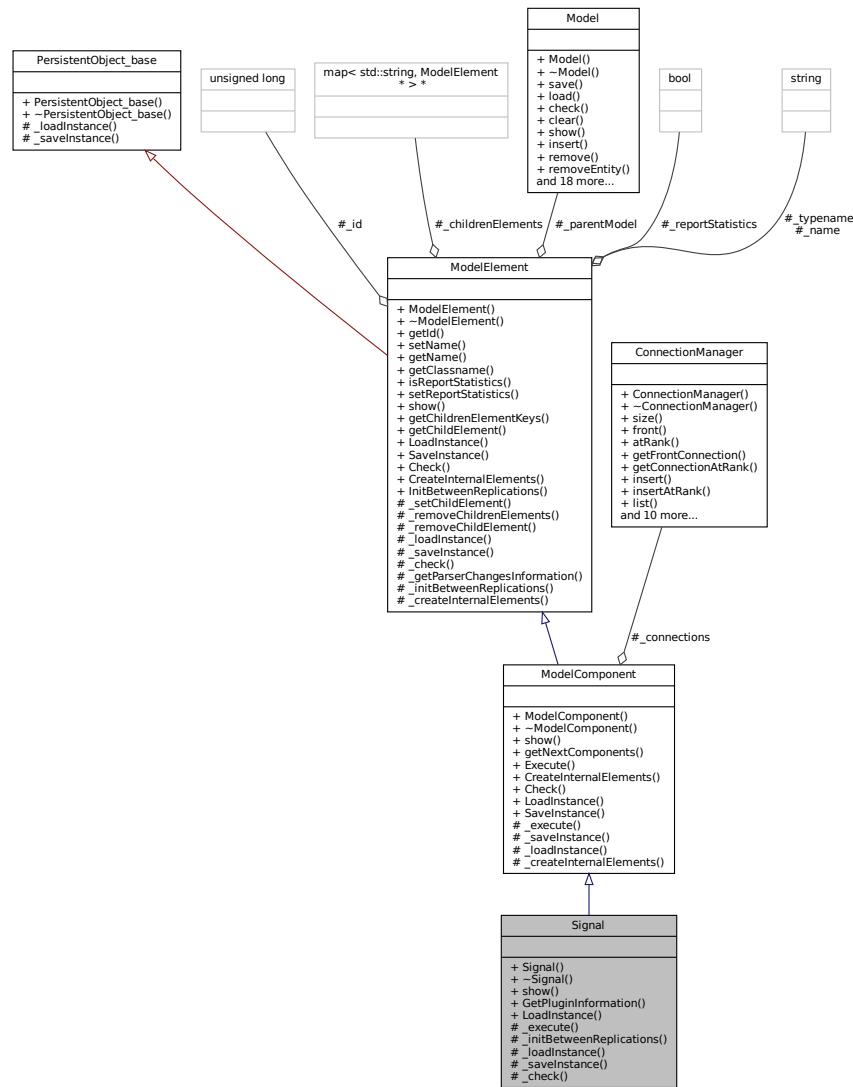
8.119 Signal Class Reference

```
#include <Signal.h>
```

Inheritance diagram for Signal:



Collaboration diagram for Signal:



Public Member Functions

- **Signal** (Model *model, std::string name="")
 - virtual ~**Signal** ()=default
 - virtual std::string **show** ()

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
 - static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void _execute (Entity *entity)
 - virtual void _initBetweenReplications ()
 - virtual bool _loadInstance (std::map< std::string, std::string > *fields)
 - virtual std::map< std::string, std::string > * _saveInstance ()
 - virtual bool check (std::string *errorMessage)

Additional Inherited Members

8.119.1 Detailed Description

Signal module DESCRIPTION The **Signal** module sends a signal value to each **Hold** module in the model set to Wait for **Signal** and releases the maximum specified number of entities. When an entity arrives at a **Signal** module, the signal is evaluated and the signal code is sent. At this time, entities at **Hold** modules that are waiting for the same signal are removed from their queues. The entity sending the signal continues processing until it encounters a delay, enters a queue, or is disposed. TYPICAL USES Analyzing traffic patterns at an intersection (signal when the light turns green) Signaling an operator to complete an order that was waiting for a component part PROMPT TS Prompt Description Name Unique module identifier displayed on the module shape. **Signal** Value Value of the signal to be sent to entities in **Hold** modules. Limit Maximum number of entities that are to be released from any **Hold** modules when the signal is received.

Definition at line 38 of file [Signal.h](#).

8.119.2 Constructor & Destructor Documentation

```
8.119.2.1 Signal() Signal::Signal (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file [Signal.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Signal>(), name) {
00019 }
```

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

Here is the caller graph for this function:



```
8.119.2.2 ~Signal() virtual Signal::~Signal ( ) [virtual], [default]
```

8.119.3 Member Function Documentation

```
8.119.3.1 _check() bool Signal::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Signal.cpp](#).

```
00057
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
```

```
8.119.3.2 _execute() void Signal::_execute (
    Entity * entity ) [protected], [virtual]
```

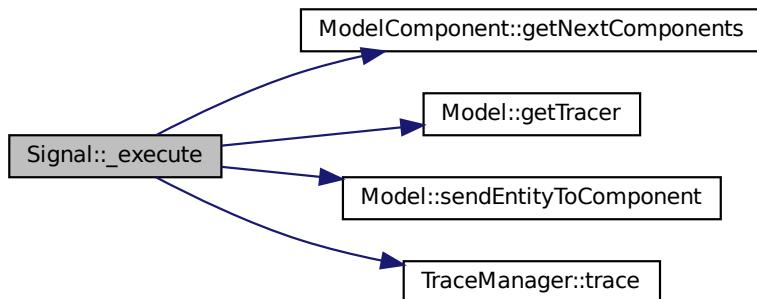
Implements [ModelComponent](#).

Definition at line 35 of file [Signal.cpp](#).

```
00035
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00038     0.0);
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent\(\)](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



```
8.119.3.3 _initBetweenReplications() void Signal::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Signal.cpp](#).

```
00048
00049 }
```

```
8.119.3.4 _loadInstance() bool Signal::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

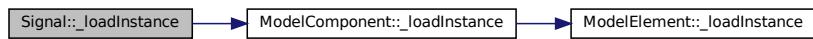
Definition at line 40 of file [Signal.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

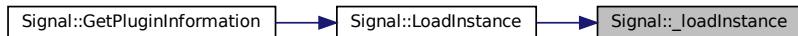
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.119.3.5 _saveInstance() std::map< std::string, std::string > * Signal::_saveInstance ( )
[protected], [virtual]
```

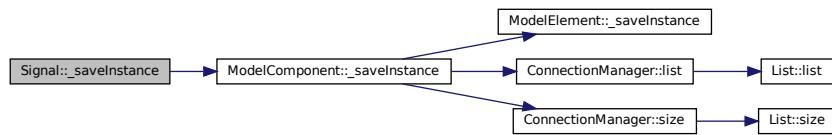
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Signal.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



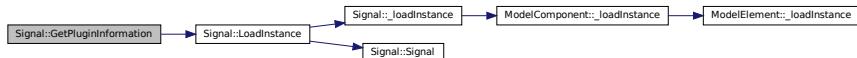
8.119.3.6 GetPluginInformation() `PluginInformation * Signal::GetPluginInformation () [static]`

Definition at line 63 of file [Signal.cpp](#).

```
00063 {  
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Signal>(), &Signal::LoadInstance);  
00065     // ...  
00066     return info;  
00067 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.119.3.7 LoadInstance() `ModelComponent * Signal::LoadInstance (`

```
Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

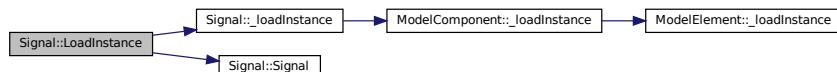
Definition at line 25 of file [Signal.cpp](#).

```
00025 {  
00026     Signal* newComponent = new Signal(model);  
00027     try {  
00028         newComponent->_loadInstance(fields);  
00029     } catch (const std::exception& e) {  
00030     }  
00031     return newComponent;  
00032 }  
00033 }
```

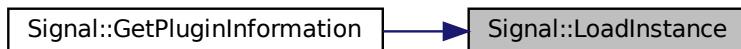
References [_loadInstance\(\)](#), and [Signal\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.119.3.8 `show()` `std::string Signal::show() [virtual]`

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Signal.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



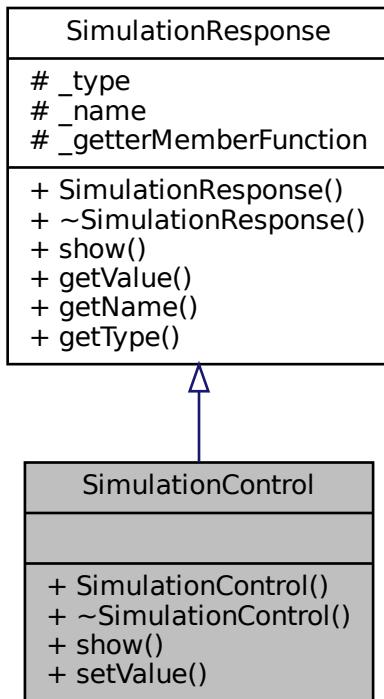
The documentation for this class was generated from the following files:

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

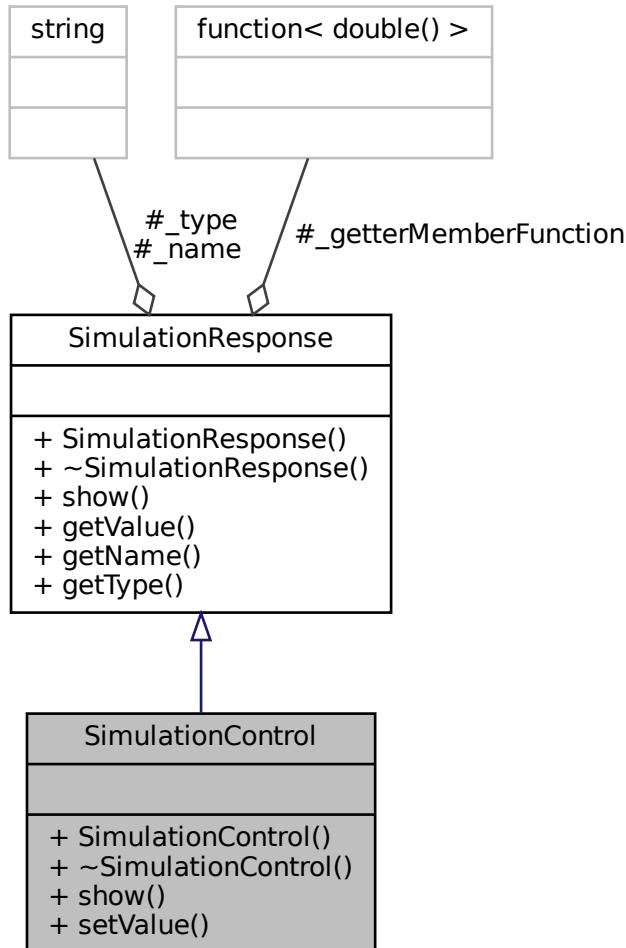
8.120 SimulationControl Class Reference

```
#include <SimulationControl.h>
```

Inheritance diagram for SimulationControl:



Collaboration diagram for SimulationControl:



Public Member Functions

- `SimulationControl (std::string type, std::string name, GetterMember getterMember, SetterMember setterMember)`
- `virtual ~SimulationControl ()=default`
- `std::string show ()`
- `void setValue (double value)`

Additional Inherited Members

8.120.1 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

Definition at line 24 of file [SimulationControl.h](#).

8.120.2 Constructor & Destructor Documentation

8.120.2.1 SimulationControl() `SimulationControl::SimulationControl (std::string type, std::string name, GetterMember getterMember, SetterMember setterMember)`

Definition at line 18 of file [SimulationControl.cpp](#).

```
00018     : SimulationResponse(type, name, getterMember) {  
00019     this->_type = type;  
00020     this->_setMemberFunction = setterMember;  
00021 }
```

References [SimulationResponse::_type](#).

8.120.2.2 ~SimulationControl() `virtual SimulationControl::~SimulationControl () [virtual], [default]`

8.120.3 Member Function Documentation

8.120.3.1 setValue() `void SimulationControl::setValue (double value)`

Definition at line 27 of file [SimulationControl.cpp](#).

```
00027 {  
00028     this->_setMemberFunction(value);  
00029 }
```

8.120.3.2 show() `std::string SimulationControl::show ()`

Definition at line 23 of file [SimulationControl.cpp](#).

```
00023 {  
00024     return "name=" + this->_name + ", type=" + this->_type;  
00025 }
```

References [SimulationResponse::_name](#), and [SimulationResponse::_type](#).

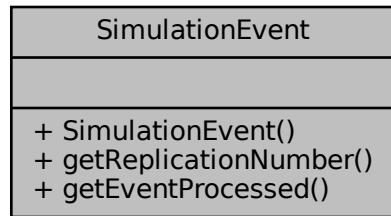
The documentation for this class was generated from the following files:

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

8.121 SimulationEvent Class Reference

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

Collaboration diagram for SimulationEvent:



Public Member Functions

- [SimulationEvent](#) (unsigned int replicationNumber, [Event](#) *event)
- unsigned int [getReplicationNumber](#) () const
- [Event](#) * [getEventProcessed](#) () const

8.121.1 Detailed Description

Definition at line 28 of file [OnEventManager.h](#).

8.121.2 Constructor & Destructor Documentation

```
8.121.2.1 SimulationEvent() SimulationEvent::SimulationEvent (
    unsigned int replicationNumber,
    Event * event ) [inline]
```

Definition at line 31 of file [OnEventManager.h](#).

```
00031
00032     _replicationNumber = replicationNumber;
00033     _event = event;
00034 }
```

8.121.3 Member Function Documentation

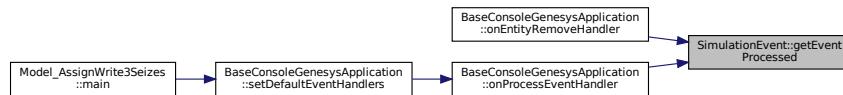
8.121.3.1 `getEventProcessed()` `Event* SimulationEvent::getEventProcessed() const [inline]`

Definition at line 41 of file [OnEventManager.h](#).

```
00041     {
00042         return _event;
00043     }
```

Referenced by [BaseConsoleGenesysApplication::onEntityRemoveHandler\(\)](#), and [BaseConsoleGenesysApplication::onProcessEventHandler\(\)](#).

Here is the caller graph for this function:



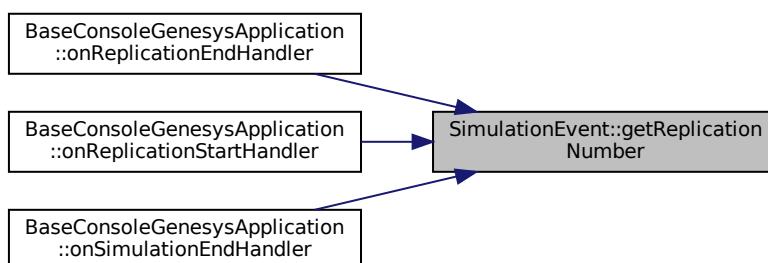
8.121.3.2 `getReplicationNumber()` `unsigned int SimulationEvent::getReplicationNumber() const [inline]`

Definition at line 37 of file [OnEventManager.h](#).

```
00037     {
00038         return _replicationNumber;
00039     }
```

Referenced by [BaseConsoleGenesysApplication::onReplicationEndHandler\(\)](#), [BaseConsoleGenesysApplication::onReplicationStartHandler\(\)](#), and [BaseConsoleGenesysApplication::onSimulationEndHandler\(\)](#).

Here is the caller graph for this function:



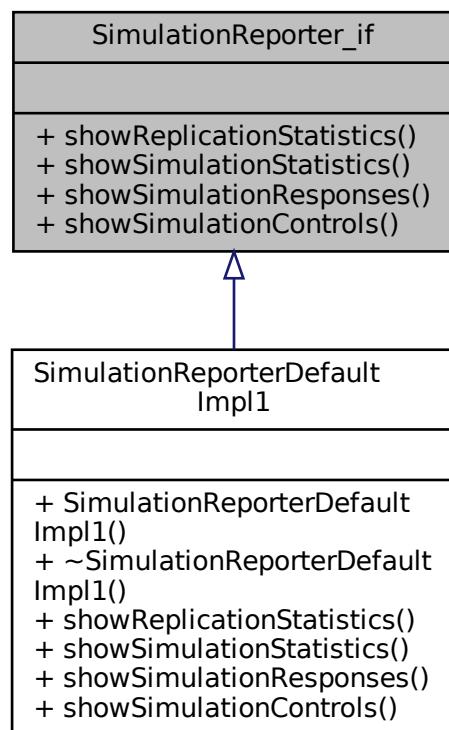
The documentation for this class was generated from the following file:

- [OnEventManager.h](#)

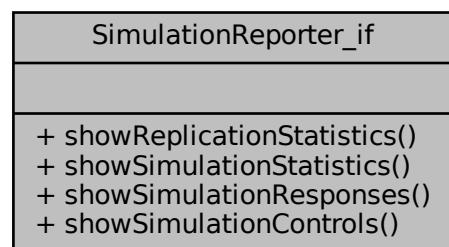
8.122 SimulationReporter_if Class Reference

```
#include <SimulationReporter_if.h>
```

Inheritance diagram for SimulationReporter_if:



Collaboration diagram for SimulationReporter_if:



Public Member Functions

- virtual void `showReplicationStatistics ()=0`
- virtual void `showSimulationStatistics ()=0`
- virtual void `showSimulationResponses ()=0`
- virtual void `showSimulationControls ()=0`

8.122.1 Detailed Description

Definition at line 20 of file [SimulationReporter_if.h](#).

8.122.2 Member Function Documentation

8.122.2.1 showReplicationStatistics() virtual void SimulationReporter_if::showReplicationStatistics()
() [pure virtual]

Implemented in [SimulationReporterDefaultImpl1](#).

8.122.2.2 showSimulationControls() virtual void SimulationReporter_if::showSimulationControls()
() [pure virtual]

Implemented in [SimulationReporterDefaultImpl1](#).

8.122.2.3 showSimulationResponses() virtual void SimulationReporter_if::showSimulationResponses()
() [pure virtual]

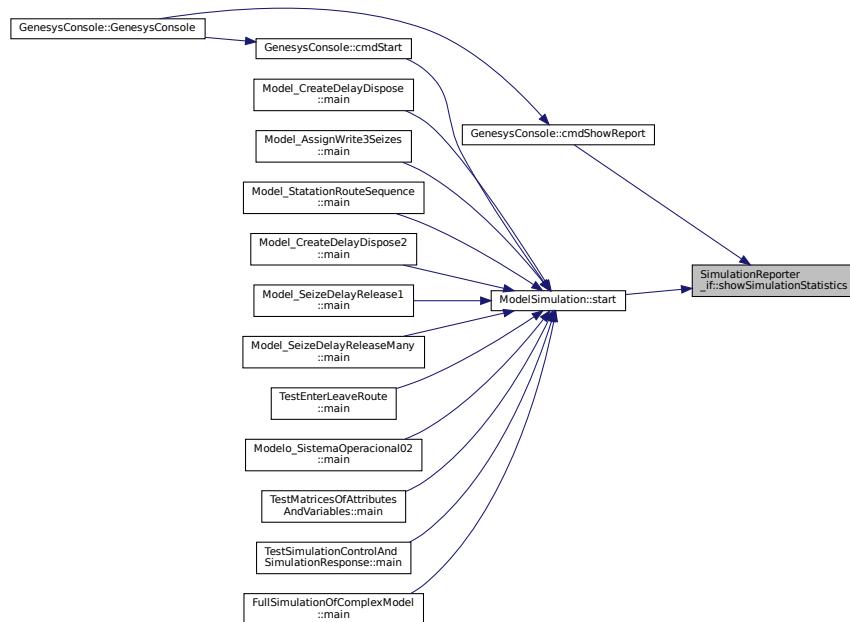
Implemented in [SimulationReporterDefaultImpl1](#).

8.122.2.4 showSimulationStatistics() virtual void SimulationReporter_if::showSimulationStatistics()
 () [pure virtual]

Implemented in [SimulationReporterDefaultImpl1](#).

Referenced by [GenesysConsole::cmdShowReport\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



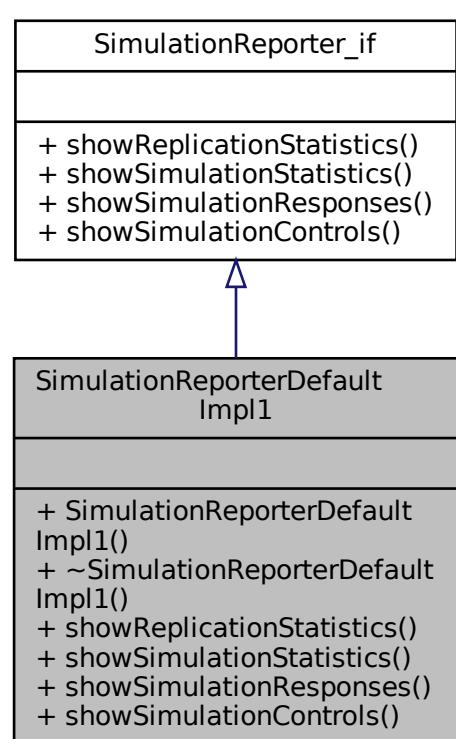
The documentation for this class was generated from the following file:

- [SimulationReporter_if.h](#)

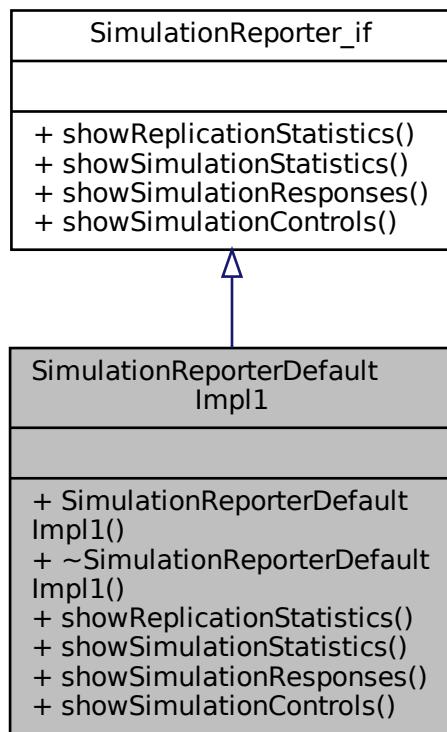
8.123 SimulationReporterDefaultImpl1 Class Reference

```
#include <SimulationReporterDefaultImpl1.h>
```

Inheritance diagram for SimulationReporterDefaultImpl1:



Collaboration diagram for SimulationReporterDefaultImpl1:



Public Member Functions

- `SimulationReporterDefaultImpl1 (ModelSimulation *simulation, Model *model, List< ModelElement * > *statsCountersSimulation)`
- virtual `~SimulationReporterDefaultImpl1 ()=default`
- virtual void `showReplicationStatistics ()`
- virtual void `showSimulationStatistics ()`
- virtual void `showSimulationResponses ()`
- virtual void `showSimulationControls ()`

8.123.1 Detailed Description

Class that implements `SimulationReporter_if` interface and is responsible for building and showing replication and simulation reports

Definition at line 26 of file `SimulationReporterDefaultImpl1.h`.

8.123.2 Constructor & Destructor Documentation

8.123.2.1 SimulationReporterDefaultImpl1() `SimulationReporterDefaultImpl1::SimulationReporter<~DefaultImpl1 (`

```
    ModelSimulation * simulation,
    Model * model,
    List< ModelElement * > * statsCountersSimulation )
```

Definition at line 22 of file [SimulationReporterDefaultImpl1.cpp](#).

```
00022
00023     _simulation = simulation;
00024     _model = model;
00025     _statsCountersSimulation = statsCountersSimulation;
00026 }
```

8.123.2.2 ~SimulationReporterDefaultImpl1() `virtual SimulationReporterDefaultImpl1::~Simulation<~ReporterDefaultImpl1 () [virtual], [default]`

8.123.3 Member Function Documentation

8.123.3.1 showReplicationStatistics() `void SimulationReporterDefaultImpl1::showReplication<~Statistics () [virtual]`

Implements [SimulationReporter_if](#).

Definition at line 40 of file [SimulationReporterDefaultImpl1.cpp](#).

```
00040
00041     _model->getTracer()->traceReport("");
00042     _model->getTracer()->traceReport("Begin of Report for replication " +
00043         std::to_string(_simulation->getCurrentReplicationNumber()) + " of " +
00044         std::to_string(_model->getSimulation()->getNumberOfReplications()));
00045     /* \todo: StatisticsCollector and Counter should NOT be special classes. It should iterate classes
00046     looking for classes that can generate reports.
00047     StatisticsCollector and Counter should override an inherited attribute from ModelElement to
00048     specify they generate report information
00049     look for _generateReportInformation = true; using bool generateReportInformation() const;
00050     */
00051     const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00052     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00053     // runs over all elements and list the statistics for each one, and then the statistics with no
00054     parent
00055     Util::IncIndent();
00056     //this->showSimulationControls();
00057     // copy the list of statistics and counters into a single new list
00058     std::list<ModelElement*>* statisticsAndCounters = new
00059     std::list<ModelElement*>(*(_model->getElements()->getElementList(UtilTypeOfStatisticsCollector)->list()));
00060     std::list<ModelElement*>* counters = new
00061     std::list<ModelElement*>(*(_model->getElements()->getElementList(UtilTypeOfCounter)->list()));
00062     statisticsAndCounters->merge(*counters);
00063     // organizes statistics into a map of maps
00064     std::map< std::string, std::map<std::string, std::list<ModelElement*>>>* mapMapTypeStat = new
00065     std::map<std::string, std::map<std::string, std::list<ModelElement*>>>();
00066
00067     for (std::list<ModelElement*>::iterator it = statisticsAndCounters->begin(); it != statisticsAndCounters->end(); it++) {
00068         std::string parentName, parentTypename;
00069         ModelElement* statOrCnt = (*it);
00070         //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00071         if ((*it)->getClassName() == UtilTypeOfStatisticsCollector) {
00072             StatisticsCollector* stat = dynamic_cast<StatisticsCollector*> (statOrCnt);
00073             parentName = stat->getParent()->getName();
00074             parentTypename = stat->getParent()->getClassName();
00075         } else {
00076             if ((*it)->getClassName() == UtilTypeOfCounter) {
00077                 Counter* cnt = dynamic_cast<Counter*> (statOrCnt);
00078                 parentName = cnt->getParent()->getName();
00079                 parentTypename = cnt->getParent()->getClassName();
00080             }
00081         }
00082     }
00083 }
```

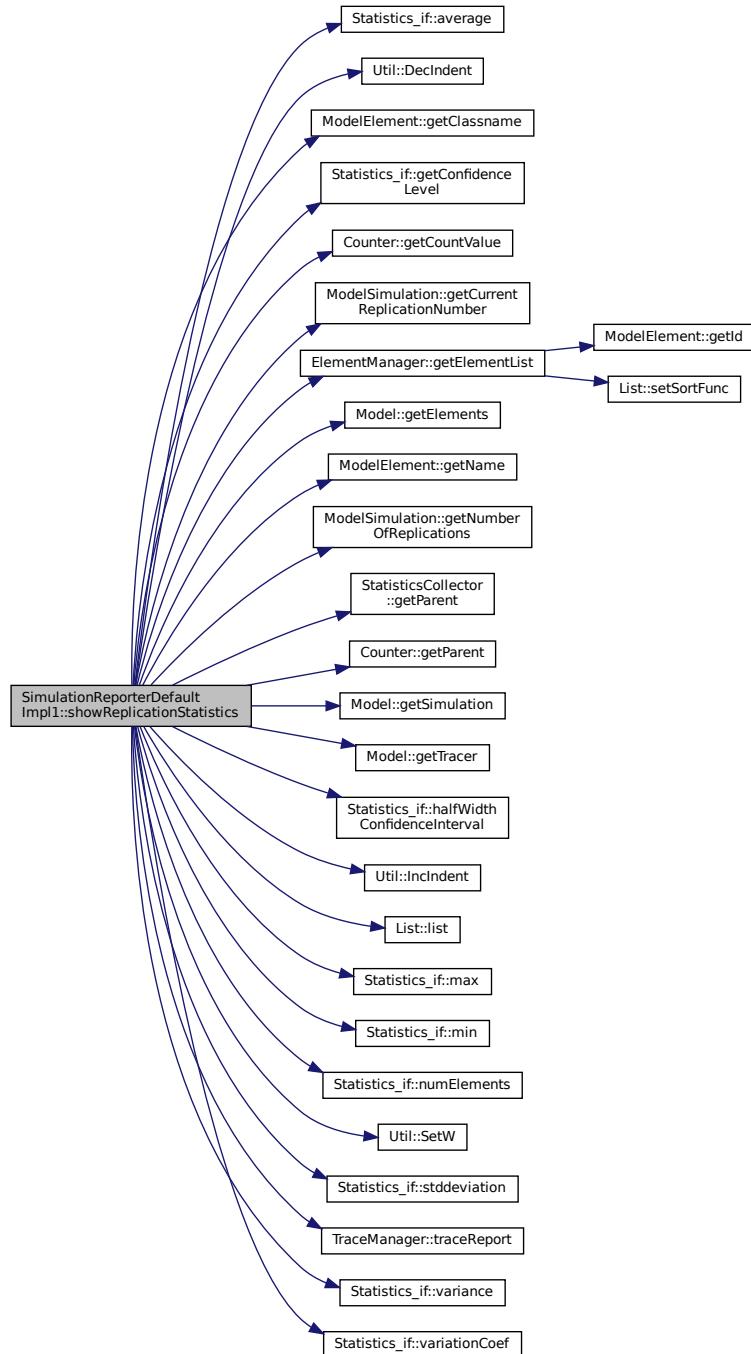
```

00072         }
00073     }
00074     // look for key=parentTypename
00075     std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt =
00076     mapMapTypeStat->find(parentTypename);
00077     if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00078         std::pair< std::string, std::map<std::string, std::list<ModelElement*>>>* newPair = new
00079         std::pair<std::string, std::map<std::string, std::list<ModelElement*>>>(parentTypename, new
00080         std::map<std::string, std::list<ModelElement*>>());
00081         mapMapTypeStat->insert(*newPair);
00082         mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00083     }
00084     //assert(mapMapIt != mapMapTypeStat->end());
00085     std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00086     assert(mapTypeStat != nullptr);
00087     // look for key=parentName
00088     std::map<std::string, std::list<ModelElement*>>>::iterator mapIt =
00089     mapTypeStat->find(parentName);
00090     if (mapIt == mapTypeStat->end()) { // parentName does not exists in map. Include it.
00091         std::pair< std::string, std::list<ModelElement*>>>* newPair = new std::pair<std::string,
00092         std::list<ModelElement*>>();
00093         mapTypeStat->insert(*newPair);
00094         mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00095     }
00096     // get the list and insert the stat in that list
00097     std::list<ModelElement*>>* listStatAndCount = (*mapIt).second;
00098     assert(listStatAndCount != nullptr);
00099     listStatAndCount->insert(listStatAndCount->end(), statOrCnt);
00100     //_model->getTraceManager()->traceReport(parentTypename + " -> " + parentName + " -> " +
00101     stat->show());
00102     }
00103     // now runs over that map of maps showing the statistics
00104     Util::IncIndent();
00105     Util::IncIndent();
00106     _model->getTracer()->traceReport(Util::SetW("name", _nameW) + Util::SetW("elems", _w) +
00107     Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) + Util::SetW("variance",
00108     _w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
00109     Util::SetW("confLevel", _w));
00110     Util::DecIndent();
00111     Util::DecIndent();
00112     for (auto const mapmapItem : *mapMapTypeStat) {
00113         _model->getTracer()->traceReport("Statistis for " + mapmapItem.first + ":" );
00114         Util::IncIndent();
00115         {
00116             for (auto const mapItem : *(mapmapItem.second)) {
00117                 _model->getTracer()->traceReport(mapItem.first + ":" );
00118                 Util::IncIndent();
00119                 {
00120                     // _model->getTracer()->traceReport(Util::SetW("name", _nameW) +
00121                     Util::SetW("elems", _w) + Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) +
00122                     Util::SetW("variance", _w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) +
00123                     Util::SetW("confInterv", _w) + Util::SetW("confLevel", _w));
00124                     for (ModelElement * const item : *(mapItem.second)) {
00125                         if (item->getclassname() == UtilTypeOfStatisticsCollector) {
00126                             Statistics_if* stat = dynamic_cast<StatisticsCollector*>
00127                             (item)->getStatistics();
00128                             _model->getTracer()->traceReport(Util::TraceLevel::report,
00129                             Util::SetW(item->getName() + std::string(_nameW, '.'), _nameW - 1)
00130                             + " " +
00131                             Util::SetW(std::to_string(stat->numElements()), _w) +
00132                             Util::SetW(std::to_string(stat->min()), _w) +
00133                             Util::SetW(std::to_string(stat->max()), _w) +
00134                             Util::SetW(std::to_string(stat->average()), _w) +
00135                             Util::SetW(std::to_string(stat->variance()), _w) +
00136                             Util::SetW(std::to_string(stat->stdDeviation()), _w) +
00137                             Util::SetW(std::to_string(stat->variationCoef()), _w) +
00138                             Util::SetW(std::to_string(stat->halfWidthConfidenceInterval()), _w) +
00139                             Util::SetW(std::to_string(stat->getConfidenceLevel()), _w)
00140                         );
00141                     } else {
00142                         if (item->getclassname() == UtilTypeOfCounter) {
00143                             Counter* count = dynamic_cast<Counter*> (item);
00144                             _model->getTracer()->traceReport(Util::TraceLevel::report,
00145                             Util::SetW(count->getName() + std::string(_nameW, '.'), _nameW
00146                             - 1) + " " +
00147                             Util::SetW(std::to_string(count->getCountValue()), _w)
00148                         );
00149                     }
00150                 }
00151             }
00152         }
00153     }
00154     Util::DecIndent();
00155 }
```

```
00143      }
00144      //this->showSimulationResponses();
00145      Util::DecIndent();
00146      _model->getTracer()->traceReport("End of Report for replication " +
00147          std::to_string(_simulation->getCurrentReplicationNumber()) + " of " +
00148          std::to_string(_model->getSimulation()->getNumberOfReplications()));
00149      _model->getTracer()->traceReport("-----");
00148 }
```

References [Statistics_if::average\(\)](#), [Util::DeclIndent\(\)](#), [ModelElement::getClassname\(\)](#), [Statistics_if::getConfidenceLevel\(\)](#), [Counter::getCountValue\(\)](#), [ModelSimulation::getCurrentReplicationNumber\(\)](#), [ElementManager::getElementList\(\)](#), [Model::getElements\(\)](#), [ModelElement::getName\(\)](#), [ModelSimulation::getNumberOfReplications\(\)](#), [StatisticsCollector::getParent\(\)](#), [Counter::getParent\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [Statistics_if::halfWidthConfidenceInterval\(\)](#), [Util::InclIndent\(\)](#), [List< T >::list\(\)](#), [Statistics_if::max\(\)](#), [Statistics_if::min\(\)](#), [Statistics_if::numElements\(\)](#), [Util::report](#), [Util::SetW\(\)](#), [Statistics_if::stddeviation\(\)](#), [TraceManager::traceReport\(\)](#), [Statistics_if::variance\(\)](#), and [Statistics_if::variationCoef\(\)](#).

Here is the call graph for this function:



8.123.3.2 showSimulationControls() void SimulationReporterDefaultImpl1::showSimulationControls()
() [virtual]

Implements [SimulationReporter_if](#).

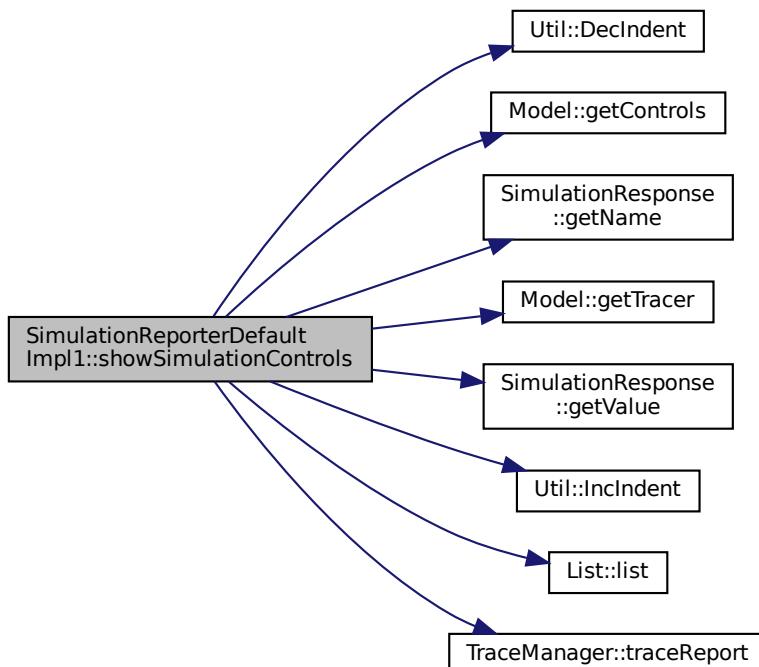
Definition at line 28 of file [SimulationReporterDefaultImpl1.cpp](#).

```

00028     _model->getTracer()->traceReport("Simulation Controls:");
00029     Util::IncIndent();
00030     {
00031         for (std::list<SimulationControl*>::iterator it = _model->getControls()->list()->begin(); it
00032             != _model->getControls()->list()->end(); it++) {
00033             SimulationControl* control = (*it);
00034             _model->getTracer()->traceReport(control->getName() + ":" + 
00035             std::to_string(control->getValue()));
00036         }
00037     Util::DecIndent();
00038 }
```

References [Util::DeclIndent\(\)](#), [Model::getControls\(\)](#), [SimulationResponse::getName\(\)](#), [Model::getTracer\(\)](#), [SimulationResponse::getValue\(\)](#), [Util::InlIndent\(\)](#), [List< T >::list\(\)](#), and [TraceManager::traceReport\(\)](#).

Here is the call graph for this function:



8.123.3.3 showSimulationResponses() void SimulationReporterDefaultImpl1::showSimulationResponses () [virtual]

Implements [SimulationReporter_if](#).

Definition at line 150 of file [SimulationReporterDefaultImpl1.cpp](#).

```

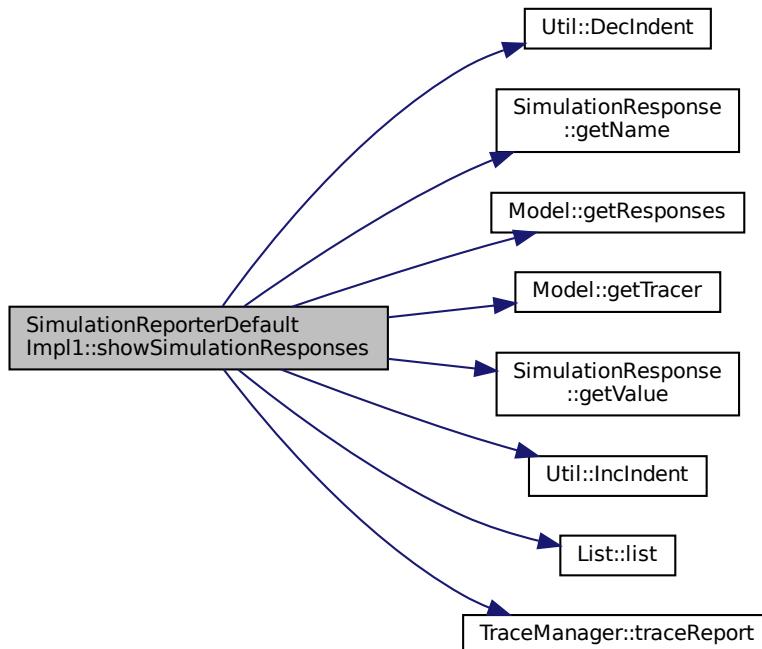
00150
00151     _model->getTracer()->traceReport("Simulation Responses:");
00152     Util::IncIndent();
00153     {
00154         for (std::list<SimulationResponse*>::iterator it = _model->getResponses()->list()->begin(); it
00155             != _model->getResponses()->list()->end(); it++) {
00156             SimulationResponse* response = (*it);
```

```

00156     _model->getTracer()->traceReport(response->getName() + ":" +
00157         std::to_string(response->getValue()));
00158     }
00159     Util::DecIndent();
00160 }
```

References [Util::DeclIndent\(\)](#), [SimulationResponse::getName\(\)](#), [Model::getResponses\(\)](#), [Model::getTracer\(\)](#), [SimulationResponse::getValue\(\)](#), [Util::IncIndent\(\)](#), [List< T >::list\(\)](#), and [TraceManager::traceReport\(\)](#).

Here is the call graph for this function:



8.123.3.4 showSimulationStatistics()

```
void SimulationReporterDefaultImpl1::showSimulationStatistics()
() [virtual]
```

Implements [SimulationReporter_if](#).

Definition at line 162 of file [SimulationReporterDefaultImpl1.cpp](#).

```

00162
00163     cstatsSimulation) {
00164         _model->getTracer()->traceReport("");
00165         _model->getTracer()->traceReport("Begin of Report for Simulation (based on " +
00166             std::to_string(_model->getSimulation()->getNumberOfReplications()) + " replications)");
00167         const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00168         const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00169         // runs over all elements and list the statistics for each one, and then the statistics with no
00170         parent
00171         Util::IncIndent();
00172         // COPY the list of statistics and counters into a single new list
00173         //std::list<ModelElement*>> statisticsAndCounters = //new
00174         std::list<ModelElement*>(*(this->_statsCountersSimulation->list()));
00175         // organizes statistics into a map of maps
00176         std::map< std::string, std::map<std::string, std::list<ModelElement*>>> * mapMapTypeStat = new
00177         std::map<std::string, std::map<std::string, std::list<ModelElement*>>>();
```

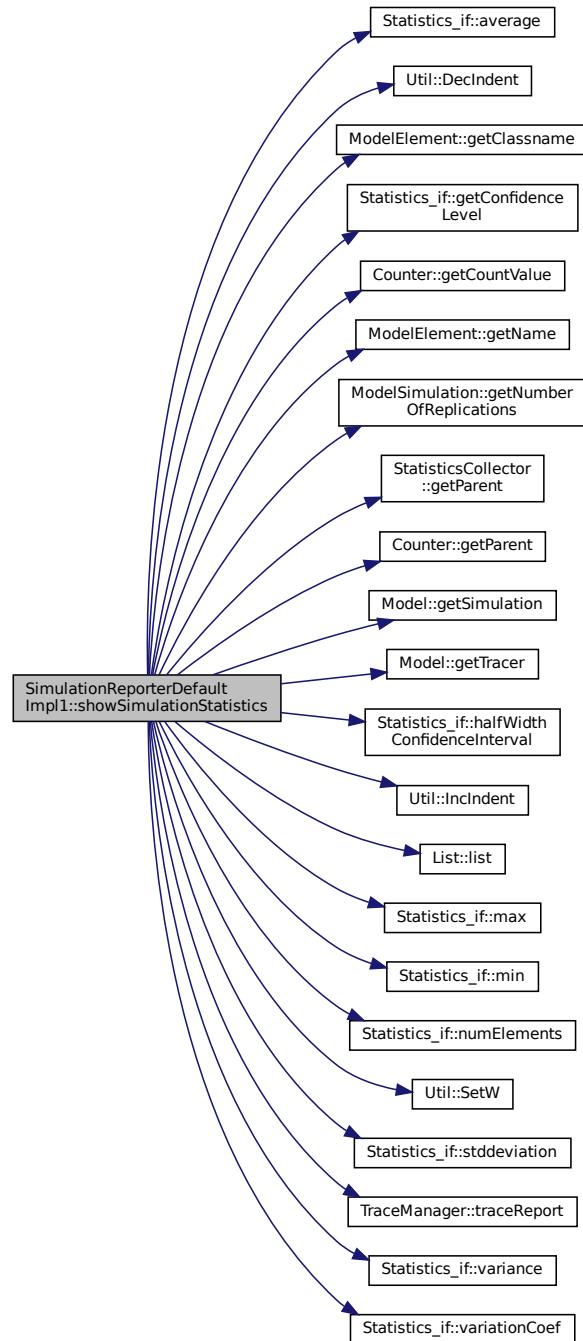
```

00173
00174     for (std::list<ModelElement*>::iterator it = _statsCountersSimulation->list()->begin(); it != _statsCountersSimulation->list()->end(); it++) {
00175         std::string parentName, parentTypename;
00176         ModelElement* statOrCnt = (*it);
00177         //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00178         if ((*it)->getclassname() == UtilTypeOfStatisticsCollector) {
00179             StatisticsCollector* stat = dynamic_cast<StatisticsCollector*> (statOrCnt);
00180             parentName = stat->getParent()->getName();
00181             parentTypename = stat->getParent()->getclassname();
00182         } else {
00183             if ((*it)->getclassname() == UtilTypeOfCounter) {
00184                 Counter* cnt = dynamic_cast<Counter*> (statOrCnt);
00185                 parentName = cnt->getParent()->getName();
00186                 parentTypename = cnt->getParent()->getclassname();
00187             }
00188         }
00189         // look for key=parentTypename
00190         std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt = mapMapTypeStat->find(parentTypename);
00191         if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00192             std::pair< std::string, std::map<std::string, std::list<ModelElement*>>> newPair = new std::pair< std::string, std::list<ModelElement*>> (parentTypename, new std::map<std::string, std::list<ModelElement*>>());
00193             mapMapTypeStat->insert(*newPair);
00194             mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00195         }
00196         assert(mapMapIt != mapMapTypeStat->end());
00197         std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00198         assert(mapTypeStat != nullptr);
00199         // look for key=parentName
00200         std::map<std::string, std::list<ModelElement*>>>::iterator mapIt = mapTypeStat->find(parentName);
00201         if (mapIt == mapTypeStat->end()) { // parentName does not exists in map. Include it.
00202             std::pair< std::string, std::list<ModelElement*>>> newPair = new std::pair< std::string, std::list<ModelElement*>> ();
00203             std::list<ModelElement*>>> (parentName, new std::list<ModelElement*> ());
00204             mapTypeStat->insert(*newPair);
00205             mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00206         }
00207         // get the list and insert the stat in that list
00208         std::list<ModelElement*>> listStat = (*mapIt).second;
00209         listStat->insert(listStat->end(), statOrCnt);
00210         //_model->getTraceManager()->traceReport (parentTypename + " -> " + parentName + " -> " +
00211         stat->show());
00212         // now runs over that map of maps showing the statistics
00213         //int w = 12;
00214         Util::IncIndent();
00215         Util::IncIndent();
00216         _model->getTracer()->traceReport (Util::SetW("name", _nameW) + Util::SetW("elems", _w) +
00217         Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) + Util::SetW("variance",
00218         _w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
00219         Util::SetW("confLevel", _w));
00220         Util::DecIndent();
00221         Util::DecIndent();
00222         for (auto const mapmapItem : *mapMapTypeStat) {
00223             _model->getTracer()->traceReport ("Statistis for " + mapmapItem.first + ":");
00224             Util::IncIndent();
00225         {
00226             for (auto const mapItem : *(mapmapItem.second)) {
00227                 _model->getTracer()->traceReport (mapItem.first + ":");
00228                 Util::IncIndent();
00229             {
00230                 for (ModelElement * const item : *(mapItem.second)) {
00231                     if (item->getclassname() == UtilTypeOfStatisticsCollector) {
00232                         Statistics_if* stat = dynamic_cast<StatisticsCollector*>
00233                         (item)->getStatistics();
00234                         _model->getTracer()->traceReport (Util::TraceLevel::report,
00235                         Util::SetW(item->getName() + std::string(_nameW, '.'), _nameW - 1)
00236                         + " " +
00237                         Util::SetW(std::to_string(stat->numElements()), _w) +
00238                         Util::SetW(std::to_string(stat->min()), _w) +
00239                         Util::SetW(std::to_string(stat->max()), _w) +
00240                         Util::SetW(std::to_string(stat->average()), _w) +
00241                         Util::SetW(std::to_string(stat->variance()), _w) +
00242                         Util::SetW(std::to_string(stat->stddeviation()), _w) +
00243                         Util::SetW(std::to_string(stat->variationCoef()), _w) +
00244                         Util::SetW(std::to_string(stat->halfWidthConfidenceInterval()), _w) +
00245                         Util::SetW(std::to_string(stat->getConfidenceLevel()), _w));
00246                     } else {
00247                         if (item->getclassname() == UtilTypeOfCounter) {
00248                             Counter* cnt = dynamic_cast<Counter*> (item);
00249                             _model->getTracer()->traceReport (Util::TraceLevel::report,
00250                             Util::SetW(cnt->getName() + std::string(_nameW, '.'), _nameW -
00251                             1));
00252                         }
00253                     }
00254                 }
00255             }
00256         }
00257     }
00258 }
```

```
1) + " " +
00247                               Util::SetW(std::to_string(cnt->getCountValue()), _w)
00248                               );
00249
00250                         }
00251                     }
00252                 }
00253             }
00254         Util::DecIndent();
00255     }
00256 }
00257 Util::DecIndent();
00258 }
00259 Util::DecIndent();
00260 _model->getTracer()->traceReport("End of Report for Simulation");
00261
00262 }
```

References [Statistics_if::average\(\)](#), [Util::DecIndent\(\)](#), [ModelElement::getClassname\(\)](#), [Statistics_if::getConfidenceLevel\(\)](#), [Counter::getCountValue\(\)](#), [ModelElement::getName\(\)](#), [ModelSimulation::getNumberOfReplications\(\)](#), [StatisticsCollector::getParent\(\)](#), [Counter::getParent\(\)](#), [Model::getSimulation\(\)](#), [Model::getTracer\(\)](#), [Statistics_if::halfWidthConfidenceInterval\(\)](#), [Util::InlIndent\(\)](#), [List< T >::list\(\)](#), [Statistics_if::max\(\)](#), [Statistics_if::min\(\)](#), [Statistics_if::numElements\(\)](#), [Util::report](#), [Util::SetW\(\)](#), [Statistics_if::stddeviation\(\)](#), [TraceManager::traceReport\(\)](#), [Statistics_if::variance\(\)](#), and [Statistics_if::variationCoef\(\)](#).

Here is the call graph for this function:



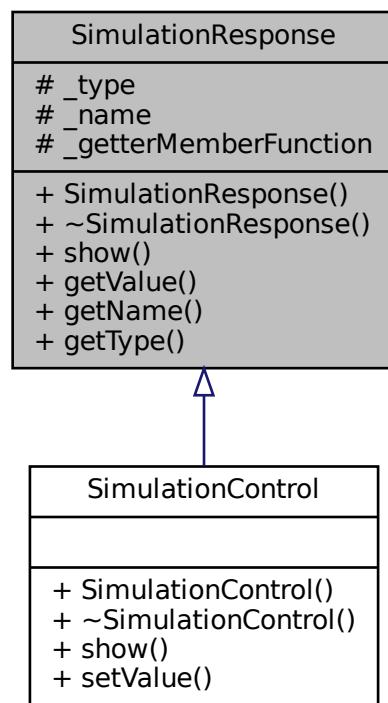
The documentation for this class was generated from the following files:

- [SimulationReporterDefaultImpl1.h](#)
- [SimulationReporterDefaultImpl1.cpp](#)

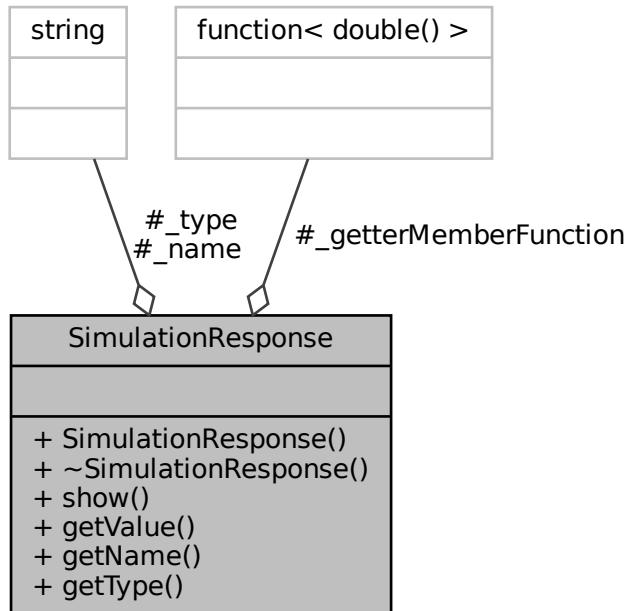
8.124 SimulationResponse Class Reference

```
#include <SimulationResponse.h>
```

Inheritance diagram for SimulationResponse:



Collaboration diagram for SimulationResponse:



Public Member Functions

- `SimulationResponse` (`std::string type, std::string name, GetterMember getterMember)`
- virtual `~SimulationResponse ()`=default
- `std::string show ()`
- `double getValue ()`
- `std::string getName () const`
- `std::string getType () const`

Protected Attributes

- `std::string _type`
- `std::string _name`
- `GetterMember _getterMemberFunction`

8.124.1 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 has 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

Definition at line 25 of file `SimulationResponse.h`.

8.124.2 Constructor & Destructor Documentation

8.124.2.1 **SimulationResponse()**

```
SimulationResponse::SimulationResponse (
    std::string type,
    std::string name,
    GetterMember getterMember )
```

Definition at line 19 of file [SimulationResponse.cpp](#).

```
00019
00020     _type = type;
00021     _name = name;
00022     _getterMemberFunction = getterMember;
00023 }
```

References [_getterMemberFunction](#), [_name](#), and [_type](#).

8.124.2.2 **~SimulationResponse()**

```
virtual SimulationResponse::~SimulationResponse () [virtual],  
[default]
```

8.124.3 Member Function Documentation

8.124.3.1 **getName()**

```
std::string SimulationResponse::getName () const
```

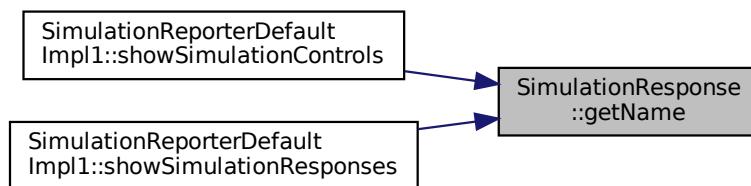
Definition at line 29 of file [SimulationResponse.cpp](#).

```
00029
00030     return _name;
00031 }
```

References [_name](#).

Referenced by [SimulationReporterDefaultImpl1::showSimulationControls\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#).

Here is the caller graph for this function:



8.124.3.2 `getType()` `std::string SimulationResponse::getType () const`

Definition at line 33 of file [SimulationResponse.cpp](#).

```
00033
00034     return _type;
00035 }
```

References [_type](#).

8.124.3.3 `getValue()` `double SimulationResponse::getValue ()`

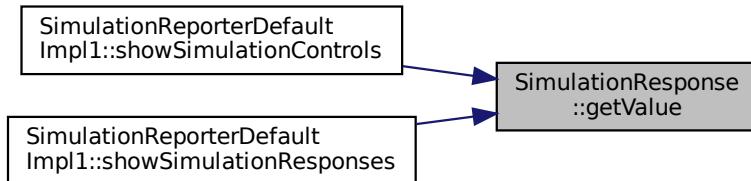
Definition at line 37 of file [SimulationResponse.cpp](#).

```
00037
00038     return this->_getterMemberFunction();
00039 }
```

References [_getterMemberFunction](#).

Referenced by [SimulationReporterDefaultImpl1::showSimulationControls\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#).

Here is the caller graph for this function:

**8.124.3.4 `show()`** `std::string SimulationResponse::show ()`

Definition at line 25 of file [SimulationResponse.cpp](#).

```
00025
00026     return "name=" + this->_name + ", type=" + this->_type;
00027 }
```

References [_name](#), and [_type](#).

8.124.4 Member Data Documentation

8.124.4.1 _getterMemberFunction `GetterMember` `SimulationResponse::_getterMemberFunction` [protected]

Definition at line 38 of file [SimulationResponse.h](#).

Referenced by [getValue\(\)](#), and [SimulationResponse\(\)](#).

8.124.4.2 _name `std::string` `SimulationResponse::_name` [protected]

Definition at line 37 of file [SimulationResponse.h](#).

Referenced by [getName\(\)](#), [SimulationControl::show\(\)](#), [show\(\)](#), and [SimulationResponse\(\)](#).

8.124.4.3 _type `std::string` `SimulationResponse::_type` [protected]

Definition at line 36 of file [SimulationResponse.h](#).

Referenced by [getType\(\)](#), [SimulationControl::show\(\)](#), [show\(\)](#), [SimulationControl::SimulationControl\(\)](#), and [SimulationResponse\(\)](#).

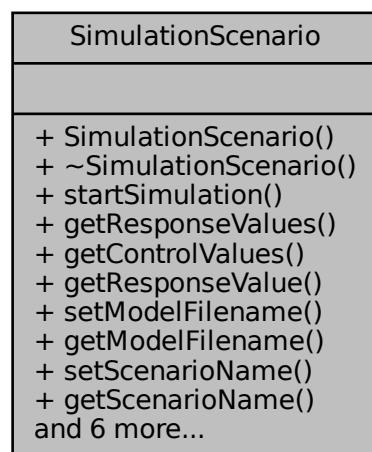
The documentation for this class was generated from the following files:

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

8.125 SimulationScenario Class Reference

```
#include <SimulationScenario.h>
```

Collaboration diagram for SimulationScenario:



Public Member Functions

- `SimulationScenario ()`
- `virtual ~SimulationScenario ()=default`
- `bool startSimulation (std::string *errorMessage)`
- `std::list< std::pair< std::string, double > * > * getResponseValues () const`
- `std::list< std::pair< std::string, double > * > * getControlValues () const`
- `double getResponseValue (std::string responseName)`
- `void setModelFilename (std::string _modelFilename)`
- `std::string getModelFilename () const`
- `void setScenarioName (std::string _name)`
- `std::string getScenarioName () const`
- `void setScenarioDescription (std::string _scenarioDescription)`
- `std::string getScenarioDescription () const`
- `std::list< std::pair< std::string, double > * > * getSelectedControls () const`
- `double getControlValue (std::string controlName)`
- `void setControlValue (std::string controlName, double value)`
- `std::list< std::string > * getSelectedResponses () const`

8.125.1 Detailed Description

Represents a scenario where a specific model (defined by 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.

Definition at line 25 of file `SimulationScenario.h`.

8.125.2 Constructor & Destructor Documentation

8.125.2.1 `SimulationScenario()` `SimulationScenario::SimulationScenario ()`

Definition at line 16 of file `SimulationScenario.cpp`.

```
00016 {  
00017 }
```

8.125.2.2 `~SimulationScenario()` `virtual SimulationScenario::~SimulationScenario () [virtual], [default]`

8.125.3 Member Function Documentation

8.125.3.1 `getControlValue()` `double SimulationScenario::getControlValue (std::string controlName)`

8.125.3.2 getControlValues() std::list<std::pair<std::string, double>*>* SimulationScenario::getControlValues () const

8.125.3.3 getModelFilename() std::string SimulationScenario::getModelFilename () const

Definition at line 41 of file [SimulationScenario.cpp](#).

```
00041
00042     return _modelFilename;
00043 }
```

8.125.3.4 getResponseValue() double SimulationScenario::getResponseValue (std::string responseName)

8.125.3.5 getResponseValues() std::list<std::pair<std::string, double>*>* SimulationScenario::getResponseValues () const

The final result of the simulationScenario

8.125.3.6 getScenarioDescription() std::string SimulationScenario::getScenarioDescription () const

Definition at line 57 of file [SimulationScenario.cpp](#).

```
00057
00058     return _scenarioDescription;
00059 }
```

8.125.3.7 getScenarioName() std::string SimulationScenario::getScenarioName () const

Definition at line 33 of file [SimulationScenario.cpp](#).

```
00033
00034     return _scenarioName;
00035 }
```

8.125.3.8 getSelectedControls() std::list< std::pair< std::string, double > * > * SimulationScenario::getSelectedControls () const

Definition at line 49 of file [SimulationScenario.cpp](#).

```
00049
00050     return _selectedControls;
00051 }
```

8.125.3.9 `getSelectedResponses()` `std::list< std::string > * SimulationScenario::getSelectedResponses () const`

Definition at line 45 of file [SimulationScenario.cpp](#).

```
00045 {  
00046     return _selectedResponses;  
00047 }
```

8.125.3.10 `setControlValue()` `void SimulationScenario::setControlValue (std::string controlName, double value)`

8.125.3.11 `setModelFilename()` `void SimulationScenario::setModelFilename (std::string _modelFilename)`

Definition at line 37 of file [SimulationScenario.cpp](#).

```
00037 {  
00038     this->_modelFilename = _modelFilename;  
00039 }
```

8.125.3.12 `setScenarioDescription()` `void SimulationScenario::setScenarioDescription (std::string _scenarioDescription)`

Definition at line 53 of file [SimulationScenario.cpp](#).

```
00053 {  
00054     this->_scenarioDescription = _scenarioDescription;  
00055 }
```

8.125.3.13 `setScenarioName()` `void SimulationScenario::setScenarioName (std::string _name)`

Definition at line 29 of file [SimulationScenario.cpp](#).

```
00029 {  
00030     this->_scenarioName = _name;  
00031 }
```

8.125.3.14 `startSimulation()` `bool SimulationScenario::startSimulation (std::string * errorMessage)`

Definition at line 19 of file [SimulationScenario.cpp](#).

```
00019 {  
00020     // model->loadmodel _modelFilename  
00021     // set values for the _selectedControls  
00022     // model->startSimulation  
00023     // get the value of the _selectedResponses from the model and store results on _responseValues  
00024     // clear selected controls and responses  
00025     // close the model  
00026     return false;  
00027 }
```

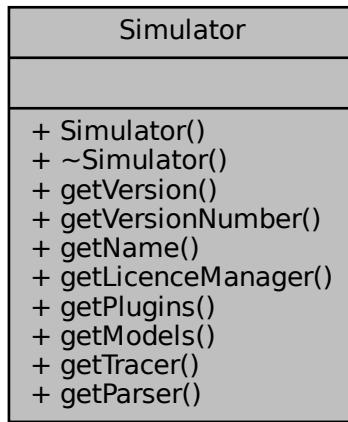
The documentation for this class was generated from the following files:

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

8.126 Simulator Class Reference

```
#include <Simulator.h>
```

Collaboration diagram for Simulator:



Public Member Functions

- [Simulator \(\)](#)
- virtual [~Simulator \(\)=default](#)
- [std::string getVersion \(\) const](#)
- [unsigned int getVersionNumber \(\) const](#)
- [std::string getName \(\) const](#)
- [LicenceManager * getLicenceManager \(\) const](#)
- [PluginManager * getPlugins \(\) const](#)
- [ModelManager * getModels \(\) const](#)
- [TraceManager * getTracer \(\) const](#)
- [ParserManager * getParser \(\) const](#)

8.126.1 Detailed Description

The main class of the ReGenesys KERNEL simulation. It gives access to simulation models and tools. Simulation is the top level class and is supposed to be available to application as a dynamic linked library.

Definition at line [33](#) of file [Simulator.h](#).

8.126.2 Constructor & Destructor Documentation

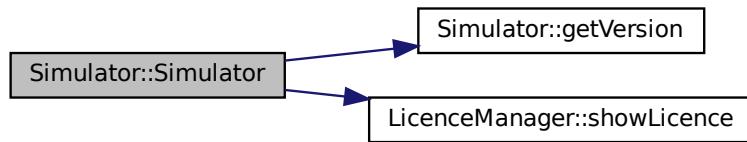
8.126.2.1 Simulator() Simulator::Simulator ()

Definition at line 37 of file [Simulator.cpp](#).

```
00037     {
00038     // This is the ONLY method in the entire software where std::cout is allowed.
00039     std::cout << "STARTING " << _name << ", version " << getVersion() << std::endl;
00040     _licenceManager = new LicenceManager(this);
00041     _pluginManager = new PluginManager(this);
00042     _modelManager = new ModelManager(this);
00043     //_toolManager = new ToolManager(this);
00044     _traceManager = new TraceManager(this);
00045     std::cout << '|' << '\t' << _licenceManager->showLicence() << std::endl;
00046     //std::cout << '|' << '\t' << _licenceManager->showActivationCode() << std::endl;
00047     //std::cout << '|' << '\t' << _licenceManager->showLimits() << std::endl;
00048 }
```

References [getVersion\(\)](#), and [LicenceManager::showLicence\(\)](#).

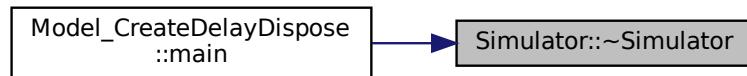
Here is the call graph for this function:



8.126.2.2 ~Simulator() virtual Simulator::~Simulator () [virtual], [default]

Referenced by [Model_CreateDelayDispose::main\(\)](#).

Here is the caller graph for this function:



8.126.3 Member Function Documentation

8.126.3.1 getLicenceManager() `LicenceManager * Simulator::getLicenceManager () const`

Definition at line 78 of file [Simulator.cpp](#).

```
00078
00079     return _licenceManager;
00080 }
```

Referenced by [ModelCheckerDefaultImpl1::checkLimits\(\)](#).

Here is the caller graph for this function:

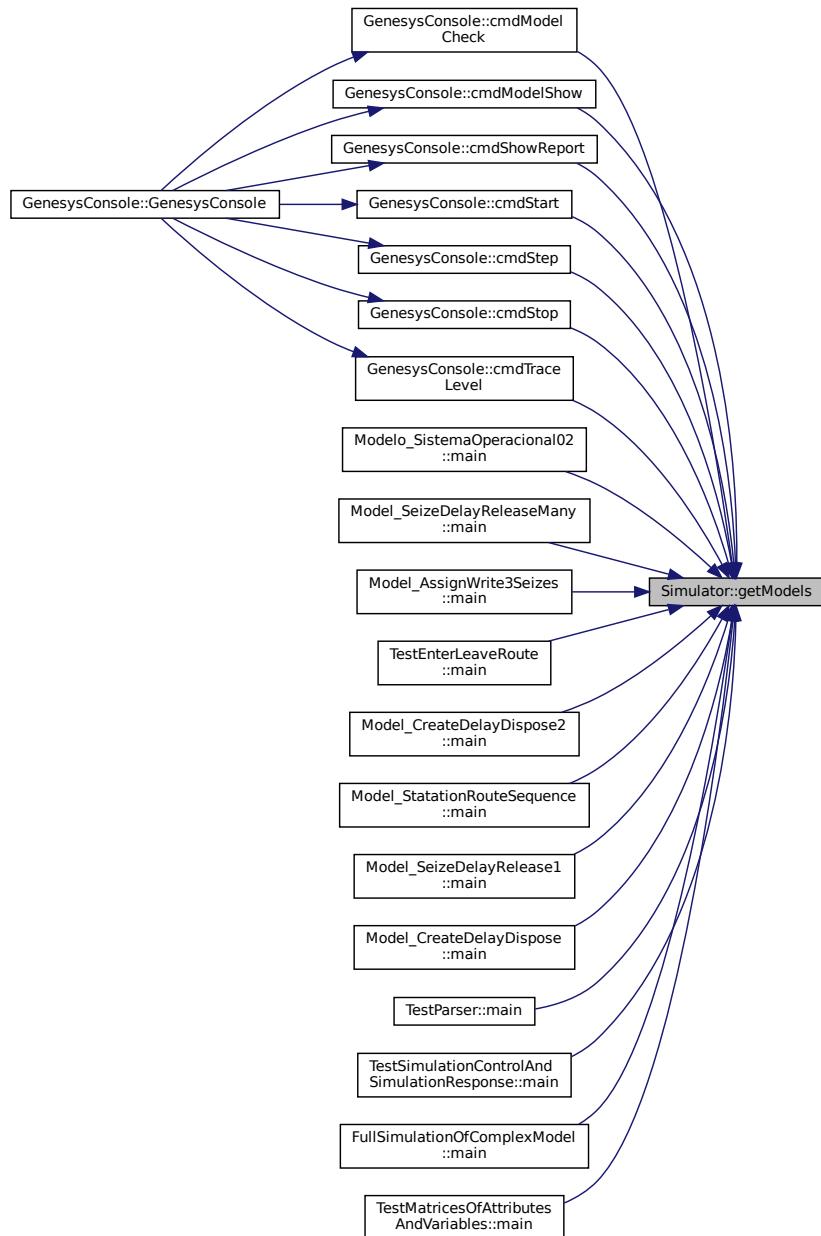
**8.126.3.2 getModel()** `ModelManager * Simulator::getModel () const`

Definition at line 54 of file [Simulator.cpp](#).

```
00054
00055     return _modelManager;
00056 }
```

Referenced by [GenesysConsole::cmdModelCheck\(\)](#), [GenesysConsole::cmdModelShow\(\)](#), [GenesysConsole::cmdShowReport\(\)](#), [GenesysConsole::cmdStart\(\)](#), [GenesysConsole::cmdStep\(\)](#), [GenesysConsole::cmdStop\(\)](#), [GenesysConsole::cmdTraceLevel\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestParser::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.126.3.3 getName() std::string Simulator::getName() const

Definition at line 74 of file [Simulator.cpp](#).

```

00074
00075     return _name;
00076 }
```

8.126.3.4 getParser() `ParserManager * Simulator::getParser () const`

Definition at line 62 of file [Simulator.cpp](#).

```
00062
00063     return _parserManager;
00064 }
```

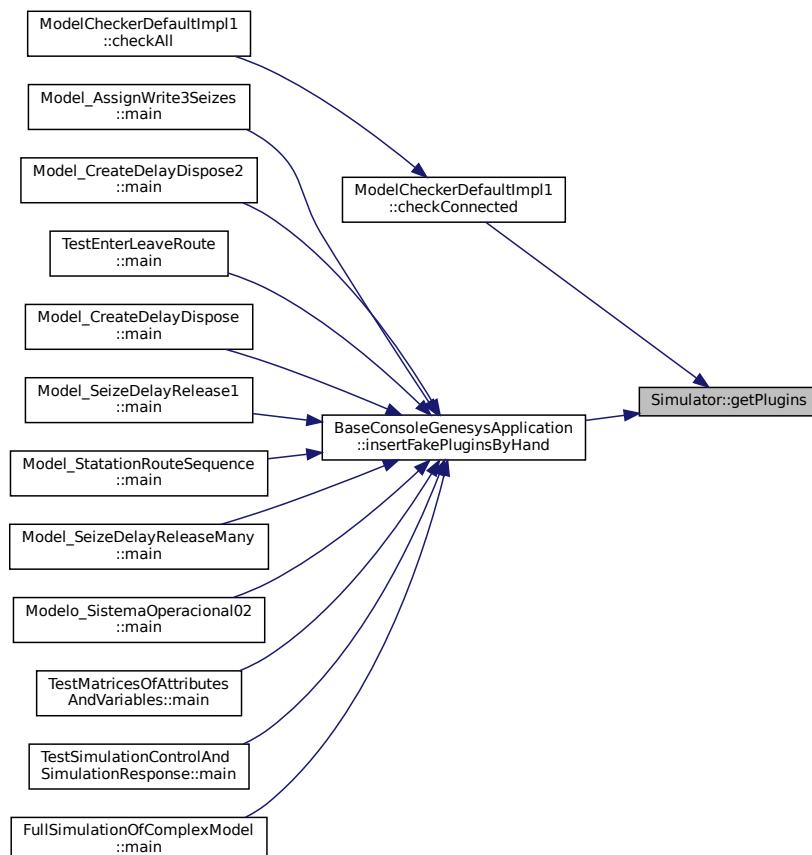
8.126.3.5 getPlugins() `PluginManager * Simulator::getPlugins () const`

Definition at line 50 of file [Simulator.cpp](#).

```
00050
00051     return _pluginManager;
00052 }
```

Referenced by [ModelCheckerDefaultImpl1::checkConnected\(\)](#), and [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#).

Here is the caller graph for this function:



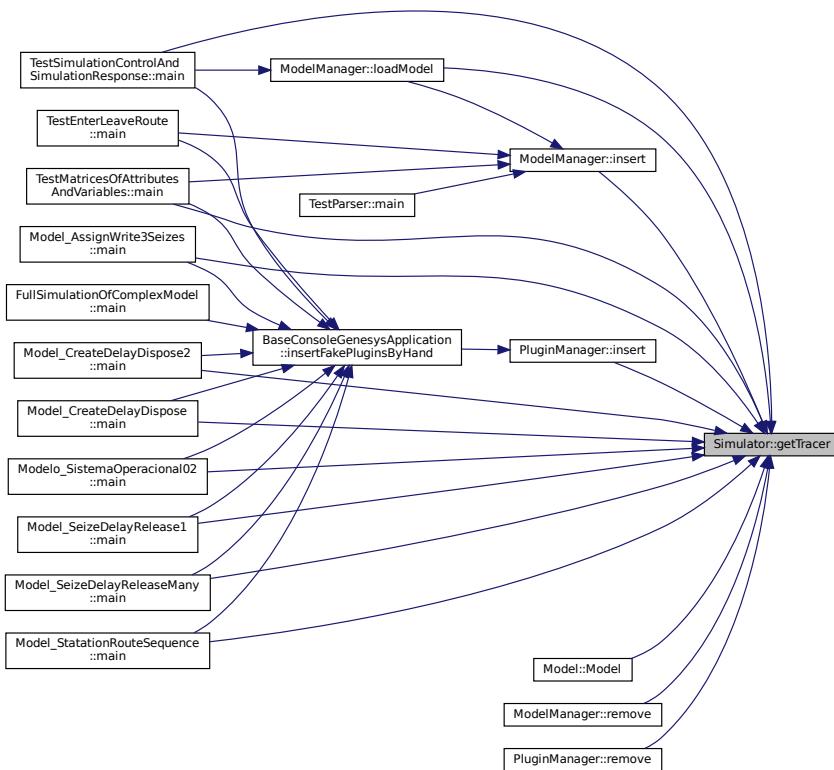
8.126.3.6 `getTracer()` `TraceManager * Simulator::getTracer () const`

Definition at line 58 of file [Simulator.cpp](#).

```
00058     {
00059         return _traceManager;
00060     }
```

Referenced by [ModelManager::insert\(\)](#), [PluginManager::insert\(\)](#), [ModelManager::loadModel\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), [Model::Model\(\)](#), [ModelManager::remove\(\)](#), and [PluginManager::remove\(\)](#).

Here is the caller graph for this function:



8.126.3.7 `getVersion()` `std::string Simulator::getVersion () const`

Definition at line 66 of file [Simulator.cpp](#).

```
00066     {
00067         return std::to_string(_versionNumber) + " (" + _versionName + ")";
00068     }
```

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

Here is the caller graph for this function:



8.126.3.8 getVersionNumber()

```
unsigned int Simulator::getVersionNumber() const
```

Definition at line 70 of file [Simulator.cpp](#).

```
00070
00071     return _versionNumber;
00072 }
```

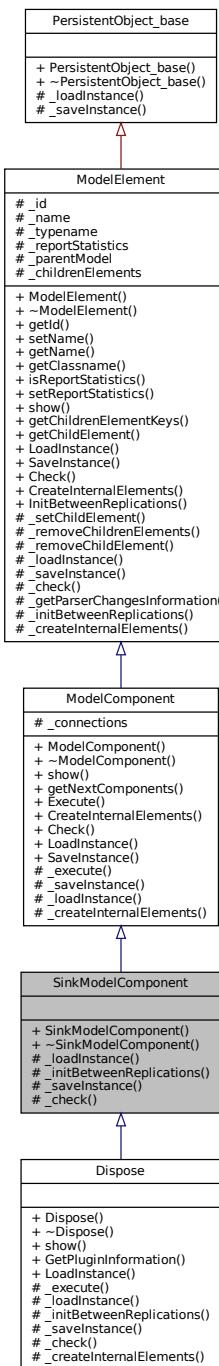
The documentation for this class was generated from the following files:

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

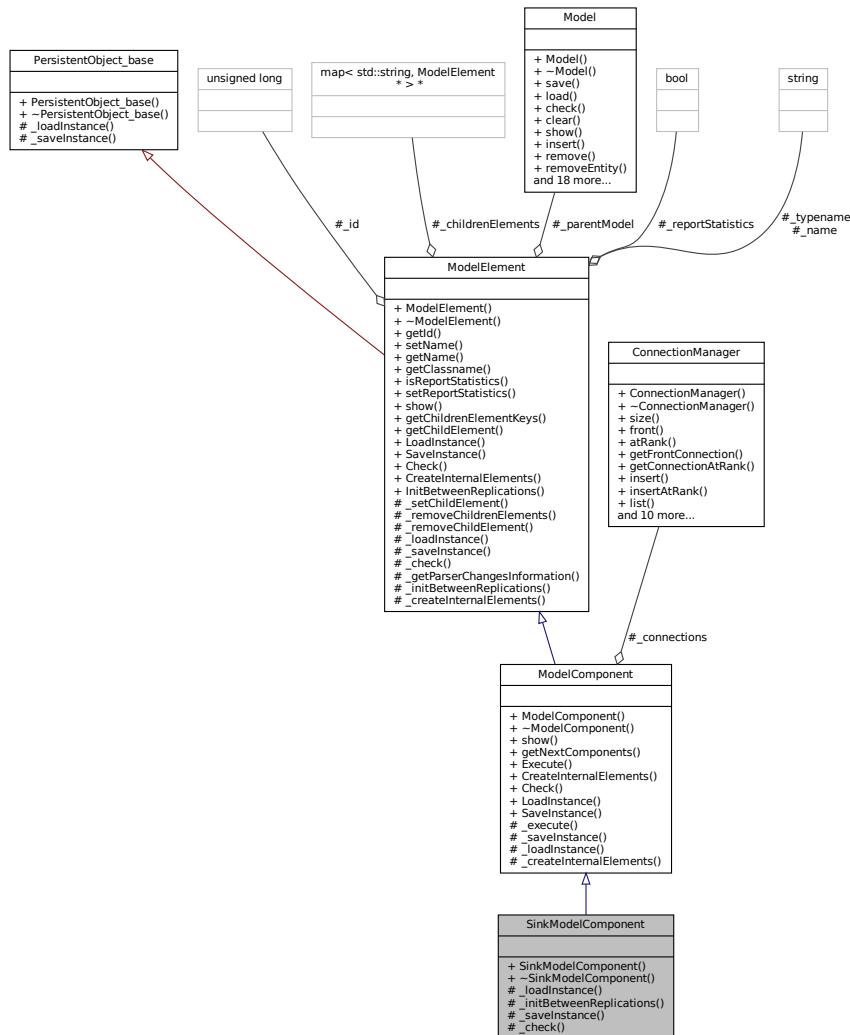
8.127 SinkModelComponent Class Reference

```
#include <SinkModelComponent.h>
```

Inheritance diagram for SinkModelComponent:



Collaboration diagram for SinkModelComponent:



Public Member Functions

- `SinkModelComponent (Model *model, std::string componentTypename, std::string name="")`
- virtual `~SinkModelComponent ()=default`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.127.1 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.

Definition at line 25 of file [SinkModelComponent.h](#).

8.127.2 Constructor & Destructor Documentation

8.127.2.1 `SinkModelComponent()` `SinkModelComponent::SinkModelComponent (`
 `Model * model,`
 `std::string componentTypename,`
 `std::string name = "")`

Definition at line 18 of file [SinkModelComponent.cpp](#).

```
00018     ModelComponent(model, componentTypename, name) {  
00019 }
```

:

8.127.2.2 `~SinkModelComponent()` `virtual SinkModelComponent::~SinkModelComponent () [virtual],`
[default]

8.127.3 Member Function Documentation

8.127.3.1 `_check()` `bool SinkModelComponent::_check (`
 `std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Reimplemented in [Dispose](#).

Definition at line 38 of file [SinkModelComponent.cpp](#).

```
00038  
00039     return true;  
00040 }
```

{

8.127.3.2 `_initBetweenReplications()` `void SinkModelComponent::_initBetweenReplications () [protected],`
[virtual]

Reimplemented from [ModelElement](#).

Reimplemented in [Dispose](#).

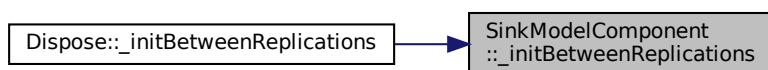
Definition at line 29 of file [SinkModelComponent.cpp](#).

```
00029  
00030 }
```

{

Referenced by [Dispose::_initBetweenReplications\(\)](#).

Here is the caller graph for this function:



```
8.127.3.3 _loadInstance() bool SinkModelComponent::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Reimplemented in [Dispose](#).

Definition at line 21 of file [SinkModelComponent.cpp](#).

```
00021
00022     bool res = ModelComponent::_loadInstance(fields);
00023     if (res) {
00024         //this->_reportStatistics = std::stoi((*fields->find("collectStatistics")).second) != 0;
00025     }
00026     return res;
00027 }
```

References [ModelComponent::_loadInstance\(\)](#).

Here is the call graph for this function:



```
8.127.3.4 _saveInstance() std::map< std::string, std::string > * SinkModelComponent::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

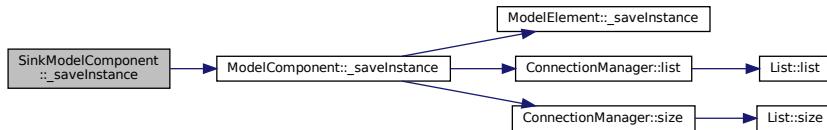
Reimplemented in [Dispose](#).

Definition at line 32 of file [SinkModelComponent.cpp](#).

```
00032
00033     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00034     //fields->emplace("collectStatistics", std::to_string(this->_reportStatistics));
00035     return fields;
00036 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



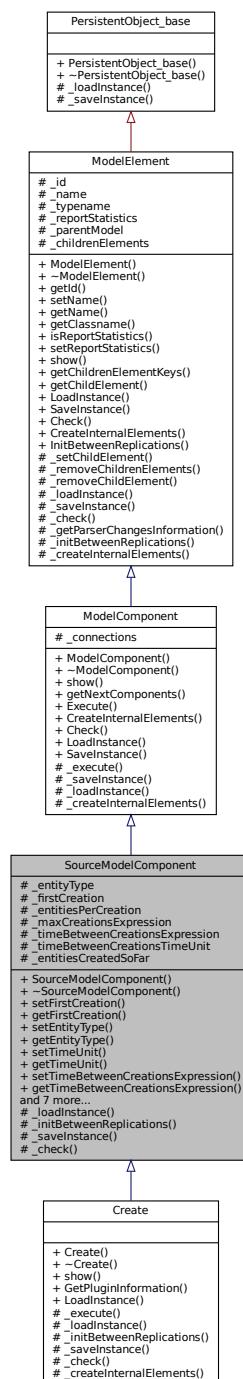
The documentation for this class was generated from the following files:

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

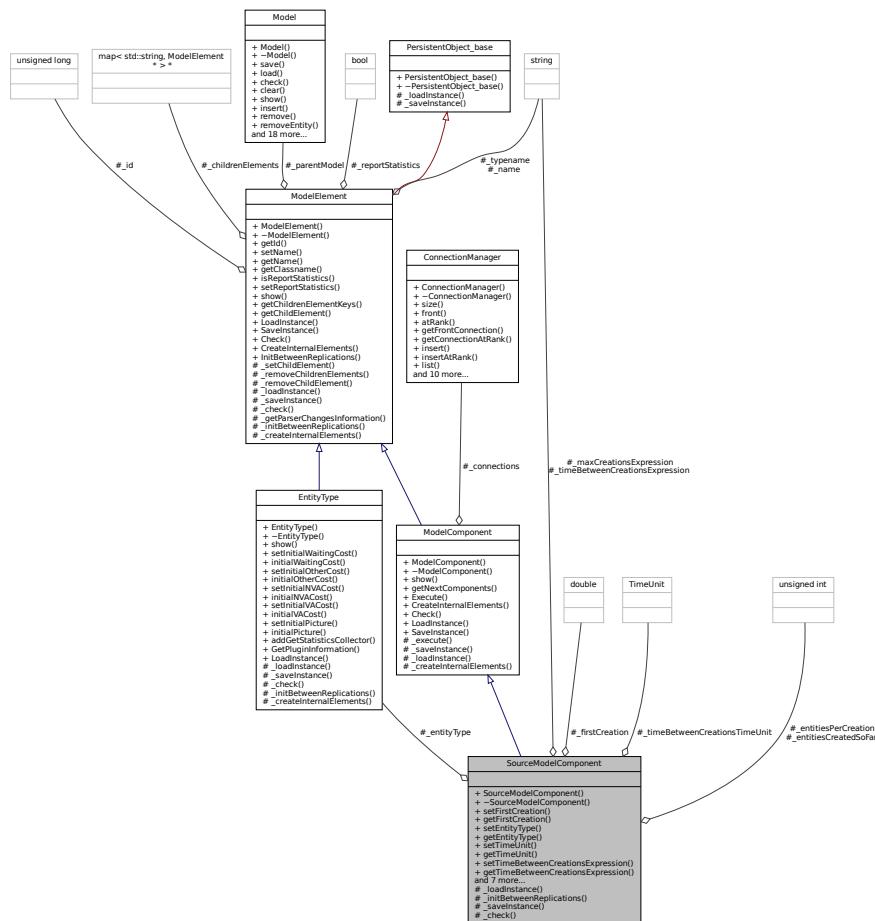
8.128 SourceModelComponent Class Reference

```
#include <SourceModelComponent.h>
```

Inheritance diagram for SourceModelComponent:



Collaboration diagram for SourceModelComponent:



Public Member Functions

- **SourceModelComponent (Model *model, std::string componentTypename, std::string name="")**
- virtual ~SourceModelComponent ()=default
- void **setFirstCreation (double _firstCreation)**
- double **getFirstCreation () 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 (std::string _maxCreationsExpression)**
- std::string **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 Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual void `_initBetweenReplications` ()
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)

Protected Attributes

- `EntityType * _entityType`
- double `_firstCreation` = 0.0
- unsigned int `_entitiesPerCreation` = 1
- std::string `_maxCreationsExpression` = std::to_string(std::numeric_limits<unsigned int>::max())
- std::string `_timeBetweenCreationsExpression` = "EXPO(1)"
- `Util::TimeUnit _timeBetweenCreationsTimeUnit` = `Util::TimeUnit::second`
- unsigned int `_entitiesCreatedSoFar` = 0

Additional Inherited Members

8.128.1 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

Definition at line 28 of file [SourceModelComponent.h](#).

8.128.2 Constructor & Destructor Documentation

8.128.2.1 SourceModelComponent() `SourceModelComponent::SourceModelComponent (`
 `Model * model,`
 `std::string componentTypename,`
 `std::string name = "")`

Definition at line 20 of file [SourceModelComponent.cpp](#).

```
00020     ModelComponent(model, componentTypename, name) {  
00021 }
```

:

8.128.2.2 ~SourceModelComponent() `virtual SourceModelComponent::~SourceModelComponent ()`
[virtual], [default]

8.128.3 Member Function Documentation

```
8.128.3.1 _check() bool SourceModelComponent::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Reimplemented in [Create](#).

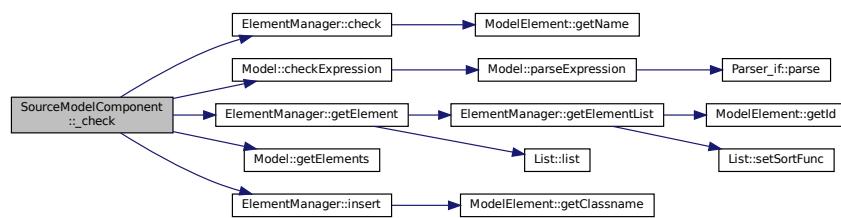
Definition at line 60 of file [SourceModelComponent.cpp](#).

```
00060
00061     /* include attributes needed */
00062     ElementManager* elements = _parentModel->getElements();
00063     std::vector<std::string> neededNames = {"Entity.ArrivalTime"};
00064     std::string neededName;
00065     for (unsigned int i = 0; i < neededNames.size(); i++) {
00066         neededName = neededNames[i];
00067         if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00068             Attribute* attrl = new Attribute(_parentModel, neededName);
00069             elements->insert(attrl);
00070         }
00071     }
00072     bool resultAll = true;
00073     resultAll &= _parentModel->getElements()->check(Util::TypeOf<EntityType>(), _entityType,
00074     "entitytype", errorMessage);
00075     resultAll &= _parentModel->checkExpression(this->_timeBetweenCreationsExpression, "time between
00076     creations", errorMessage);
00077     return resultAll;
00078 }
```

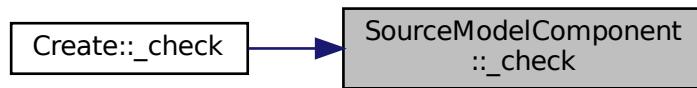
References [_entityType](#), [ModelElement::_parentModel](#), [_timeBetweenCreationsExpression](#), [ElementManager::check\(\)](#), [Model::checkExpression\(\)](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), and [ElementManager::insert\(\)](#).

Referenced by [Create::_check\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.128.3.2 `_initBetweenReplications()` void SourceModelComponent::_initBetweenReplications ()
 [protected], [virtual]

Reimplemented from [ModelElement](#).

Reimplemented in [Create](#).

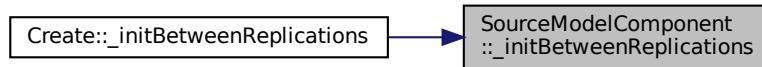
Definition at line 44 of file [SourceModelComponent.cpp](#).

```
00044
00045     this->_entitiesCreatedSoFar = 0;
00046 }
```

References [_entitiesCreatedSoFar](#).

Referenced by [Create::_initBetweenReplications\(\)](#).

Here is the caller graph for this function:



8.128.3.3 `_loadInstance()` bool SourceModelComponent::_loadInstance (std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelComponent](#).

Reimplemented in [Create](#).

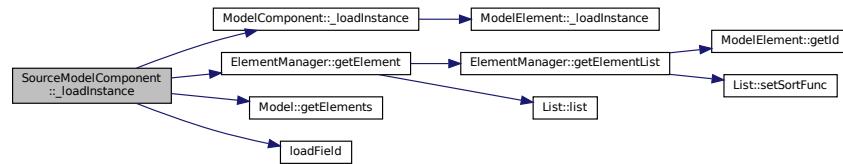
Definition at line 30 of file [SourceModelComponent.cpp](#).

```
00030
00031     bool res = ModelComponent::_loadInstance(fields);
00032     if (res) {
00033         this->_entitiesPerCreation = std::stoi(loadField(fields, "entitiesPerCreation", "1"));
00034         this->_firstCreation = std::stod(loadField(fields, "firstCreation", "0.0"));
00035         this->_timeBetweenCreationsExpression = (*fields->find("timeBetweenCreations")).second;
00036         this->_timeBetweenCreationsTimeUnit = static_cast<Util::TimeUnit>(std::stoi(loadField(fields,
00037             "timeBetweenCreationsTimeUnit", std::to_string(static_cast<int>(Util::TimeUnit::second)))));
00038         this->_maxCreationsExpression = loadField(fields, "maxCreations",
00039             std::to_string(std::numeric_limits<unsigned int>::max()));
00040         std::string entityTypename = (*fields->find("entityTypename")).second;
00041         this->_entityType = dynamic_cast<EntityType*>
00042             (_parentModel->getElements()->getElement(Util::TypeOf<EntityType>(), entityTypename));
00043     }
00044     return res;
00045 }
```

References [_entitiesPerCreation](#), [_entityType](#), [_firstCreation](#), [ModelComponent::_loadInstance\(\)](#), [_maxCreationsExpression](#), [ModelElement::_parentModel](#), [_timeBetweenCreationsExpression](#), [_timeBetweenCreationsTimeUnit](#), [ElementManager::getElement\(\)](#), [Model::getElements\(\)](#), [loadField\(\)](#), and [Util::second](#).

Referenced by [Create::_loadInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.128.3.4 `_saveInstance()` `std::map< std::string, std::string > * SourceModelComponent::_saveInstance() [protected], [virtual]`

Reimplemented from [ModelComponent](#).

Reimplemented in [Create](#).

Definition at line 48 of file [SourceModelComponent.cpp](#).

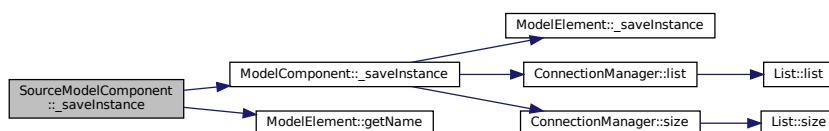
```

00048
00049     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00050     {
00051         if (_entitiesPerCreation != 1) fields->emplace("entitiesPerCreation",
00052             std::to_string(this->_entitiesPerCreation));
00053         if (_firstCreation != 0.0) fields->emplace("firstCreation", std::to_string(this->_firstCreation));
00054         fields->emplace("timeBetweenCreations", "\"" + this->_timeBetweenCreationsExpression + "\"");
00055         if (_timeBetweenCreationsTimeUnit != Util::TimeUnit::second)
00056             fields->emplace("timeBetweenCreationsTimeUnit", std::to_string(static_cast<int>
00057                 (this->_timeBetweenCreationsTimeUnit)));
00058         if (_maxCreationsExpression != std::to_string(std::numeric_limits<unsigned int>::max()))
00059             fields->emplace("maxCreations", "\"" + this->_maxCreationsExpression + "\"");
00060         fields->emplace("entityTypename", (this->_entityType->getName())); // save the name
00061         //fields->emplace("collectStatistics", std::to_string(this->_collectStatistics));
00062     }
  
```

References `_entitiesPerCreation`, `_entityType`, `_firstCreation`, `_maxCreationsExpression`, `ModelComponent::_saveInstance()`, `_timeBetweenCreationsExpression`, `_timeBetweenCreationsTimeUnit`, `ModelElement::getName()`, and `Util::second`.

Referenced by [Create::_saveInstance\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.128.3.5 **getEntitiesCreated()** `unsigned int SourceModelComponent::getEntitiesCreated() const`

Definition at line 126 of file [SourceModelComponent.cpp](#).

```

00126
00127     return _entitiesCreatedSoFar;
00128 }
  
```

References [_entitiesCreatedSoFar](#).

8.128.3.6 **getEntitiesPerCreation()** `unsigned int SourceModelComponent::getEntitiesPerCreation() const`

Definition at line 139 of file [SourceModelComponent.cpp](#).

```

00139
00140     return _entitiesPerCreation;
00141 }
  
```

References [_entitiesPerCreation](#).

Referenced by [Create::Create\(\)](#).

Here is the caller graph for this function:



8.128.3.7 **getEntityType()** `EntityType * SourceModelComponent::getEntityType() const`

Definition at line 98 of file [SourceModelComponent.cpp](#).

```

00098
00099     return _entityType;
00100 }
  
```

References [_entityType](#).

8.128.3.8 getFirstCreation() double SourceModelComponent::getFirstCreation () const

Definition at line 82 of file [SourceModelComponent.cpp](#).

```
00082     return _firstCreation;
00083 }
00084 }
```

References [_firstCreation](#).

8.128.3.9 getMaxCreations() std::string SourceModelComponent::getMaxCreations () const

Definition at line 122 of file [SourceModelComponent.cpp](#).

```
00122 {
00123     return _maxCreationsExpression;
00124 }
```

References [_maxCreationsExpression](#).

8.128.3.10 getTimeBetweenCreationsExpression() std::string SourceModelComponent::getTimeBetweenCreationsExpression () const

Definition at line 114 of file [SourceModelComponent.cpp](#).

```
00114 {
00115     return _timeBetweenCreationsExpression;
00116 }
```

References [_timeBetweenCreationsExpression](#).

8.128.3.11 getTimeUnit() Util::TimeUnit SourceModelComponent::getTimeUnit () const

Definition at line 106 of file [SourceModelComponent.cpp](#).

```
00106 {
00107     return this->_timeBetweenCreationsTimeUnit;
00108 }
```

References [_timeBetweenCreationsTimeUnit](#).

8.128.3.12 setEntitiesCreated() void SourceModelComponent::setEntitiesCreated (unsigned int _entitiesCreated)

Definition at line 130 of file [SourceModelComponent.cpp](#).

```
00130 {
00131     this->_entitiesCreatedSoFar = _entitiesCreated;
00132 }
```

References [_entitiesCreatedSoFar](#).

8.128.3.13 setEntitiesPerCreation() void SourceModelComponent::setEntitiesPerCreation (unsigned int _entitiesPerCreation)

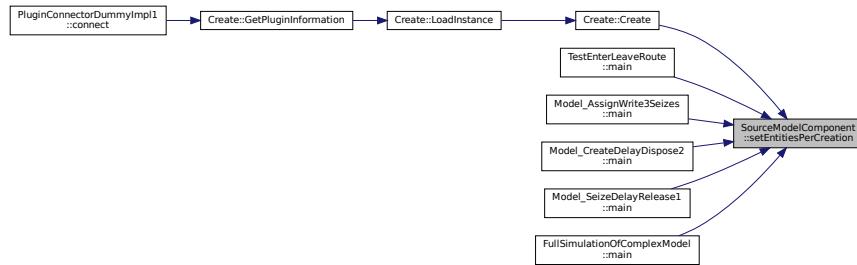
Definition at line 134 of file [SourceModelComponent.cpp](#).

```
00134
00135     this->_entitiesPerCreation = _entitiesPerCreation;
00136     //this->_entitiesCreatedSoFar = _entitiesPerCreation; // that's because "entitiesPerCreation"
00137     entities are included in the future events list BEFORE replication starts (to initialize events list)
```

References [_entitiesPerCreation](#).

Referenced by [Create::Create\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.128.3.14 setEntityType() void SourceModelComponent::setEntityType (EntityType * _entityType)

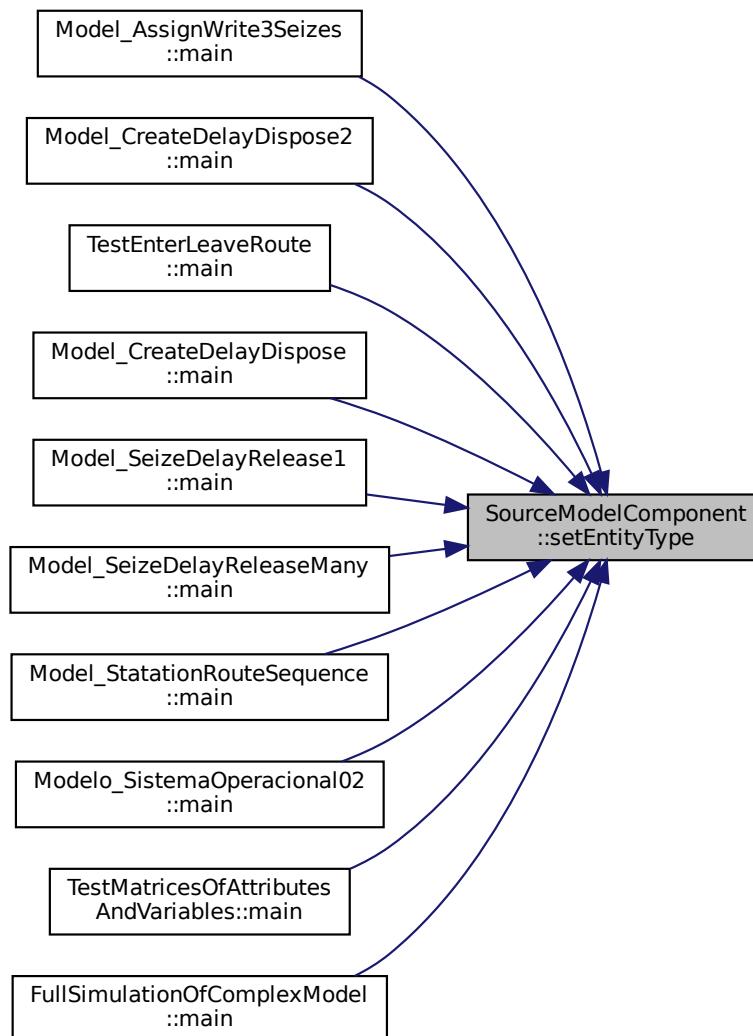
Definition at line 94 of file [SourceModelComponent.cpp](#).

```
00094
00095     _entityType = entityType;
00096 }
```

References [_entityType](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.128.3.15 setFirstCreation() void SourceModelComponent::setFirstCreation (double _firstCreation)

Definition at line 78 of file [SourceModelComponent.cpp](#).

```

00078
00079     this->_firstCreation = _firstCreation;
00080 }
```

References [_firstCreation](#).

Referenced by [Model_SeizeDelayRelease1::main\(\)](#).

Here is the caller graph for this function:



8.128.3.16 setMaxCreations() void SourceModelComponent::setMaxCreations (std::string _maxCreationsExpression)

Definition at line 118 of file [SourceModelComponent.cpp](#).

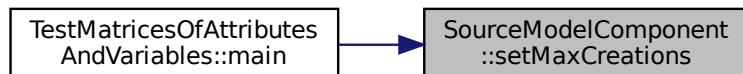
```

00118
00119     this->_maxCreationsExpression = _maxCreationsExpression;
00120 }
```

References [_maxCreationsExpression](#).

Referenced by [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the caller graph for this function:



8.128.3.17 setTimeBetweenCreationsExpression() void SourceModelComponent::setTimeBetweenCreationsExpression (std::string _timeBetweenCreations)

Definition at line 110 of file [SourceModelComponent.cpp](#).

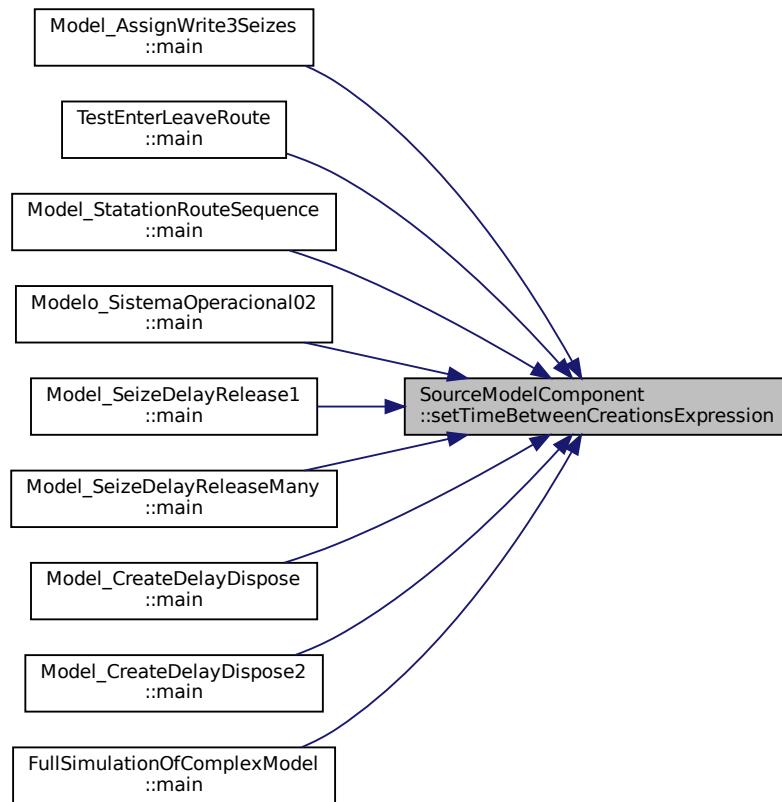
```

00110
00111     this->_timeBetweenCreationsExpression = _timeBetweenCreations;
00112 }
```

References [_timeBetweenCreationsExpression](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_StatationRouteSequence::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_CreateDelayDispose::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.128.3.18 setTimeUnit() void SourceModelComponent::setTimeUnit (Util::TimeUnit _timeUnit)

Definition at line 102 of file [SourceModelComponent.cpp](#).

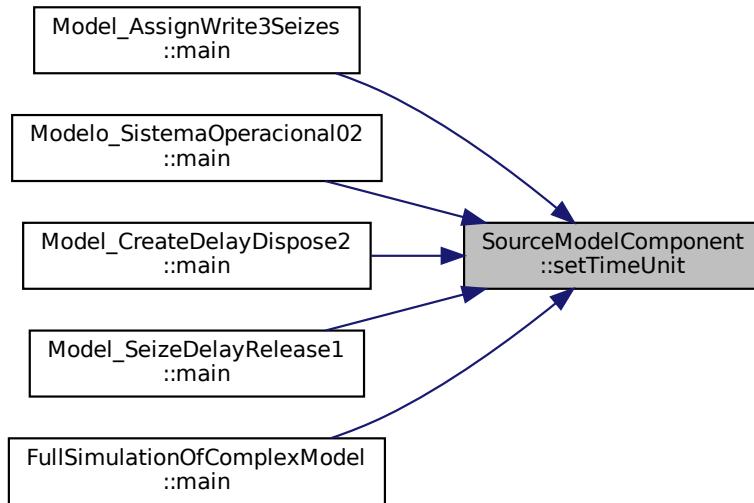
```

00102
00103     this->_timeBetweenCreationsTimeUnit = _timeUnit;
00104 }
```

References [_timeBetweenCreationsTimeUnit](#).

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



8.128.3.19 `show()` `std::string SourceModelComponent::show () [virtual]`

Reimplemented from [ModelComponent](#).

Reimplemented in [Create](#).

Definition at line 23 of file [SourceModelComponent.cpp](#).

```

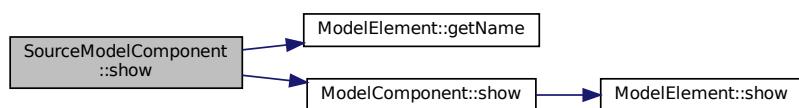
00023     {
00024         std::string text = ModelComponent::show() +
00025             ",entityType=\"" + _entityType->getName() + "\"" +
00026             ",firstCreation=\"" + std::to_string(_firstCreation);
00027         return text;
00028     }

```

References [_entityType](#), [_firstCreation](#), [ModelElement::getName\(\)](#), and [ModelComponent::show\(\)](#).

Referenced by [Create::show\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.128.4 Member Data Documentation

8.128.4.1 `_entitiesCreatedSoFar` `unsigned int SourceModelComponent::_entitiesCreatedSoFar = 0` [protected]

Definition at line 61 of file [SourceModelComponent.h](#).

Referenced by [Create::_execute\(\)](#), [_initBetweenReplications\(\)](#), [getEntitiesCreated\(\)](#), and [setEntitiesCreated\(\)](#).

8.128.4.2 `_entitiesPerCreation` `unsigned int SourceModelComponent::_entitiesPerCreation = 1` [protected]

Definition at line 57 of file [SourceModelComponent.h](#).

Referenced by [Create::_execute\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getEntitiesPerCreation\(\)](#), and [setEntitiesPerCreation\(\)](#).

8.128.4.3 `_entityType` `EntityType* SourceModelComponent::_entityType` [protected]

Definition at line 55 of file [SourceModelComponent.h](#).

Referenced by [_check\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getEntityType\(\)](#), [setEntityType\(\)](#), and [show\(\)](#).

8.128.4.4 `_firstCreation` `double SourceModelComponent::_firstCreation = 0.0` [protected]

Definition at line 56 of file [SourceModelComponent.h](#).

Referenced by [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getFirstCreation\(\)](#), [setFirstCreation\(\)](#), and [show\(\)](#).

8.128.4.5 `_maxCreationsExpression` `std::string SourceModelComponent::_maxCreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max())` [protected]

Definition at line 58 of file [SourceModelComponent.h](#).

Referenced by [Create::_execute\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getMaxCreations\(\)](#), and [setMaxCreations\(\)](#).

8.128.4.6 `_timeBetweenCreationsExpression` `std::string SourceModelComponent::_timeBetweenCreationsExpression = "EXPO(1)"` [protected]

Definition at line 59 of file [SourceModelComponent.h](#).

Referenced by [_check\(\)](#), [Create::_execute\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getTimeBetweenCreationsExpression\(\)](#), and [setTimeBetweenCreationsExpression\(\)](#).

8.128.4.7 `_timeBetweenCreationsTimeUnit` `Util::TimeUnit SourceModelComponent::_timeBetweenCreationsTimeUnit = Util::TimeUnit::second` [protected]

Definition at line 60 of file [SourceModelComponent.h](#).

Referenced by [Create::_execute\(\)](#), [_loadInstance\(\)](#), [_saveInstance\(\)](#), [getTimeUnit\(\)](#), and [setTimeUnit\(\)](#).

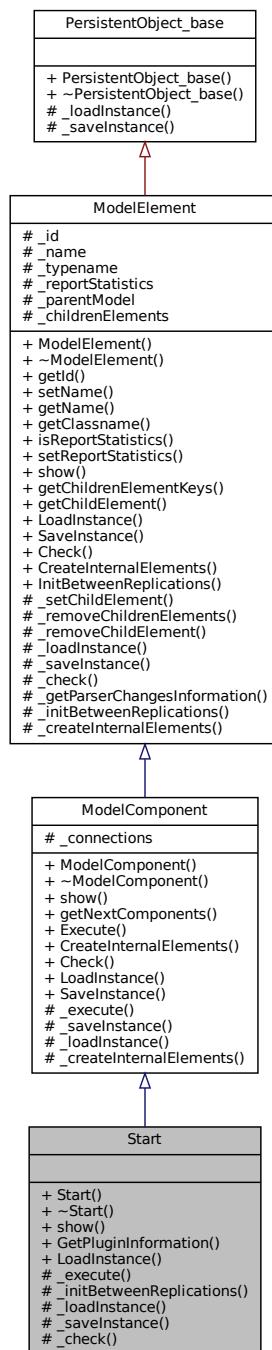
The documentation for this class was generated from the following files:

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

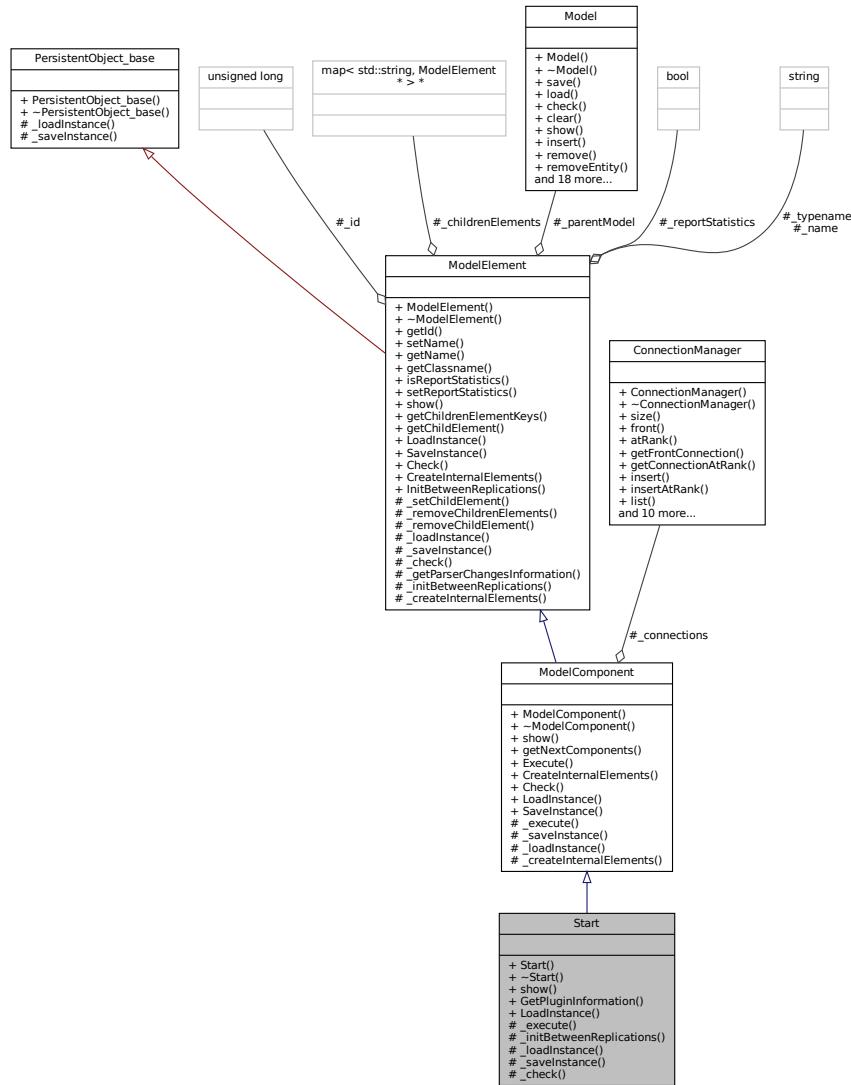
8.129 Start Class Reference

```
#include <Start.h>
```

Inheritance diagram for Start:



Collaboration diagram for Start:



Public Member Functions

- **Start** (`Model` *model, `std::string` name="")
- virtual **~Start** ()=default
- virtual `std::string` **show** ()

Static Public Member Functions

- static `PluginInformation` * **GetPluginInformation** ()
- static `ModelComponent` * **LoadInstance** (`Model` *model, `std::map< std::string, std::string >` *fields)

Protected Member Functions

- virtual void **_execute** (`Entity` *entity)
- virtual void **_initBetweenReplications** ()
- virtual bool **_loadInstance** (`std::map< std::string, std::string >` *fields)
- virtual `std::map< std::string, std::string >` * **_saveInstance** ()
- virtual bool **_check** (`std::string` *errorMessage)

Additional Inherited Members

8.129.1 Detailed Description

Start module DESCRIPTION The **Start** module changes the status of a conveyor from inactive to active. The conveyor may have been deactivated from either the **Stop** module or by initially being set to inactive at the start of the simulation. The velocity of the conveyor may be changed permanently when the conveyor is started. TYPICAL USES **Start** a bottling conveyor after scheduled maintenance **Start** a baggage claim conveyor when bags have arrived PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor to start. Velocity Speed of the conveyor once it begins to operate. This value will change the speed of the conveyor permanently, until it is changed in another module. Units Velocity time units.

Definition at line 39 of file [Start.h](#).

8.129.2 Constructor & Destructor Documentation

8.129.2.1 Start() `Start::Start (`
 `Model * model,`
 `std::string name = "")`

Definition at line 18 of file [Start.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Start>(), name) {  
00019 }
```

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

Here is the caller graph for this function:



8.129.2.2 ~Start() `virtual Start::~Start () [virtual], [default]`

8.129.3 Member Function Documentation

```
8.129.3.1 _check() bool Start::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Start.cpp](#).

```
00057                                     {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
```

```
8.129.3.2 _execute() void Start::_execute (
    Entity * entity ) [protected], [virtual]
```

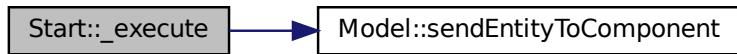
Implements [ModelComponent](#).

Definition at line 35 of file [Start.cpp](#).

```
00035                                     {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



```
8.129.3.3 _initBetweenReplications() void Start::_initBetweenReplications( ) [protected],
[virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Start.cpp](#).

```
00048                                     {
00049 }
```

```
8.129.3.4 _loadInstance() bool Start::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

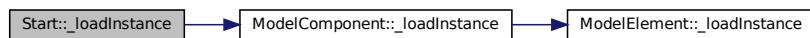
Definition at line 40 of file [Start.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

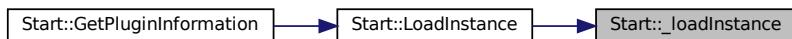
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.129.3.5 _saveInstance() std::map< std::string, std::string > * Start::_saveInstance ( )
[protected], [virtual]
```

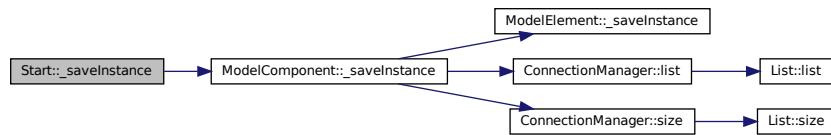
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Start.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.129.3.6 GetPluginInformation() `PluginInformation * Start::GetPluginInformation () [static]`

Definition at line 63 of file `Start.cpp`.

```
00063                                         {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Start>(), &Start::LoadInstance);
00065     // ...
00066     return info;
00067 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.129.3.7 LoadInstance() `ModelComponent * Start::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

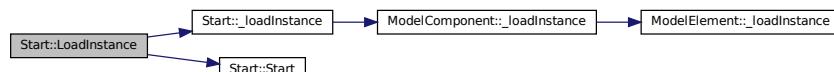
Definition at line 25 of file `Start.cpp`.

```
00025                                         {
00026     Start* newComponent = new Start(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
```

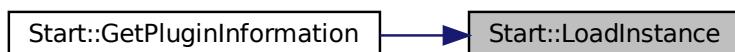
References `_loadInstance()`, and `Start()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.129.3.8 show() std::string Start::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Start.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



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

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

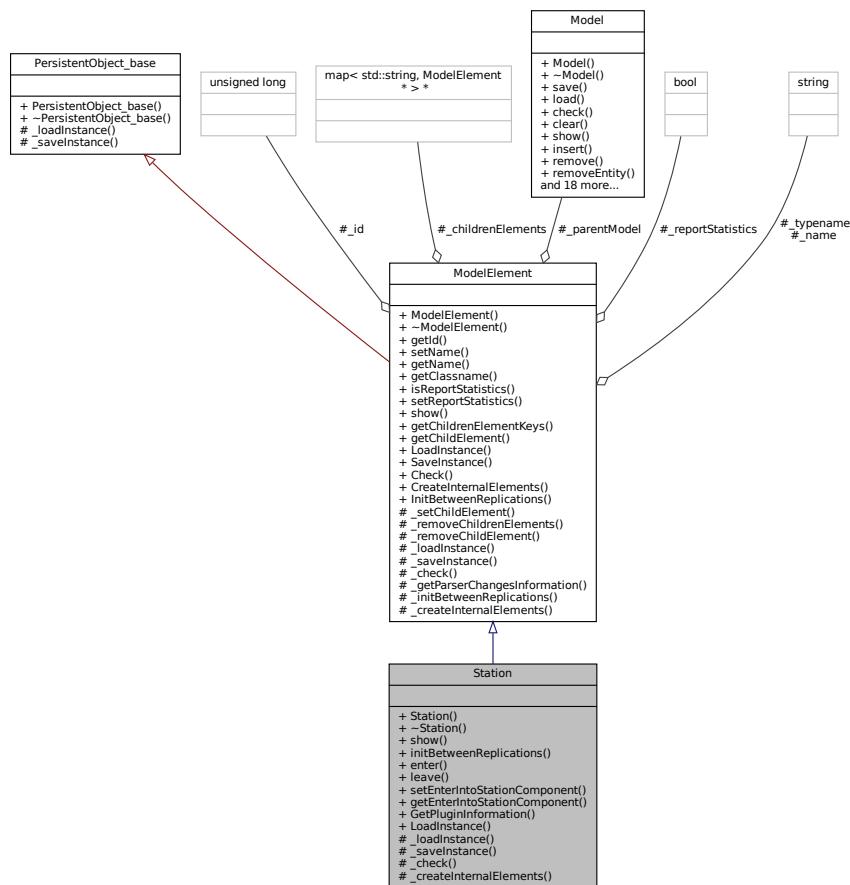
8.130 Station Class Reference

```
#include <Station.h>
```

Inheritance diagram for Station:



Collaboration diagram for Station:



Public Member Functions

- `Station (Model *model, std::string name="")`
- `virtual ~Station ()`
- `virtual std::string show ()`
- `void initBetweenReplications ()`
- `void enter (Entity *entity)`
- `void leave (Entity *entity)`
- `void setEnterIntoStationComponent (ModelComponent *_enterIntoStationComponent)`
- `ModelComponent * getEnterIntoStationComponent () const`

Static Public Member Functions

- `static PluginInformation * GetPluginInformation ()`
- `static ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)
- virtual void [_createInternalElements](#) ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Additional Inherited Members

8.130.1 Detailed Description

Definition at line 64 of file [Station.h](#).

8.130.2 Constructor & Destructor Documentation

8.130.2.1 Station() Station::Station (

```
Model * model,
std::string name = "")
```

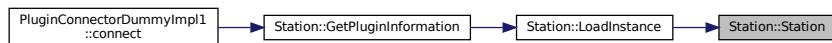
Definition at line 19 of file [Station.cpp](#).

```
00019 : ModelElement(model, Util::TypeOf<Station>(), name) {
```

```
00020 }
```

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

Here is the caller graph for this function:



8.130.2.2 ~Station() Station::~Station () [virtual]

Definition at line 22 of file [Station.cpp](#).

```
00022 {
00023     //__parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), __cstatNumberInStation);
00024     //__parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), __cstatTimeInStation);
00025 }
```

8.130.3 Member Function Documentation

8.130.3.1 `_check()` `bool Station::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

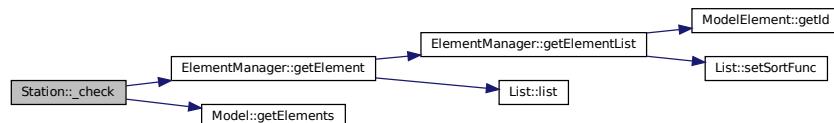
Definition at line 105 of file [Station.cpp](#).

```
00105                                     {
00106     /* include attributes needed */
00107     std::vector<std::string> neededNames = {"Entity.Station"};
00108     neededNames.insert(neededNames.begin(), "Entity.ArrivalAt" + this->getName());
00109     std::string neededName;
00110     for (unsigned int i = 0; i < neededNames.size(); i++) {
00111         neededName = neededNames[i];
00112         if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr)
00113             new Attribute(_parentModel, neededName);
00114     }
00115 }
00116 //  

00117 return true;
00118 }
```

References [ModelElement::_parentModel](#), [ElementManager::getElement\(\)](#), and [Model::getElements\(\)](#).

Here is the call graph for this function:



8.130.3.2 `_createInternalElements()` `void Station::_createInternalElements () [protected], [virtual]`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression

Reimplemented from [ModelElement](#).

Definition at line 120 of file [Station.cpp](#).

```
00120                                     {
00121     if (_reportStatistics) {
00122         if (_cstatNumberInStation == nullptr) {
00123             _cstatNumberInStation = new StatisticsCollector(_parentModel, _name + "." +
00124 "NumberInStation", this);
00125             _cstatTimeInStation = new StatisticsCollector(_parentModel, _name + "." + "TimeInStation",
00126 this);
00127             _childrenElements->insert({"NumberInStation", _cstatNumberInStation});
00128             _childrenElements->insert({"TimeInStation", _cstatTimeInStation});
00129         //
```

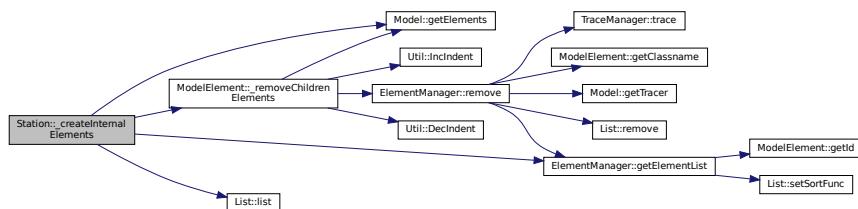
```

00128         // include StatisticsCollector needed in EntityType
00129         std::list<ModelElement*>* enttypes =
00130         _parentModel->getElements()->getElementsList(Util::TypeOf<EntityType>())->list();
00131         for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end();
00132             it++) {
00133             if ((*it)->isReportStatistics())
00134                 static_cast<EntityType*> ((*it))->addGetStatisticsCollector((*it)->getName() +
00135                                         ".TimeInStations"); // force create this CStat before simulation starts
00136         }
00137     } else
00138     if (_cstatNumberInStation != nullptr) {
00139         _removeChildrenElements();
00140     }

```

References [ModelElement::_childrenElements](#), [ModelElement::_name](#), [ModelElement::_parentModel](#), [ModelElement::_removeChildrenElements](#), [ModelElement::_reportStatistics](#), [ElementManager::getElementsList\(\)](#), [Model::getElements\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.130.3.3 `_loadInstance()` `bool Station::_loadInstance (std::map< std::string, std::string * fields > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 90 of file [Station.cpp](#).

```

00090
00091     bool res = ModelElement::_loadInstance(fields);
00092     if (res) {
00093         try {
00094             } catch (...) {
00095             }
00096     }
00097     return res;
00098 }

```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.130.3.4 `_saveInstance()` `std::map< std::string, std::string > * Station::_saveInstance ()`
[protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 100 of file [Station.cpp](#).

```

00100
00101     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00102     //Util::TypeOf<Station>());
00103     return fields;
00104 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.130.3.5 `enter()` `void Station::enter (`
`Entity * entity)`

Definition at line 42 of file [Station.cpp](#).

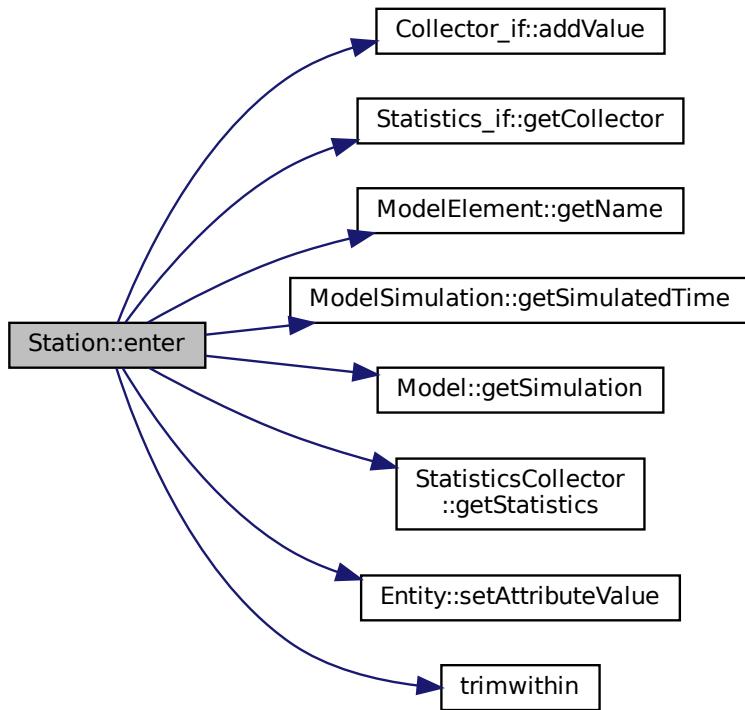
```

00042
00043     std::string attributeName = "Entity.ArrivalAt" + this->getName();
00044     trimwithin(attributeName);
00045     entity->setAttributeValue(attributeName, _parentModel->getSimulation()->getSimulatedTime());
00046     entity->setAttributeValue("Entity.Station", _id);
00047     _numberInStation++;
00048     if (_reportStatistics)
00049         this->_cstatNumberInStation->getStatistics()->getCollector()->addValue(_numberInStation);
00050 }
```

References [ModelElement::_id](#), [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [Collector_if::addValue\(\)](#), [Statistics_if::getCollector\(\)](#), [ModelElement::getName\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [Entity::setAttributeValue\(\)](#), and [trimwithin\(\)](#).

Referenced by [Enter::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.130.3.6 `getEnterIntoStationComponent()` `ModelComponent * Station::getEnterIntoStationComponent()` const

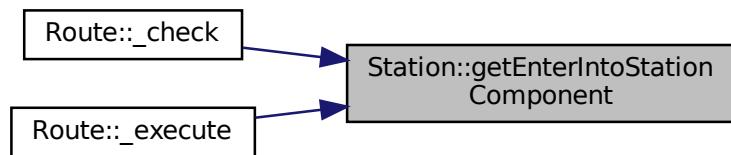
Definition at line 71 of file [Station.cpp](#).

```

00071
00072     return _enterIntoStationComponent;
00073 }
```

Referenced by [Route::_check\(\)](#), and [Route::_execute\(\)](#).

Here is the caller graph for this function:



8.130.3.7 GetPluginInformation() [PluginInformation * Station::GetPluginInformation \(\) \[static\]](#)

Definition at line 75 of file [Station.cpp](#).

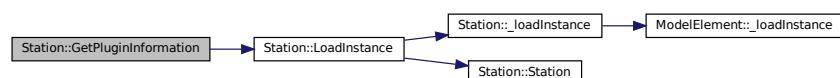
```

00075     {
00076     PluginInformation* info = new PluginInformation(Util::TypeOf<Station>(), &Station::LoadInstance);
00077     return info;
00078 }
```

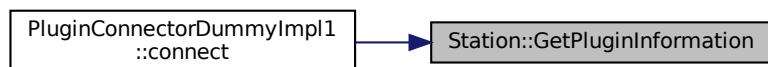
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



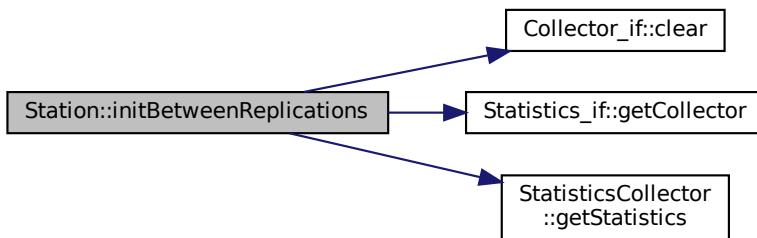
8.130.3.8 initBetweenReplications()

Definition at line 36 of file [Station.cpp](#).

```
00036     {
00037     _cstatNumberInStation->getStatistics()->getCollector()->clear();
00038     _cstatTimeInStation->getStatistics()->getCollector()->clear();
00039 }
00040 }
```

References [Collector_if::clear\(\)](#), [Statistics_if::getCollector\(\)](#), and [StatisticsCollector::getStatistics\(\)](#).

Here is the call graph for this function:



8.130.3.9 leave()

`void Station::leave (`

`Entity * entity)`

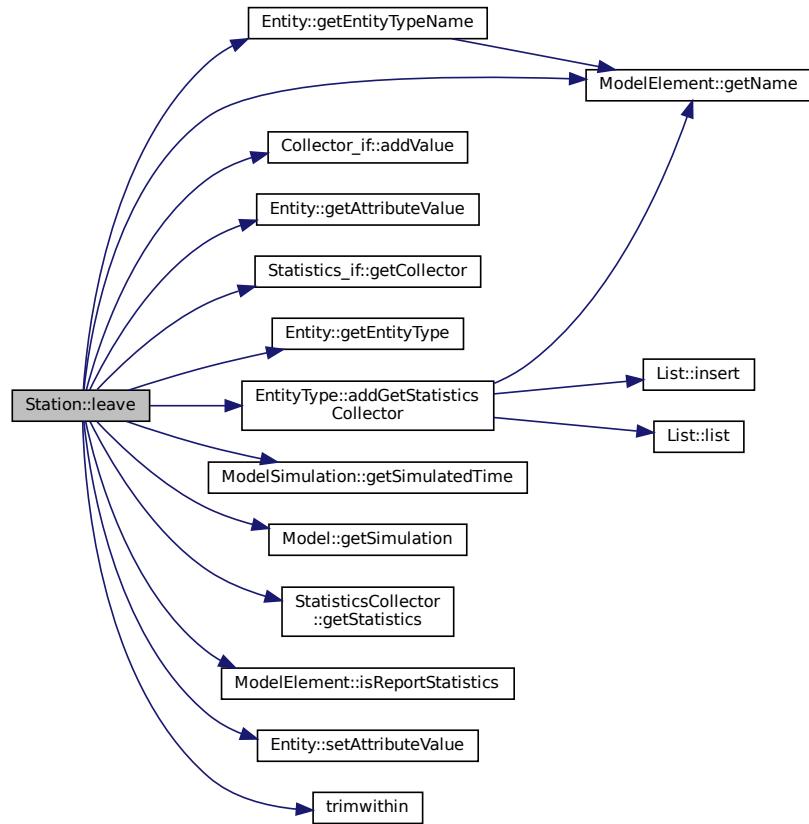
Definition at line 52 of file [Station.cpp](#).

```
00052     {
00053     std::string attributeName = "Entity.ArrivalAt" + this->getName();
00054     trimwithin(attributeName);
00055     double arrivalTime = entity->getAttributeValue(attributeName);
00056     double timeInStation = _parentModel->getSimulation()->getSimulatedTime() - arrivalTime;
00057     entity->setAttributeValue("Entity.Station", 0.0);
00058     _numberInStation--;
00059     if (_reportStatistics) {
00060         _cstatNumberInStation->getStatistics()->getCollector()->addValue(_numberInStation);
00061         _cstatTimeInStation->getStatistics()->getCollector()->addValue(timeInStation);
00062         if (entity->getEntityType()->isReportStatistics())
00063             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityTypeName() +
00064             ".TimeInStations")->getStatistics()->getCollector()->addValue(timeInStation); // \todo: should check
00065     }
00066 }
```

References [ModelElement::_parentModel](#), [ModelElement::_reportStatistics](#), [EntityType::addGetStatisticsCollector\(\)](#), [Collector_if::addValue\(\)](#), [Entity::getAttributeValue\(\)](#), [Statistics_if::getCollector\(\)](#), [Entity::getEntityType\(\)](#), [Entity::getEntityTypeName\(\)](#), [ModelElement::getName\(\)](#), [ModelSimulation::getSimulatedTime\(\)](#), [Model::getSimulation\(\)](#), [StatisticsCollector::getStatistics\(\)](#), [ModelElement::isReportStatistics\(\)](#), [Entity::setAttributeValue\(\)](#), and [trimwithin\(\)](#).

Referenced by [Leave::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.130.3.10 LoadInstance() `ModelElement * Station::LoadInstance (`

```
    Model * model,
    std::map< std::string, std::string *> * fields ) [static]
```

Definition at line 80 of file [Station.cpp](#).

```
00080
00081     Station* newElement = new Station(model);
00082     try {
00083         if (model->getEntityType() == "Station") {
00084             newElement = new Station(model);
00085         }
00086     } catch (...) {
00087         delete newElement;
00088         throw;
00089     }
00090 }
```

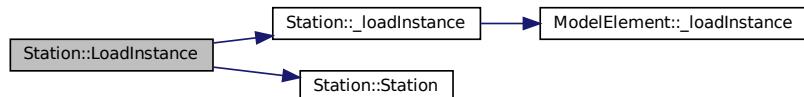
```

00083     newElement->_loadInstance(fields);
00084 } catch (const std::exception& e) {
00085 }
00086 }
00087 return newElement;
00088 }
```

References [_loadInstance\(\)](#), and [Station\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.130.3.11 setEnterIntoStationComponent() void Station::setEnterIntoStationComponent (ModelComponent * _enterIntoStationComponent)

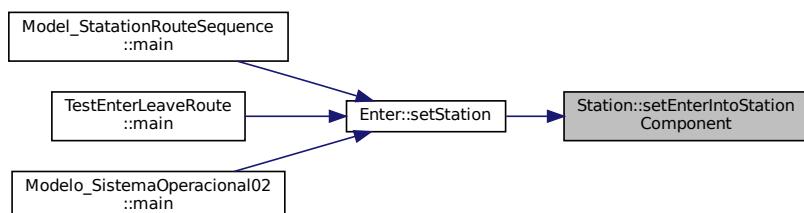
Definition at line 67 of file [Station.cpp](#).

```

00067
00068     this->_enterIntoStationComponent = _enterIntoStationComponent;
00069 }
```

Referenced by [Enter::setStation\(\)](#).

Here is the caller graph for this function:



8.130.3.12 show() std::string Station::show () [virtual]

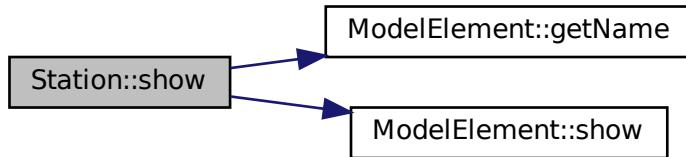
Reimplemented from [ModelElement](#).

Definition at line 27 of file [Station.cpp](#).

```
00027     {
00028         std::string msg = ModelElement::show() + ",enterIntoStationComponent=";
00029         if (_enterIntoStationComponent == nullptr)
00030             msg += "NULL";
00031         else
00032             msg += _enterIntoStationComponent->getName();
00033         return msg;
00034     }
```

References [ModelElement::getName\(\)](#), and [ModelElement::show\(\)](#).

Here is the call graph for this function:



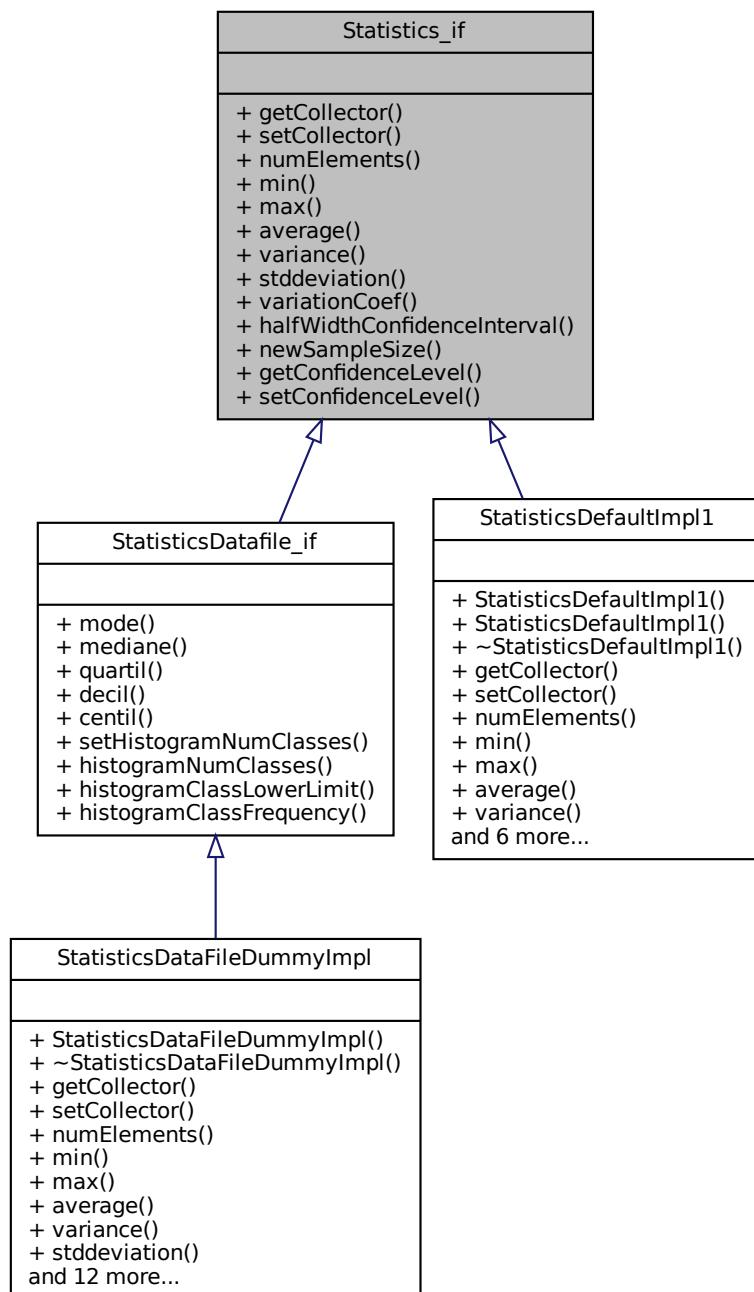
The documentation for this class was generated from the following files:

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

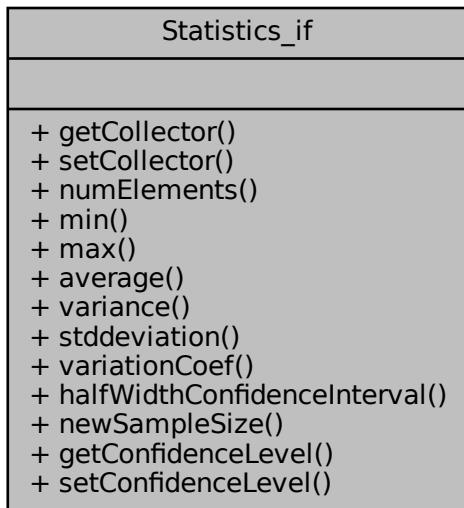
8.131 Statistics_if Class Reference

```
#include <Statistics_if.h>
```

Inheritance diagram for Statistics_if:



Collaboration 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 ()=0`
- virtual unsigned int `newSampleSize (double halfWidth)=0`
- virtual double `getConfidenceLevel ()=0`
- virtual void `setConfidenceLevel (double confidencelevel)=0`

8.131.1 Detailed Description

Interface for statistic 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.

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

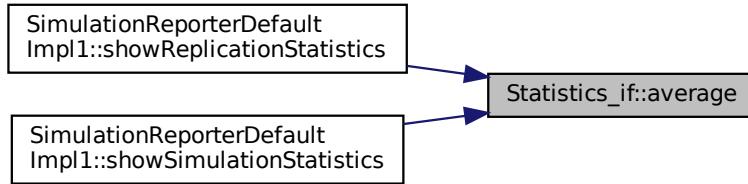
8.131.2 Member Function Documentation

8.131.2.1 average() virtual double Statistics_if::average () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

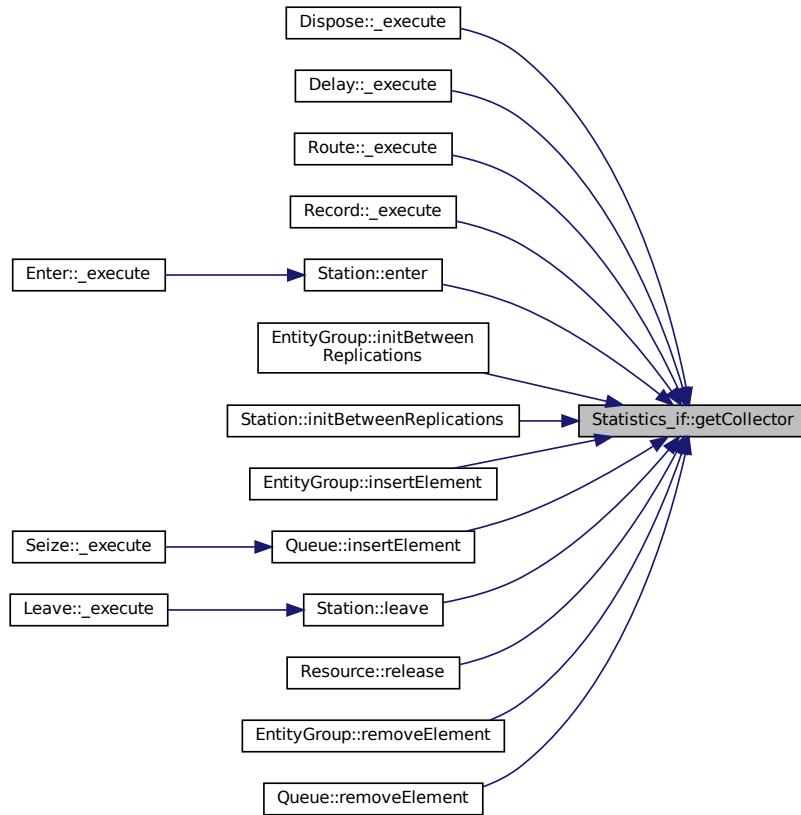
Here is the caller graph for this function:

**8.131.2.2 getCollector()** virtual Collector_if* Statistics_if::getCollector () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [Dispose::_execute\(\)](#), [Delay::_execute\(\)](#), [Route::_execute\(\)](#), [Record::_execute\(\)](#), [Station::enter\(\)](#), [EntityGroup::initBetweenReplications\(\)](#), [Station::initBetweenReplications\(\)](#), [EntityGroup::insertElement\(\)](#), [Queue::insertElement\(\)](#), [Station::leave\(\)](#), [Resource::release\(\)](#), [EntityGroup::removeElement\(\)](#), and [Queue::removeElement\(\)](#).

Here is the caller graph for this function:

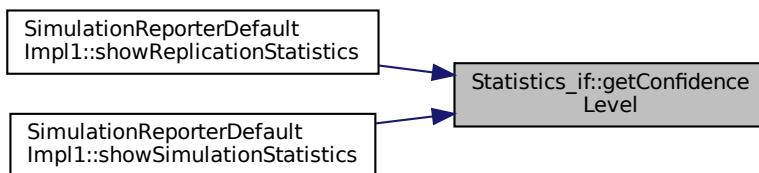


8.131.2.3 getConfidenceLevel() virtual double Statistics_if::getConfidenceLevel () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#)

Here is the caller graph for this function:

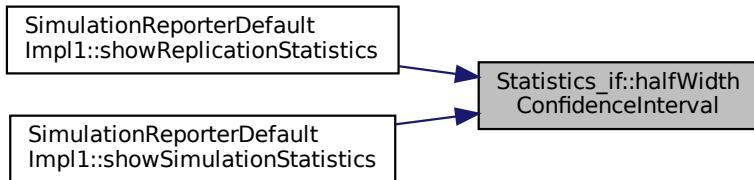


```
8.131.2.4 halfWidthConfidenceInterval() virtual double Statistics_if::halfWidthConfidence<-
Interval ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#)

Here is the caller graph for this function:

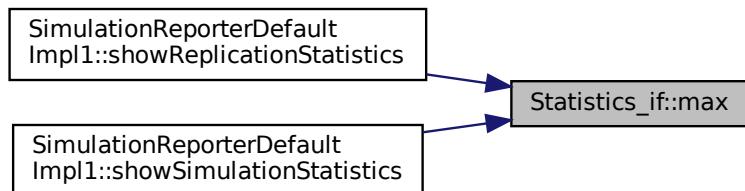


```
8.131.2.5 max() virtual double Statistics_if::max ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#)

Here is the caller graph for this function:

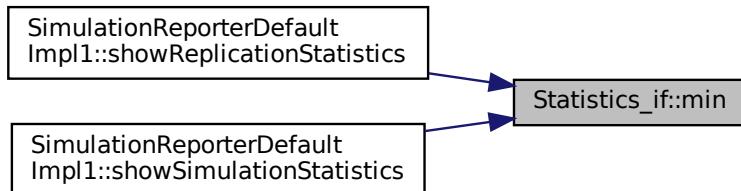


8.131.2.6 min() virtual double Statistics_if::min () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.131.2.7 newSampleSize() virtual unsigned int Statistics_if::newSampleSize (double halfWidth) [pure virtual]

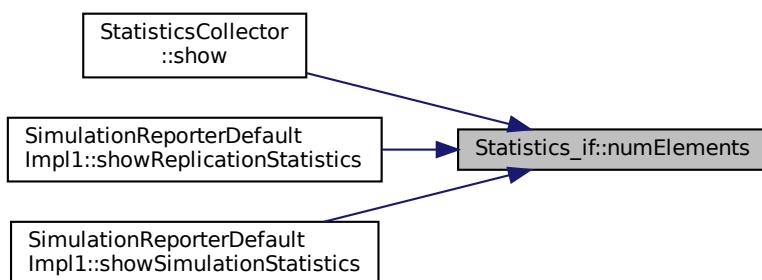
Implemented in [StatisticsDefaultImpl1](#).

8.131.2.8 numElements() virtual unsigned int Statistics_if::numElements () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [StatisticsCollector::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



```
8.131.2.9 setCollector() virtual void Statistics_if::setCollector (
    Collector_if * collector ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

```
8.131.2.10 setConfidenceLevel() virtual void Statistics_if::setConfidenceLevel (
    double confidencelevel ) [pure virtual]
```

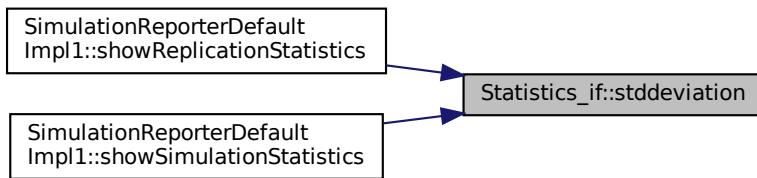
Implemented in [StatisticsDefaultImpl1](#).

```
8.131.2.11 stddeviation() virtual double Statistics_if::stddeviation () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:

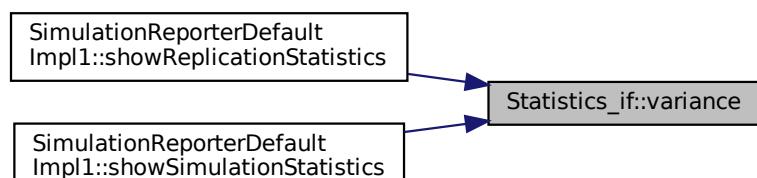


```
8.131.2.12 variance() virtual double Statistics_if::variance () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:

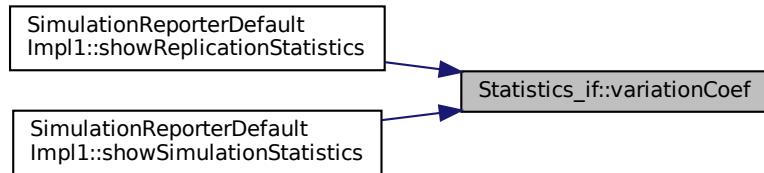


8.131.2.13 variationCoef() virtual double Statistics_if::variationCoef () [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



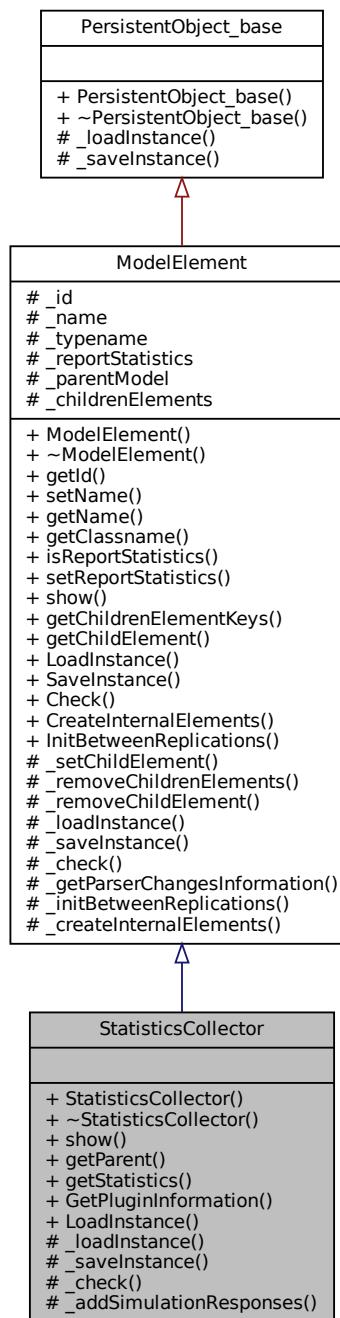
The documentation for this class was generated from the following file:

- [Statistics_if.h](#)

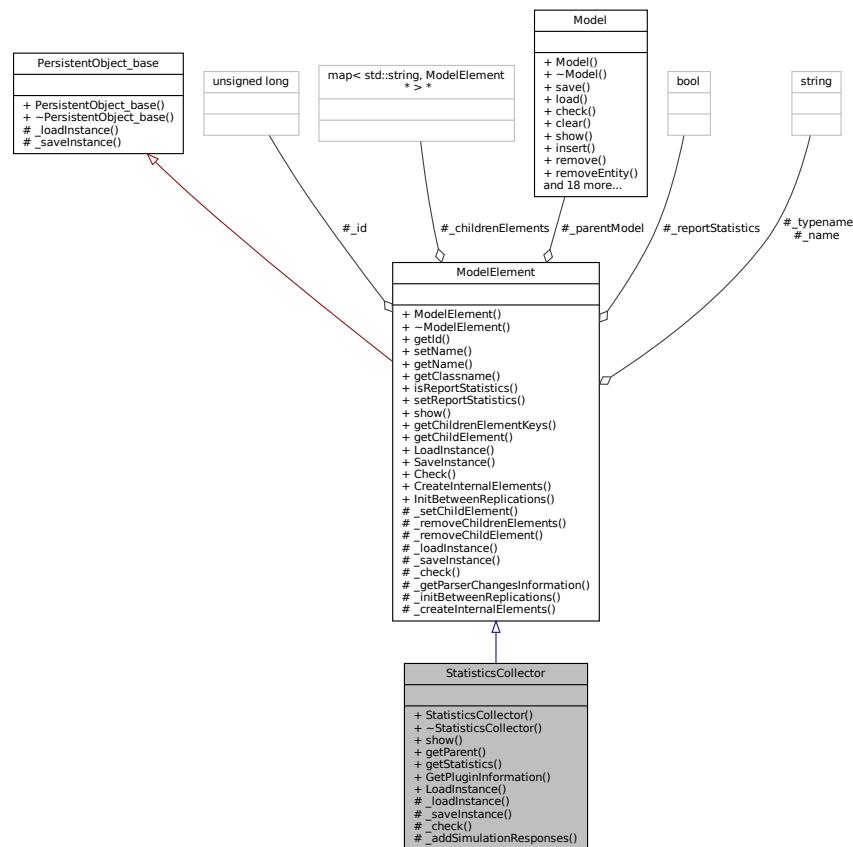
8.132 StatisticsCollector Class Reference

```
#include <StatisticsCollector.h>
```

Inheritance diagram for StatisticsCollector:



Collaboration diagram for StatisticsCollector:



Public Member Functions

- `StatisticsCollector (Model *model, std::string name="", ModelElement *parent=nullptr, bool insertIntoModel=true)`
 - `virtual ~StatisticsCollector ()=default`
 - `virtual std::string show ()`
 - `ModelElement * getParent () const`
 - `Statistics_if * getStatistics () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
 - static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool _loadInstance (std::map< std::string, std::string > *fields)
 - virtual std::map< std::string, std::string > * _saveInstance ()
 - virtual bool _check (std::string *errorMessage)
 - void addSimulationResponses ()

Additional Inherited Members

8.132.1 Detailed Description

The [StatisticsCollector](#) is the [ModelElement](#) responsible for collecting data from the model (using the Collector) and simultaneously keeping statistics updated (using the Statistics)

Definition at line 27 of file [StatisticsCollector.h](#).

8.132.2 Constructor & Destructor Documentation

```
8.132.2.1 StatisticsCollector() StatisticsCollector::StatisticsCollector (
    Model * model,
    std::string name = "",
    ModelElement * parent = nullptr,
    bool insertIntoModel = true )
```

Definition at line 22 of file [StatisticsCollector.cpp](#).

```
00022     : ModelElement(model, Util::TypeOf<StatisticsCollector>(), name, insertIntoModel) {
00023     _parent = parent;
00024     _initStaticsAndCollector();
00025     _addSimulationResponses();
00026 }
```

References [_addSimulationResponses\(\)](#).

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

Here is the call graph for this function:



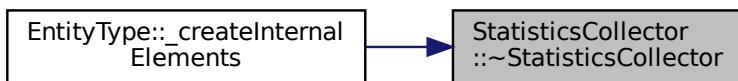
Here is the caller graph for this function:



8.132.2.2 ~StatisticsCollector() virtual StatisticsCollector::~StatisticsCollector () [virtual], [default]

Referenced by [EntityType::_createInternalElements\(\)](#).

Here is the caller graph for this function:



8.132.3 Member Function Documentation

8.132.3.1 _addSimulationResponses() void StatisticsCollector::_addSimulationResponses () [protected]

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

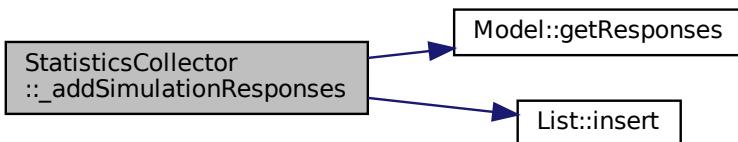
```

00028
00029     GetterMember<StatisticsClass> getterMemberAverage =
00030         DefineGetterMember<StatisticsClass>(static_cast<StatisticsClass*> (this->_statistics),
00031         &StatisticsClass::average);
00032     SimulationResponse* resp = new SimulationResponse(Util::TypeOf<StatisticsClass>(), _name +
00033         ".average", getterMemberAverage);
00034     _parentModel->getResponses()->insert(resp);
00035     // add the halfwidth as response
00036     GetterMember<StatisticsClass> getterMemberHalfWidth =
00037         DefineGetterMember<StatisticsClass>(static_cast<StatisticsClass*> (this->_statistics),
00038         &StatisticsClass::halfWidthConfidenceInterval);
00039     resp = new SimulationResponse(Util::TypeOf<StatisticsClass>(), /*parentName + ":" + */_name +
00040         ".halfWidth", getterMemberHalfWidth);
00041     _parentModel->getResponses()->insert(resp);
00042 }
```

References [ModelElement::_name](#), [ModelElement::_parentModel](#), [Model::getResponses\(\)](#), and [List< T >::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.132.3.2 `_check()` `bool StatisticsCollector::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 100 of file [StatisticsCollector.cpp](#).

```

00100
00101     return true;
00102 }
```

8.132.3.3 `_loadInstance()` `bool StatisticsCollector::_loadInstance (std::map< std::string, std::string * > * fields) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 81 of file [StatisticsCollector.cpp](#).

```

00081
00082     bool res = ModelElement::_loadInstance(fields);
00083     if (res) {
00084     }
00085     return res;
00086 }
```

References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.132.3.4 `_saveInstance()` `std::map< std::string, std::string > * StatisticsCollector::_saveInstance () [protected], [virtual]`

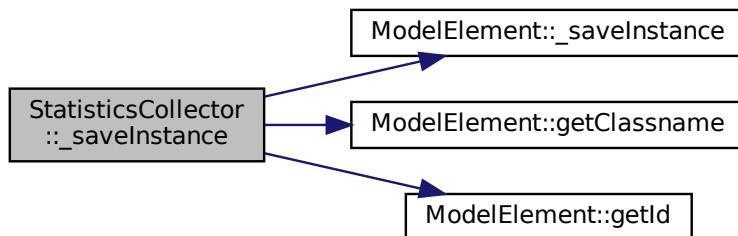
Reimplemented from [ModelElement](#).

Definition at line 88 of file [StatisticsCollector.cpp](#).

```
00088     std::map<std::string, std::string>* fields = ModelElement::_saveInstance ();
00089     //Util::TypeOf<StatisticsCollector>());
00090     std::string parentId = "", parentTypename = "";
00091     if (this->parent != nullptr) {
00092         parentId = std::to_string(_parent->getId ());
00093         parentTypename = _parent->getClassName ();
00094     }
00095     fields->emplace("parentTypename", parentTypename);
00096     fields->emplace("parentId", parentId);
00097     return fields;
00098 }
```

References [ModelElement::_saveInstance\(\)](#), [ModelElement::getClassName\(\)](#), and [ModelElement::getId\(\)](#).

Here is the call graph for this function:



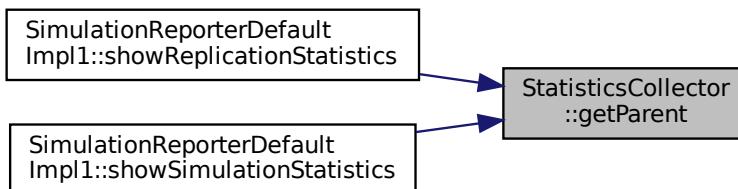
8.132.3.5 `getParent()` `ModelElement * StatisticsCollector::getParent () const`

Definition at line 57 of file [StatisticsCollector.cpp](#).

```
00057
00058     return _parent;
00059 }
```

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.132.3.6 GetPluginInformation() `PluginInformation * StatisticsCollector::GetPluginInformation () [static]`

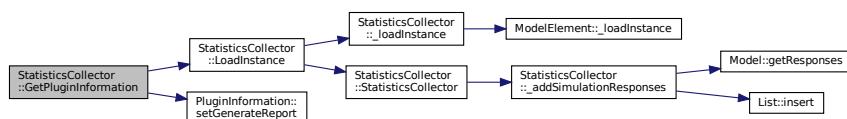
Definition at line 65 of file `StatisticsCollector.cpp`.

```
00065
00066     PluginInformation* info = new PluginInformation(Util::TypeOf<StatisticsCollector>(),
00067     &StatisticsCollector::LoadInstance);
00068     info->setGenerateReport(true);
00069 }
```

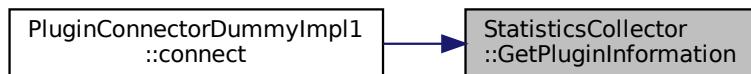
References `LoadInstance()`, and `PluginInformation::setGenerateReport()`.

Referenced by `PluginConnectorDummyImpl1::connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



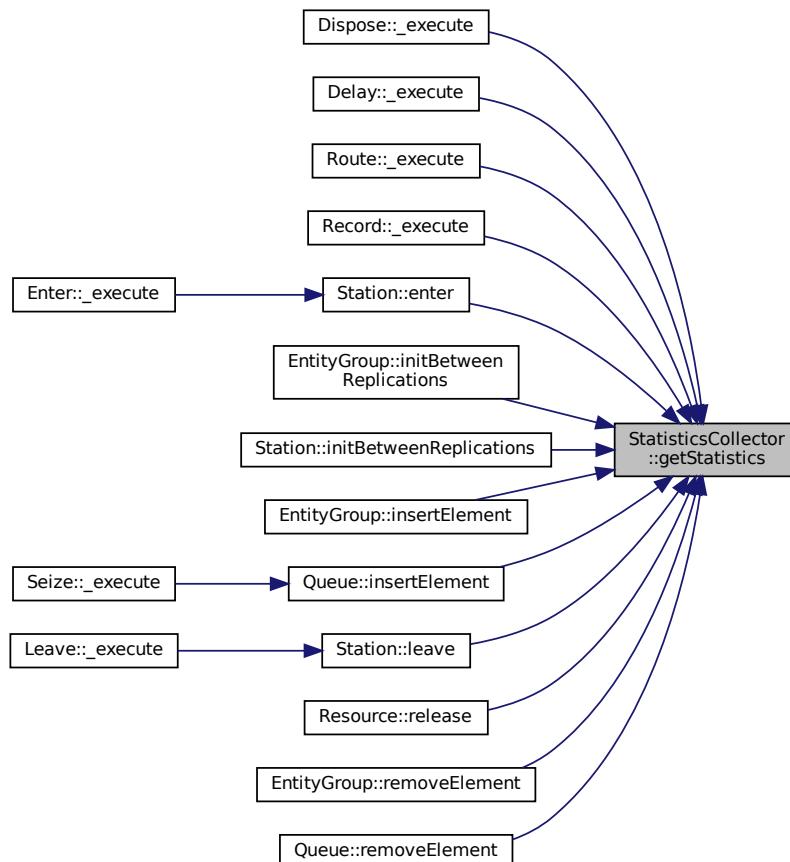
8.132.3.7 getStatistics() `Statistics_if * StatisticsCollector::getStatistics () const`

Definition at line 61 of file `StatisticsCollector.cpp`.

```
00061
00062     return _statistics;
00063 }
```

Referenced by `Dispose::_execute()`, `Delay::_execute()`, `Route::_execute()`, `Record::_execute()`, `Station::enter()`, `EntityGroup::initBetweenReplications()`, `Station::initBetweenReplications()`, `EntityGroup::insertElement()`, `Queue::insertElement()`, `Station::leave()`, `Resource::release()`, `EntityGroup::removeElement()`, and `Queue::removeElement()`.

Here is the caller graph for this function:



8.132.3.8 LoadInstance() `ModelElement * StatisticsCollector::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]`

Definition at line 71 of file `StatisticsCollector.cpp`.

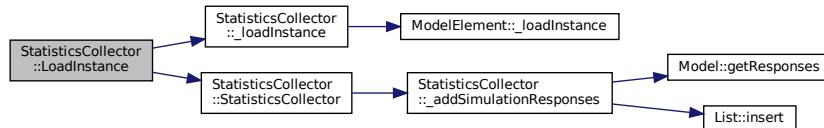
```

00071
00072     StatisticsCollector* newElement = new StatisticsCollector(model);
00073     try {
00074         newElement->_loadInstance(fields);
00075     } catch (const std::exception& e) {
00076     }
00077 }
00078     return newElement;
00079 }
```

References `_loadInstance()`, and `StatisticsCollector()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.132.3.9 show() std::string StatisticsCollector::show () [virtual]

Reimplemented from [ModelElement](#).

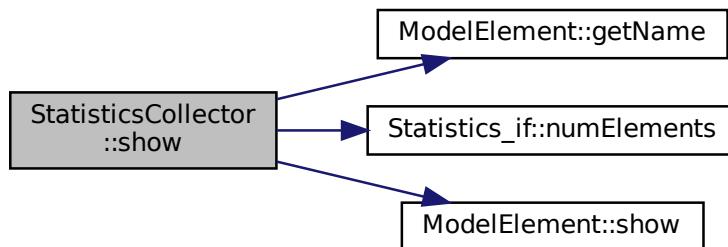
Definition at line 43 of file [StatisticsCollector.cpp](#).

```

00043     {
00044         std::string parentStr = "";
00045         if (_parent != nullptr) {
00046             try {
00047                 parentStr = _parent->getName();
00048             } catch (...) { // if parent changed or deleted, can cause seg fault
00049                 parentStr = "«INCONSISTENT»"; /* \todo: +*/
00050             }
00051         }
00052         return ModelElement::show() +
00053             ",parent=\"" + parentStr + "\""
00054             ",numElements=" + std::to_string(_statistics->numElements());
00055     }
  
```

References [ModelElement::getName\(\)](#), [Statistics_if::numElements\(\)](#), and [ModelElement::show\(\)](#).

Here is the call graph for this function:



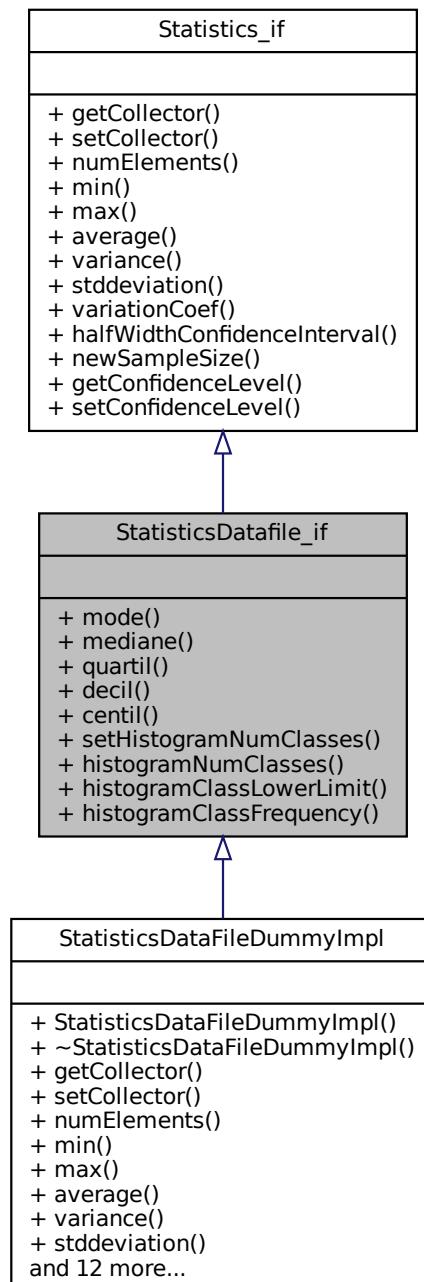
The documentation for this class was generated from the following files:

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

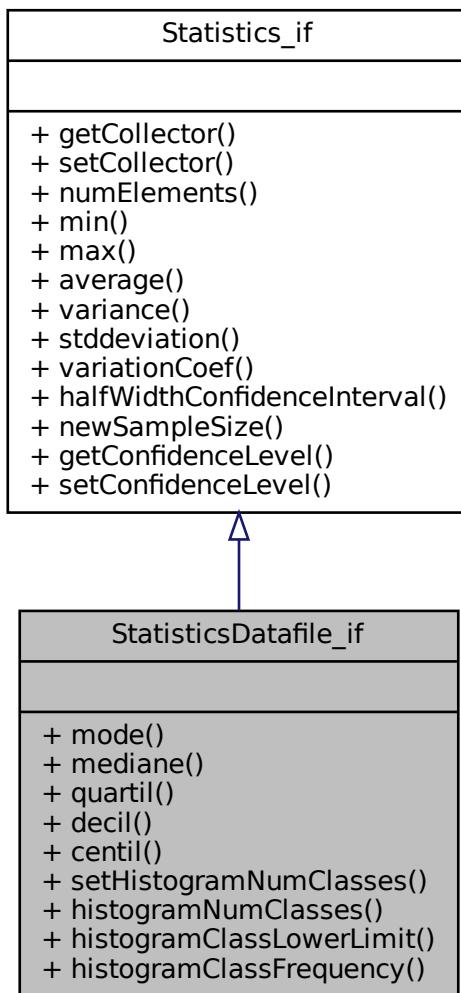
8.133 StatisticsDatafile_if Class Reference

```
#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`

8.133.1 Detailed Description

Definition at line 20 of file [StatisticsDataFile_if.h](#).

8.133.2 Member Function Documentation

8.133.2.1 centil() virtual double StatisticsDatafile_if::centil (unsigned short *num*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.2 decil() virtual double StatisticsDatafile_if::decil (unsigned short *num*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.3 histogramClassFrequency() virtual unsigned int StatisticsDatafile_if::histogramClassFrequency (unsigned short *classNum*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.4 histogramClassLowerLimit() virtual double StatisticsDatafile_if::histogramClassLowerLimit (unsigned short *classNum*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.5 histogramNumClasses() virtual unsigned short StatisticsDatafile_if::histogramNumClasses () [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.6 mediane() virtual double StatisticsDatafile_if::mediane () [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.7 mode() virtual double StatisticsDatafile_if::mode () [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.8 quartil() virtual double StatisticsDatafile_if::quartil (unsigned short *num*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

8.133.2.9 setHistogramNumClasses() virtual void StatisticsDatafile_if::setHistogramNumClasses (unsigned short *num*) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

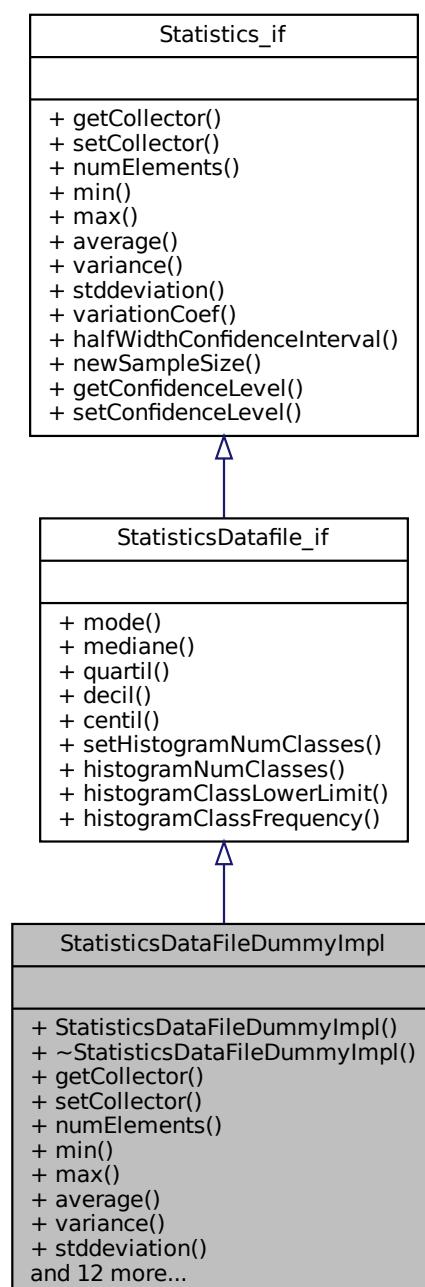
The documentation for this class was generated from the following file:

- [StatisticsDataFile_if.h](#)

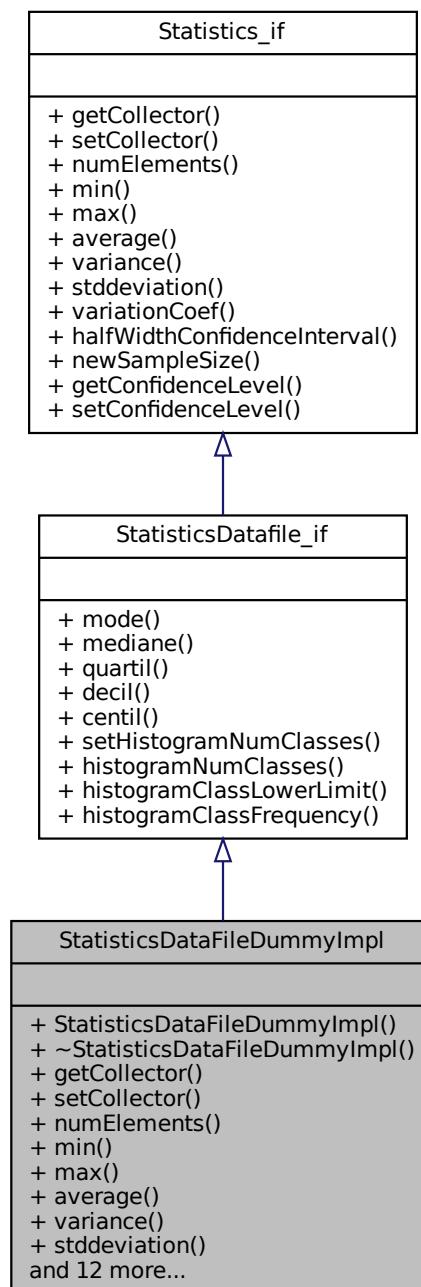
8.134 StatisticsDataFileDummyImpl Class Reference

```
#include <StatisticsDataFileDefaultImpl.h>
```

Inheritance diagram for StatisticsDataFileDummyImpl:



Collaboration diagram for StatisticsDataFileDummyImpl:



Public Member Functions

- `StatisticsDataFileDummyImpl ()`
- virtual `~StatisticsDataFileDummyImpl ()=default`
- 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)`

8.134.1 Detailed Description

Definition at line 19 of file [StatisticsDataFileDefaultImpl.h](#).

8.134.2 Constructor & Destructor Documentation

8.134.2.1 `StatisticsDataFileDummyImpl()` `StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl ()`

Definition at line 18 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00018
00019     _collector = new Traits<Statistics_if>::CollectorImplementation();
00020 }
```

8.134.2.2 `~StatisticsDataFileDummyImpl()` `virtual StatisticsDataFileDummyImpl::~StatisticsDataFileDummyImpl () [virtual], [default]`

8.134.3 Member Function Documentation

8.134.3.1 `average()` `double StatisticsDataFileDummyImpl::average () [virtual]`

Implements [Statistics_if](#).

Definition at line 34 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00034
00035
00036 }
```

8.134.3.2 centil() double StatisticsDataFileDummyImpl::centil (unsigned short num) [virtual]

Implements [StatisticsDatafile_if](#).

Definition at line 74 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00074
00075     return 0.0; // dummy
00076 }
```

8.134.3.3 decil() double StatisticsDataFileDummyImpl::decil (unsigned short num) [virtual]

Implements [StatisticsDatafile_if](#).

Definition at line 70 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00070
00071     return 0.0; // dummy
00072 }
```

8.134.3.4 getCollector() Collector_if * StatisticsDataFileDummyImpl::getCollector () [virtual]

Implements [Statistics_if](#).

Definition at line 93 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00093
00094     return this->_collector;
00095 }
```

8.134.3.5 halfWidthConfidenceInterval() double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval (double confidencelevel) [virtual]

Definition at line 58 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00058
00059     return 0.0; // dummy
00060 }
```

8.134.3.6 histogramClassFrequency() unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency (unsigned short classNum) [virtual]

Implements [StatisticsDatafile_if](#).

Definition at line 89 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00089
00090     return 0; // dummy
00091 }
```

```
8.134.3.7 histogramClassLowerLimit() double StatisticsDataFileDummyImpl::histogramClassLowerLimit ( unsigned short classNum ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 85 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00085 {  
00086     return 0.0; // dummy  
00087 }
```

```
8.134.3.8 histogramNumClasses() unsigned short StatisticsDataFileDummyImpl::histogramNumClasses ( ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 81 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00081 {  
00082     return 0; // dummy  
00083 }
```

```
8.134.3.9 max() double StatisticsDataFileDummyImpl::max ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 30 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00030 {  
00031     return 0.0; // dummy  
00032 }
```

```
8.134.3.10 mediane() double StatisticsDataFileDummyImpl::mediane ( ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 42 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00042 {  
00043     return 0.0; // dummy  
00044 }
```

```
8.134.3.11 min() double StatisticsDataFileDummyImpl::min ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 26 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00026 {  
00027     return 0.0; // dummy  
00028 }
```

8.134.3.12 mode() double StatisticsDataFileDummyImpl::mode () [virtual]

Implements [StatisticsDatafile_if](#).

Definition at line 38 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00038 {  
00039     return 0.0; // dummy  
00040 }
```

8.134.3.13 newSampleSize() unsigned int StatisticsDataFileDummyImpl::newSampleSize (

```
    double confidencelevel,  
    double halfWidth ) [virtual]
```

Definition at line 62 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00062 {  
00063     return 0; // dummy  
00064 }
```

8.134.3.14 numElements() unsigned int StatisticsDataFileDummyImpl::numElements () [virtual]

Implements [Statistics_if](#).

Definition at line 22 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00022 {  
00023     return 0; // dummy  
00024 }
```

8.134.3.15 quartil() double StatisticsDataFileDummyImpl::quartil (

```
    unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 66 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00066 {  
00067     return 0.0; // dummy  
00068 }
```

8.134.3.16 setCollector() void StatisticsDataFileDummyImpl::setCollector (

```
    Collector_if * collector ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 97 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00097 {  
00098  
00099 }
```

```
8.134.3.17 setHistogramNumClasses() void StatisticsDataFileDummyImpl::setHistogramNumClasses  
(  
    unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 78 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00078  
00079 }
```

```
8.134.3.18 stddeviation() double StatisticsDataFileDummyImpl::stddeviation () [virtual]
```

Implements [Statistics_if](#).

Definition at line 50 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00050 {  
00051     return 0.0; // dummy  
00052 }
```

```
8.134.3.19 variance() double StatisticsDataFileDummyImpl::variance () [virtual]
```

Implements [Statistics_if](#).

Definition at line 46 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00046 {  
00047     return 0.0; // dummy  
00048 }
```

```
8.134.3.20 variationCoef() double StatisticsDataFileDummyImpl::variationCoef () [virtual]
```

Implements [Statistics_if](#).

Definition at line 54 of file [StatisticsDataFileDefaultImpl.cpp](#).

```
00054 {  
00055     return 0.0; // dummy  
00056 }
```

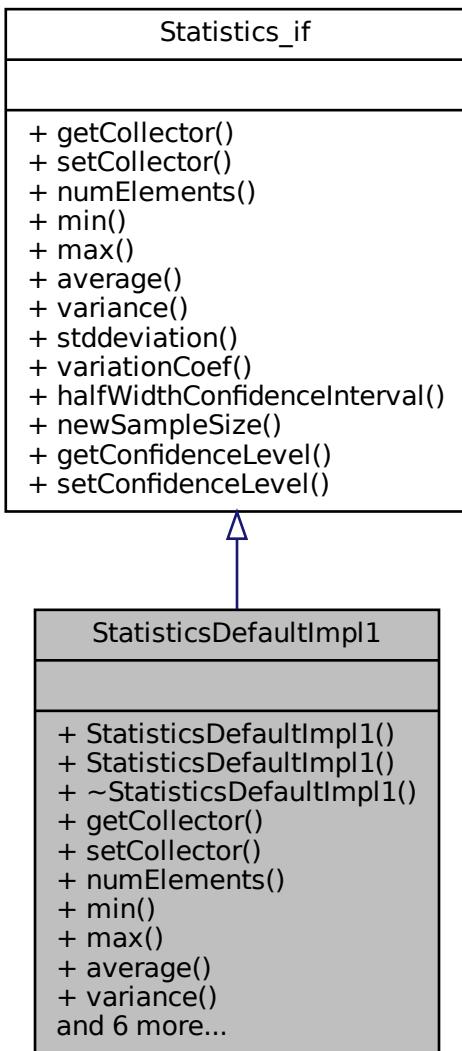
The documentation for this class was generated from the following files:

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

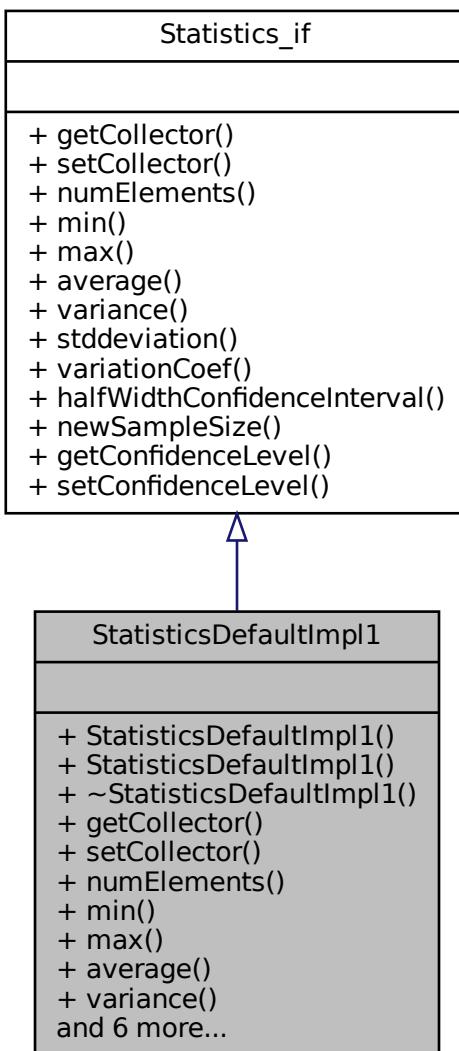
8.135 StatisticsDefaultImpl1 Class Reference

```
#include <StatisticsDefaultImpl1.h>
```

Inheritance diagram for StatisticsDefaultImpl1:



Collaboration diagram for StatisticsDefaultImpl1:



Public Member Functions

- `StatisticsDefaultImpl1 ()`
When constructor is invoked without a Collector, it is taken from `Traits<Statistics_if>::CollectorImplementation` configuration.
- `StatisticsDefaultImpl1 (Collector_if *collector)`
- `virtual ~StatisticsDefaultImpl1 ()=default`
- `virtual Collector_if * getCollector ()`
- `virtual 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 ()`
- virtual unsigned int `newSampleSize (double halfWidth)`
- virtual double `getConfidenceLevel ()`
- virtual void `setConfidenceLevel (double confidencelevel)`

8.135.1 Detailed Description

Definition at line 21 of file [StatisticsDefaultImpl1.h](#).

8.135.2 Constructor & Destructor Documentation

8.135.2.1 StatisticsDefaultImpl1() [1/2] [StatisticsDefaultImpl1::StatisticsDefaultImpl1 \(\)](#)

When constructor is invoked without a Collector, it is taken from [Traits<Statistics_if>::CollectorImplementation](#) configuration.

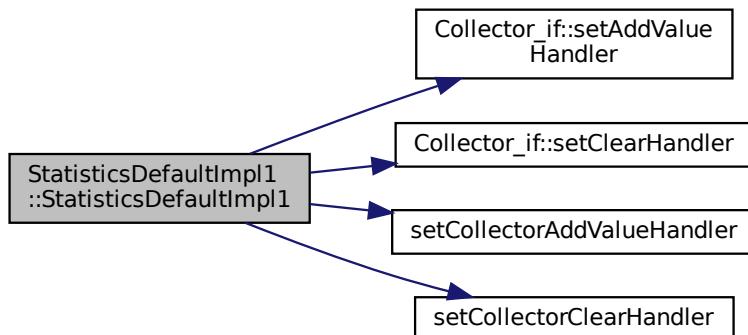
Definition at line 23 of file [StatisticsDefaultImpl1.cpp](#).

```

00023                                     {
00024     //_collector = new Traits<Statistics_if>::CollectorImplementation();
00025     _collector = new Traits<ModelComponent>::StatisticsCollector_CollectorImplementation();
00026
00027     _collector->setAddValueHandler(setCollectorAddValueHandler(&StatisticsDefaultImpl1::collectorAddHandler,
00028                                     this));
00029     _collector->setClearHandler(setCollectorClearHandler(&StatisticsDefaultImpl1::collectorClearHandler,
00030                                     this));
00031     //_collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00032                                     std::placeholders::_1));
00033     this->initStatistics();
00034 }
```

References [Collector_if::setAddValueHandler\(\)](#), [Collector_if::setClearHandler\(\)](#), [setCollectorAddValueHandler\(\)](#), and [setCollectorClearHandler\(\)](#).

Here is the call graph for this function:



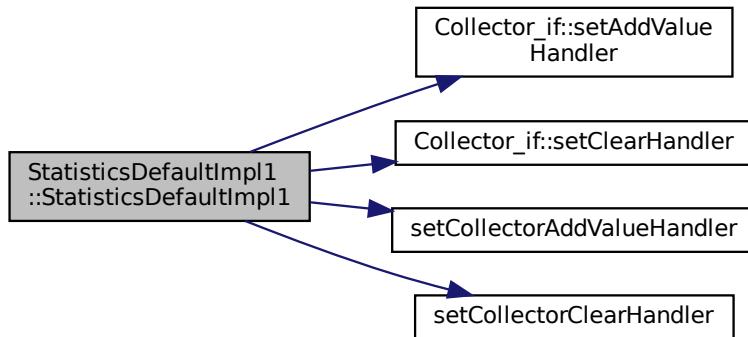
8.135.2.2 StatisticsDefaultImpl1() [2/2] StatisticsDefaultImpl1::StatisticsDefaultImpl1 (
 Collector_if * collector)

Definition at line 32 of file [StatisticsDefaultImpl1.cpp](#).

```
00032     _collector = collector;
00033
00034     _collector->setAddValueHandler(setCollectorAddValueHandler(&StatisticsDefaultImpl1::collectorAddHandler,
00035                                         this));
00036     _collector->setClearHandler(setCollectorClearHandler(&StatisticsDefaultImpl1::collectorClearHandler,
00037                                         this));
00038 // _collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00039 //                                         std::placeholders::_1));
00040     this->initStatistics();
00041
00042 }
```

References [Collector_if::setAddValueHandler\(\)](#), [Collector_if::setClearHandler\(\)](#), [setCollectorAddValueHandler\(\)](#), and [setCollectorClearHandler\(\)](#).

Here is the call graph for this function:



8.135.2.3 ~StatisticsDefaultImpl1() virtual StatisticsDefaultImpl1::~StatisticsDefaultImpl1 ()
[virtual], [default]

8.135.3 Member Function Documentation

8.135.3.1 average() double StatisticsDefaultImpl1::average () [virtual]

Implements [Statistics_if](#).

Definition at line 96 of file [StatisticsDefaultImpl1.cpp](#).

```
00096     {
00097     return _average;
00098 }
```

8.135.3.2 `getCollector()` `Collector_if * StatisticsDefaultImpl1::getCollector () [virtual]`

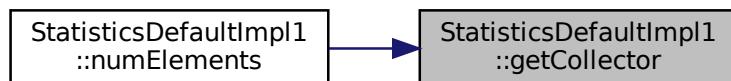
Implements [Statistics_if](#).

Definition at line 131 of file [StatisticsDefaultImpl1.cpp](#).

```
00131
00132     return this->_collector;
00133 }
```

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

Here is the caller graph for this function:



8.135.3.3 `getConfidenceLevel()` `double StatisticsDefaultImpl1::getConfidenceLevel () [virtual]`

Implements [Statistics_if](#).

Definition at line 123 of file [StatisticsDefaultImpl1.cpp](#).

```
00123
00124     return _confidenceLevel;
00125 }
```

8.135.3.4 `halfWidthConfidenceInterval()` `double StatisticsDefaultImpl1::halfWidthConfidenceInterval () [virtual]`

Implements [Statistics_if](#).

Definition at line 112 of file [StatisticsDefaultImpl1.cpp](#).

```
00112
00113     return _halfWidth;
00114 }
```

8.135.3.5 `max()` `double StatisticsDefaultImpl1::max () [virtual]`

Implements [Statistics_if](#).

Definition at line 89 of file [StatisticsDefaultImpl1.cpp](#).

```
00089
00090     if (_elems > 0)
00091         return _max;
00092     else
00093         return 0.0;
00094 }
```

8.135.3.6 min() double StatisticsDefaultImpl1::min () [virtual]

Implements [Statistics_if](#).

Definition at line 82 of file [StatisticsDefaultImpl1.cpp](#).

```
00082     if (_elems > 0) {  
00083         return _min;  
00084     else  
00085         return 0.0;  
00087 }
```

8.135.3.7 newSampleSize() unsigned int StatisticsDefaultImpl1::newSampleSize (double halfWidth) [virtual]

Implements [Statistics_if](#).

Definition at line 127 of file [StatisticsDefaultImpl1.cpp](#).

```
00127     return 0;  
00128 }  
00129 }
```

8.135.3.8 numElements() unsigned int StatisticsDefaultImpl1::numElements () [virtual]

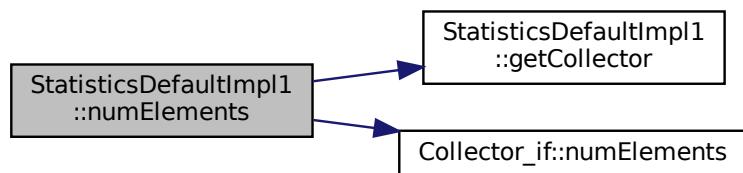
Implements [Statistics_if](#).

Definition at line 78 of file [StatisticsDefaultImpl1.cpp](#).

```
00078     {  
00079         return this->getCollector ()->numElements ();  
00080     }
```

References [getCollector\(\)](#), and [Collector_if::numElements\(\)](#).

Here is the call graph for this function:



8.135.3.9 setCollector() void StatisticsDefaultImpl1::setCollector (Collector_if * collector) [virtual]

Implements [Statistics_if](#).

Definition at line 135 of file [StatisticsDefaultImpl1.cpp](#).

```
00135     this->_collector = collector;
00136 }
00137 }
```

8.135.3.10 setConfidenceLevel() void StatisticsDefaultImpl1::setConfidenceLevel (double confidencelevel) [virtual]

Implements [Statistics_if](#).

Definition at line 116 of file [StatisticsDefaultImpl1.cpp](#).

```
00116 {
00117     _confidenceLevel = confidencelevel;
00118     //Integrator_if* integrator = new Traits<Integrator_if>::Implementation();
00119     _criticalTn_1 = 1.96; //integrator->integrate()
00120
00121 }
```

8.135.3.11 stddeviation() double StatisticsDefaultImpl1::stddeviation () [virtual]

Implements [Statistics_if](#).

Definition at line 104 of file [StatisticsDefaultImpl1.cpp](#).

```
00104 {
00105     return _stddeviation;
00106 }
```

8.135.3.12 variance() double StatisticsDefaultImpl1::variance () [virtual]

Implements [Statistics_if](#).

Definition at line 100 of file [StatisticsDefaultImpl1.cpp](#).

```
00100 {
00101     return _variance;
00102 }
```

8.135.3.13 variationCoef() double StatisticsDefaultImpl1::variationCoef () [virtual]

Implements [Statistics_if](#).

Definition at line 108 of file [StatisticsDefaultImpl1.cpp](#).

```
00108 {
00109     return _variationCoef;
00110 }
```

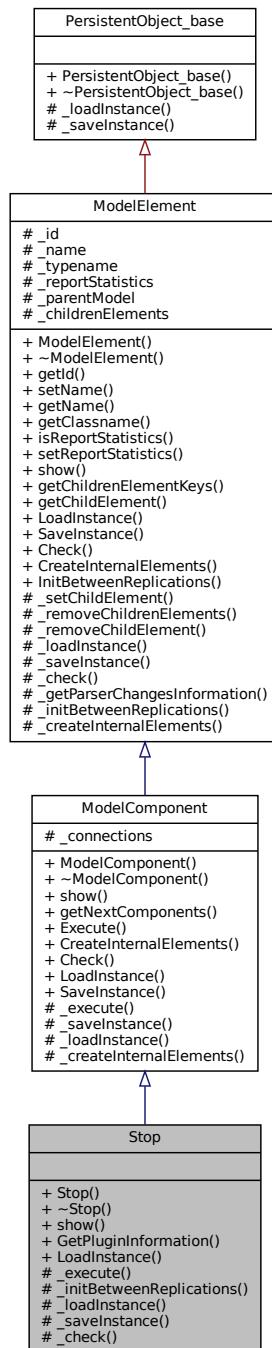
The documentation for this class was generated from the following files:

- [StatisticsDefaultImpl1.h](#)
- [StatisticsDefaultImpl1.cpp](#)

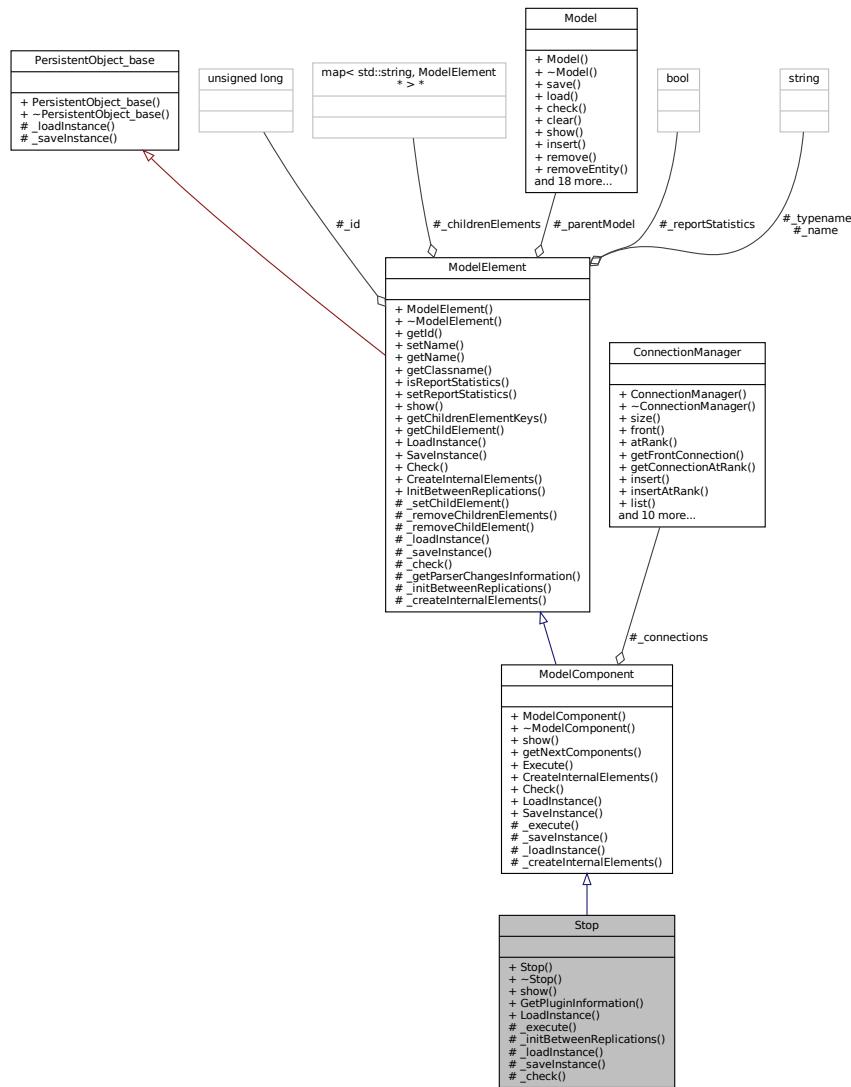
8.136 Stop Class Reference

```
#include <Stop.h>
```

Inheritance diagram for Stop:



Collaboration diagram for Stop:



Public Member Functions

- `Stop (Model *model, std::string name="")`
- virtual `~Stop ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.136.1 Detailed Description

Stop module DESCRIPTION The **Stop** module sets the operational status of a conveyor to inactive. The conveyor may have been activated from either the **Start** module or by initially being set to active at the start of the simulation. When the entity enters the **Stop** module, the conveyor will stop immediately, regardless of the type of conveyor or the number of entities currently on the conveyor. TYPICAL USES **Stop** a baggage conveyor after a pre-determined amount of time. **Stop** a conveyor for scheduled maintenance PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor to stop.

Definition at line 36 of file [Stop.h](#).

8.136.2 Constructor & Destructor Documentation

```
8.136.2.1 Stop() Stop::Stop (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file [Stop.cpp](#).

```
00018 : ModelComponent(model, Util::TypeOf<Stop>(), name) {
00019 }
```

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

Here is the caller graph for this function:



```
8.136.2.2 ~Stop() virtual Stop::~Stop ( ) [virtual], [default]
```

8.136.3 Member Function Documentation

8.136.3.1 `_check()` `bool Stop::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Stop.cpp](#).

```
00057     {  
00058     bool resultAll = true;  
00059     //...  
00060     return resultAll;  
00061 }
```

8.136.3.2 `_execute()` `void Stop::_execute (Entity * entity) [protected], [virtual]`

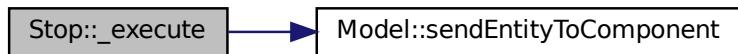
Implements [ModelComponent](#).

Definition at line 35 of file [Stop.cpp](#).

```
00035     {  
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);  
00038 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.136.3.3 `_initBetweenReplications()` `void Stop::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Stop.cpp](#).

```
00048     {  
00049 }
```

```
8.136.3.4 _loadInstance() bool Stop::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

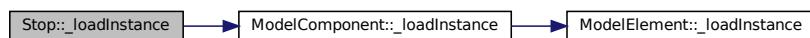
Definition at line 40 of file [Stop.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

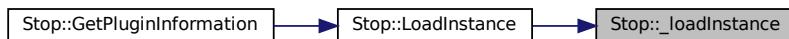
References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.136.3.5 _saveInstance() std::map< std::string, std::string > * Stop::_saveInstance () [protected], [virtual]
```

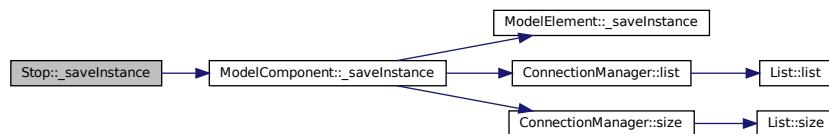
Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Stop.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



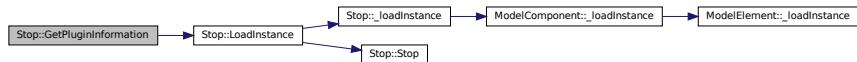
8.136.3.6 GetPluginInformation() `PluginInformation * Stop::GetPluginInformation () [static]`

Definition at line 63 of file `Stop.cpp`.

```
00063
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Stop>(), &Stop::LoadInstance);
00065     // ...
00066     return info;
00067 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.136.3.7 LoadInstance() `ModelComponent * Stop::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

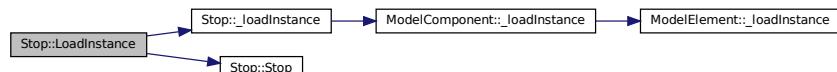
Definition at line 25 of file `Stop.cpp`.

```
00025
00026     Stop* newComponent = new Stop(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
```

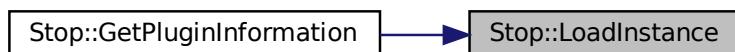
References `_loadInstance()`, and `Stop()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.136.3.8 show() std::string Stop::show() [virtual]

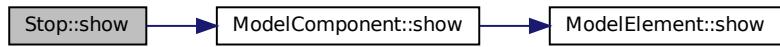
Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Stop.cpp](#).

```
00021     {
00022     return ModelComponent::show() + "";
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



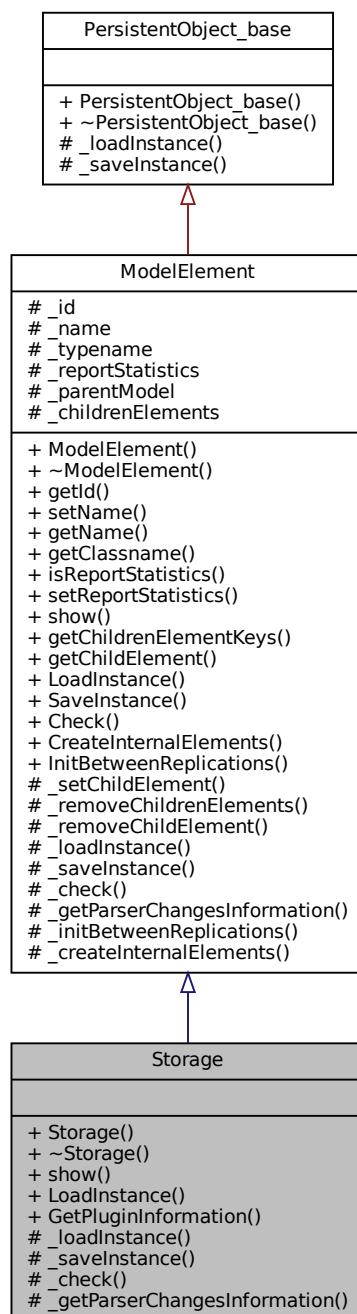
The documentation for this class was generated from the following files:

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

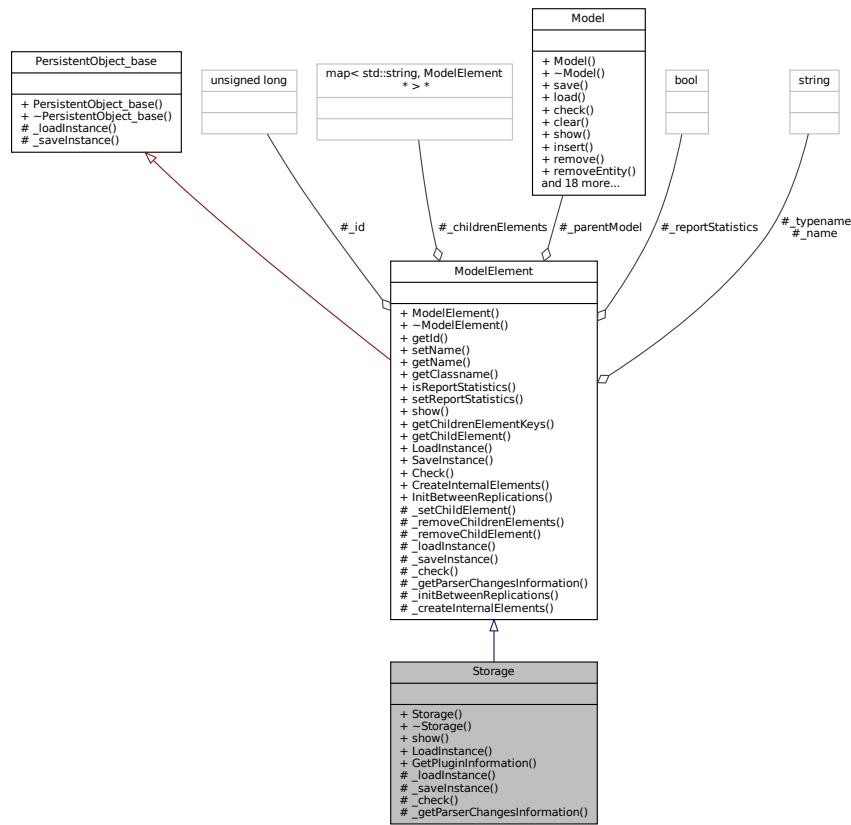
8.137 Storage Class Reference

```
#include <Storage.h>
```

Inheritance diagram for Storage:



Collaboration diagram for Storage:



Public Member Functions

- [Storage \(Model *model, std::string name=""\)](#)
- virtual [~Storage \(\)=default](#)
- virtual std::string [show \(\)](#)

Static Public Member Functions

- static ModelElement * [LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)
- static PluginInformation * [GetPluginInformation \(\)](#)

Protected Member Functions

- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)
- virtual ParserChangesInformation * [_getParserChangesInformation \(\)](#)

Additional Inherited Members

8.137.1 Detailed Description

Definition at line 37 of file [Storage.h](#).

8.137.2 Constructor & Destructor Documentation

8.137.2.1 Storage() `Storage::Storage (`
 `Model * model,`
 `std::string name = "")`

Definition at line 16 of file [Storage.cpp](#).

```
00016 : ModelElement (model, Util::TypeOf<Storage>(), name) {  
00017 }
```

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

Here is the caller graph for this function:



8.137.2.2 ~Storage() `virtual Storage::~Storage () [virtual], [default]`

8.137.3 Member Function Documentation

8.137.3.1 _check() `bool Storage::_check (`
 `std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Storage.cpp](#).

```
00058 {  
00059     bool resultAll = true;  
00060     // resultAll |= ...  
00061     return resultAll;  
00062 }
```

8.137.3.2 _getParserChangesInformation() `ParserChangesInformation * Storage::_getParserChangesInformation () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 64 of file [Storage.cpp](#).

```
00064 {  
00065     ParserChangesInformation* changes = new ParserChangesInformation();  
00066     //changes->getProductionToAdd()->insert(...);  
00067     //changes->getTokensToAdd()->insert(...);  
00068     return changes;  
00069 }
```

```
8.137.3.3 _loadInstance() bool Storage::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 39 of file [Storage.cpp](#).

```
00039
00040     bool res = ModelElement::_loadInstance(fields);
00041     if (res) {
00042         try {
00043             //this->_attributeName = (*fields->find("attributeName")).second;
00044             //this->_orderRule = static_cast<OrderRule>
00045             (std::stoi((*fields->find("orderRule")).second));
00046         } catch (...) {
00047         }
00048     return res;
00049 }
```

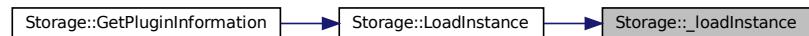
References [ModelElement::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.137.3.4 _saveInstance() std::map< std::string, std::string > * Storage::_saveInstance ( )
[protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 51 of file [Storage.cpp](#).

```
00051
00052     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00053     //Util::TypeOf<Storage>());
00054     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055     //fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00056     return fields;
00057 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.137.3.5 GetPluginInformation() `PluginInformation * Storage::GetPluginInformation () [static]`

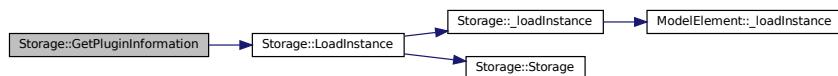
Definition at line 24 of file `Storage.cpp`.

```

00024
00025     PluginInformation* info = new PluginInformation(Util::TypeOf<Storage>(), &Storage::LoadInstance);
00026     return info;
00027 }
  
```

References `LoadInstance()`.

Here is the call graph for this function:



8.137.3.6 LoadInstance() `ModelElement * Storage::LoadInstance (`

```

        Model * model,
        std::map< std::string, std::string > * fields ) [static]
      
```

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

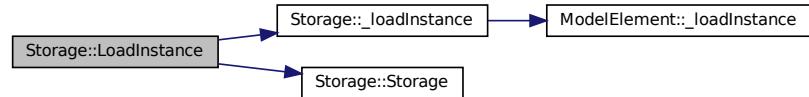
```

00029
00030     Storage* newElement = new Storage(model);
00031     try {
00032         newElement->_loadInstance(fields);
00033     } catch (const std::exception& e) {
00034
00035     }
00036     return newElement;
00037 }
  
```

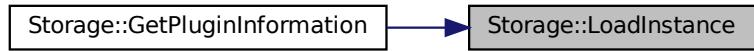
References `_loadInstance()`, and `Storage()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.137.3.7 `show()` std::string Storage::show () [virtual]

Reimplemented from [ModelElement](#).

Definition at line 19 of file [Storage.cpp](#).

```
00019         {  
00020     return ModelElement::show() +  
00021         "";  
00022 }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:



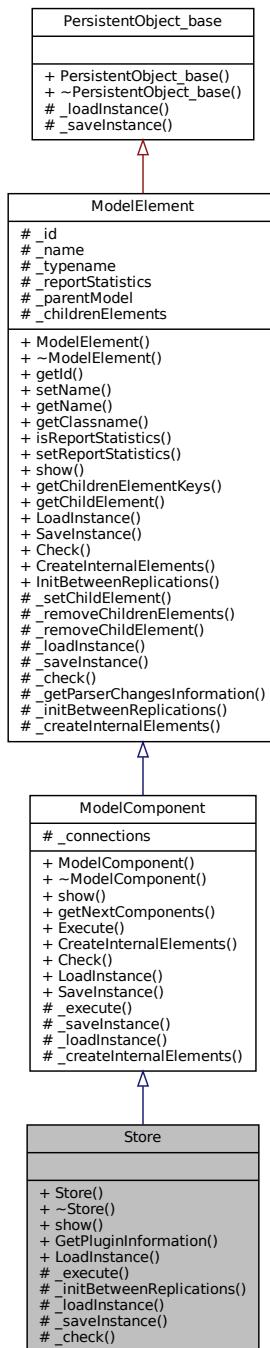
The documentation for this class was generated from the following files:

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

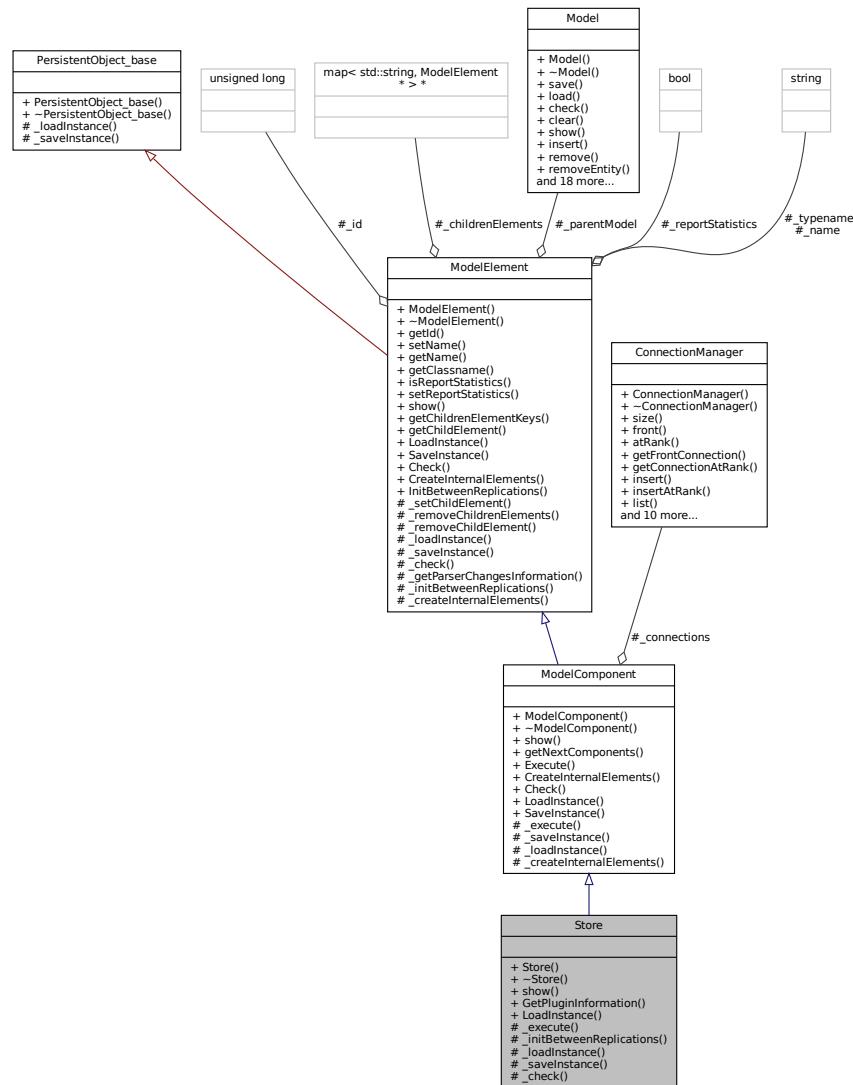
8.138 Store Class Reference

```
#include <Store.h>
```

Inheritance diagram for Store:



Collaboration diagram for Store:



Public Member Functions

- **Store (Model *model, std::string name="")**
- virtual **~Store ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.138.1 Detailed Description

Store module DESCRIPTION The **Store** module adds an entity to storage. The **Unstore** module may then be used to remove the entity from the storage. When an entity arrives at the **Store** module, the storage specified is incremented, and the entity immediately moves to the next module in the model. Storages are useful for displaying entity animation while an entity undergoes processing in other modules. Additionally, statistics may be kept on the number of entities in storage. TYPICAL USES Animating a part through a number of delay operations (load, setup, process, unload) Tracking the number of customers within a grocery store (place in storage upon entry) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of specifying the storage name as a **Storage**, **Set**, **Attribute**, or Expression. **Storage** Name Name of the storage to which the entity will be added. Applies only when the Type is **Storage**. **Set** Name Name of the storage set from which the storage is to be selected. Applies only when the Type is **Set**. **Set** Index Index into the defined storage set that contains the desired storage name. Applies only when the Type is **Set**. **Attribute** Name of the attribute whose value contains the storage. Applies only when the Type is **Attribute**. Expression Expression that is evaluated to the storage into which the entity is placed. Applies only when the Type is **Expression**.

Definition at line 50 of file [Store.h](#).

8.138.2 Constructor & Destructor Documentation

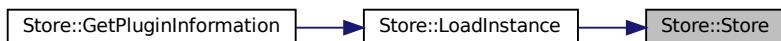
8.138.2.1 Store() `Store::Store (`
 `Model * model,`
 `std::string name = "")`

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

```
00017 : ModelComponent(model, Util::TypeOf<Store>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



8.138.2.2 ~Store() `virtual Store::~Store () [virtual], [default]`

8.138.3 Member Function Documentation

```
8.138.3.1 _check() bool Store::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Store.cpp](#).

```
00056                                     {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
```

```
8.138.3.2 _execute() void Store::_execute (
    Entity * entity ) [protected], [virtual]
```

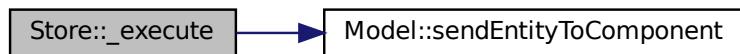
Implements [ModelComponent](#).

Definition at line 34 of file [Store.cpp](#).

```
00034                                     {
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



```
8.138.3.3 _initBetweenReplications() void Store::_initBetweenReplications ( ) [protected],
[virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Store.cpp](#).

```
00047                                     {
00048 }
```

```
8.138.3.4 _loadInstance() bool Store::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

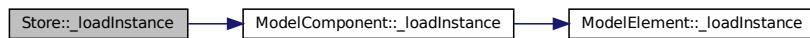
Definition at line 39 of file [Store.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.138.3.5 _saveInstance() std::map< std::string, std::string > * Store::_saveInstance ( )
[protected], [virtual]
```

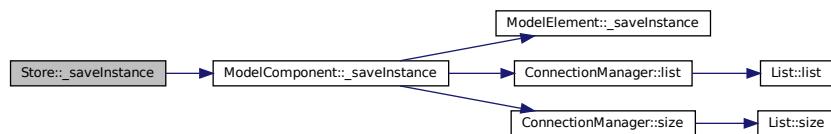
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [Store.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.138.3.6 GetPluginInformation() `PluginInformation * Store::GetPluginInformation () [static]`

Definition at line 62 of file `Store.cpp`.

```
00062
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Store>(), &Store::LoadInstance);
00064 // ...
00065     return info;
00066 }
```

References `LoadInstance()`.

Here is the call graph for this function:



8.138.3.7 LoadInstance() `ModelComponent * Store::LoadInstance (`

```
        Model * model,
        std::map< std::string, std::string > * fields ) [static]
```

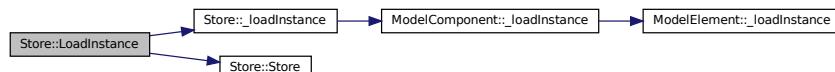
Definition at line 24 of file `Store.cpp`.

```
00024
00025     Store* newComponent = new Store(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
```

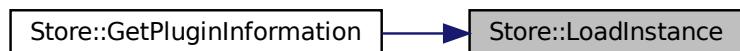
References `_loadInstance()`, and `Store()`.

Referenced by `GetPluginInformation()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.138.3.8 show() std::string Store::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Store.cpp](#).

```
00020         {  
00021     return ModelComponent::show() + "";  
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



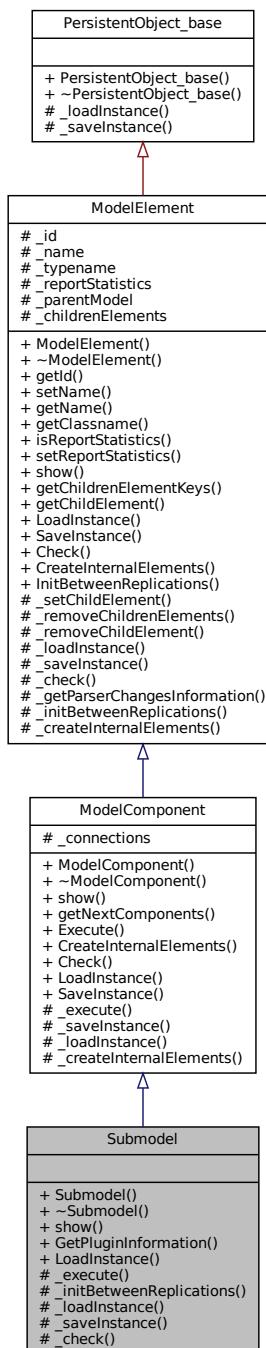
The documentation for this class was generated from the following files:

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

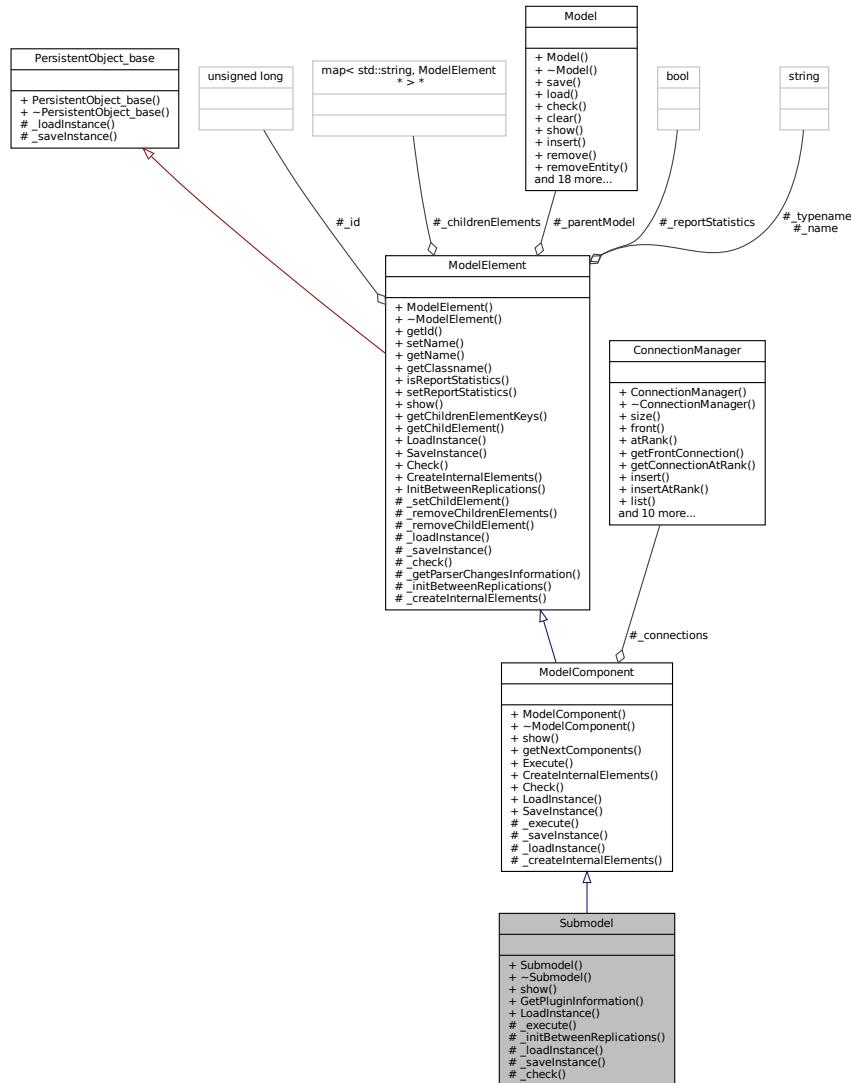
8.139 Submodel Class Reference

```
#include <Submodel.h>
```

Inheritance diagram for Submodel:



Collaboration diagram for Submodel:



Public Member Functions

- **Submodel (Model *model, std::string name="")**
- virtual **~Submodel ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual void **_execute (Entity *entity)**
- virtual void **_initBetweenReplications ()**
- virtual bool **_loadInstance (std::map< std::string, std::string > *fields)**
- virtual std::map< std::string, std::string > * **_saveInstance ()**
- virtual bool **_check (std::string *errorMessage)**

Additional Inherited Members

8.139.1 Detailed Description

This component ...

Definition at line 22 of file [Submodel.h](#).

8.139.2 Constructor & Destructor Documentation

```
8.139.2.1 Submodel() Submodel::Submodel (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file [Submodel.cpp](#).

```
00018 {
00019 }
```

```
: ModelComponent(model, Util::TypeOf<Submodel>(), name)
```

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

Here is the caller graph for this function:



```
8.139.2.2 ~Submodel() virtual Submodel::~Submodel () [virtual], [default]
```

8.139.3 Member Function Documentation

```
8.139.3.1 _check() bool Submodel::_check (
    std::string * errorMessage) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Submodel.cpp](#).

```
00057 {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
```

8.139.3.2 `_execute()` void Submodel::`_execute` (
 Entity * entity) [protected], [virtual]

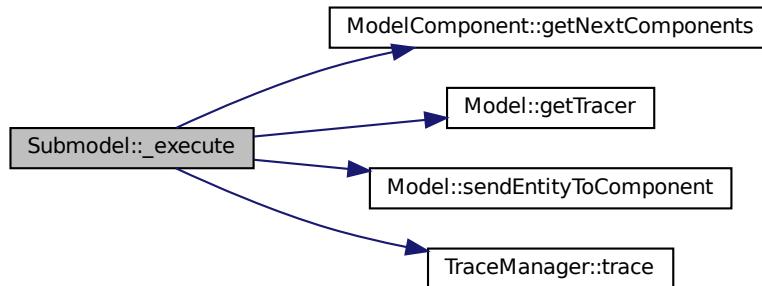
Implements [ModelComponent](#).

Definition at line 35 of file [Submodel.cpp](#).

```
00035      {
00036      _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037      this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00038      0.0);
00038 }
```

References [ModelElement::_parentModel](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [Model::sendEntityToComponent](#) and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.139.3.3 `_initBetweenReplications()` void Submodel::`_initBetweenReplications` () [protected], [virtual]

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Submodel.cpp](#).

```
00048 {
00049 }
```

8.139.3.4 `_loadInstance()` bool Submodel::`_loadInstance` (
 std::map< std::string, std::string > * fields) [protected], [virtual]

Reimplemented from [ModelComponent](#).

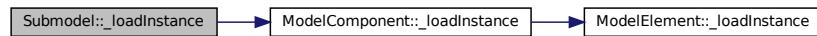
Definition at line 40 of file [Submodel.cpp](#).

```
00040
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.139.3.5 [_saveInstance\(\)](#)

```
std::map< std::string, std::string > * Submodel::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

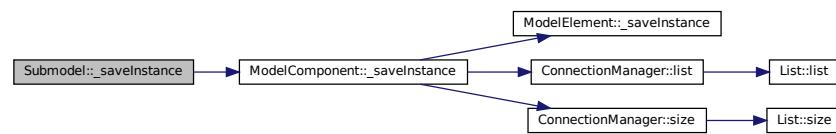
Definition at line 51 of file [Submodel.cpp](#).

```

00051
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.139.3.6 GetPluginInformation() `PluginInformation * Submodel::GetPluginInformation () [static]`

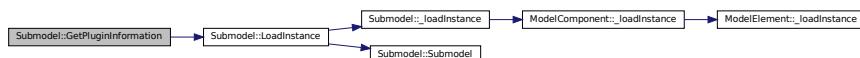
Definition at line 63 of file `Submodel.cpp`.

```
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Submodel>(),
00064         &Submodel::LoadInstance);
00065     // ...
00066     return info;
00067 }
```

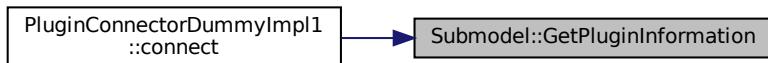
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.139.3.7 LoadInstance() `ModelComponent * Submodel::LoadInstance (`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

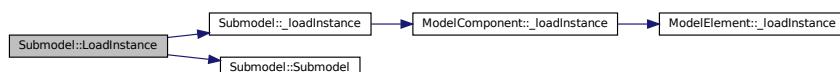
Definition at line 25 of file `Submodel.cpp`.

```
00025
00026     Submodel* newComponent = new Submodel(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030
00031     }
00032     return newComponent;
00033 }
```

References [_loadInstance\(\)](#), and [Submodel\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.139.3.8 show() std::string Submodel::show () [virtual]

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Submodel.cpp](#).

```
00021           {  
00022     return ModelComponent::show() + "";  
00023 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



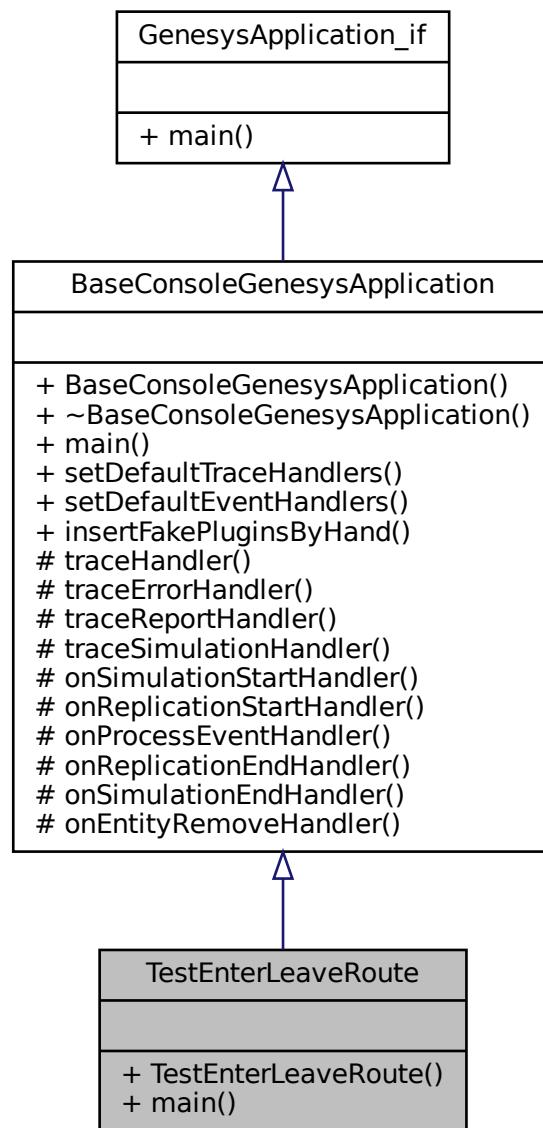
The documentation for this class was generated from the following files:

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

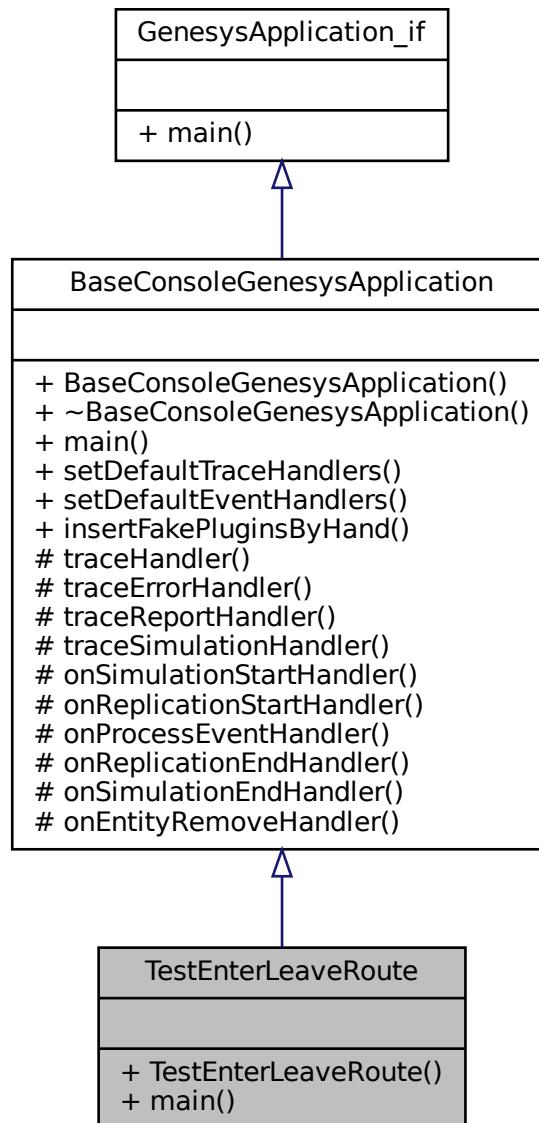
8.140 TestEnterLeaveRoute Class Reference

```
#include <TestEnterLeaveRoute.h>
```

Inheritance diagram for TestEnterLeaveRoute:



Collaboration diagram for TestEnterLeaveRoute:



Public Member Functions

- `TestEnterLeaveRoute ()`
- int `main` (int argc, char **argv)

Additional Inherited Members

8.140.1 Detailed Description

Definition at line 19 of file [TestEnterLeaveRoute.h](#).

8.140.2 Constructor & Destructor Documentation

8.140.2.1 TestEnterLeaveRoute() TestEnterLeaveRoute::TestEnterLeaveRoute ()

Definition at line 35 of file [TestEnterLeaveRoute.cpp](#).

```
00035     {
00036 }
```

8.140.3 Member Function Documentation

8.140.3.1 main() int TestEnterLeaveRoute::main (

```
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 38 of file [TestEnterLeaveRoute.cpp](#).

```
00038     {
00039     Simulator* genesys = new Simulator();
00040     // creates an empty model
00041     Model* model = new Model(genesys);
00042     // Handle traces and simulation events to output them
00043     TraceManager* tm = model->getTracer();
00044     this->setDefaultTraceHandlers(tm);
00045     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of
tracing
00046     tm->setTraceLevel(Util::TraceLevel::componentArrival);
00047     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00048     this->insertFakePluginsByHand(genesys);
00049     // get easy access to classes used to insert components and elements into a model
00050     ComponentManager* components = model->getComponents();
00051     ElementManager* elements = model->getElements();
00052     //
00053     // build the simulation model
00054     //
00055     // set general info about the model
00056     ModelSimulation* sim = model->getSimulation();
00057     sim->setReplicationLength(30);
00058     sim->setNumberOfReplications(3);
00059     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00060     EntityType* entityType1 = new EntityType(model, "AnyEntityType");
00061     elements->insert(entityType1);
00062     // create a ModelComponent of type Create, used to insert entities into the model
00063     Create* create1 = new Create(model);
00064     create1->setEntityType(entityType1);
00065     create1->setTimeBetweenCreationsExpression("5.0");
00066     create1->setEntitiesPerCreation(1);
00067     components->insert(create1);
00068     // create stations to enter and route to
00069     Station* station1 = new Station(model, "Station 1");
00070     Station* station2 = new Station(model, "Station 2");
00071     Station* station3 = new Station(model, "Station 3");
00072     elements->insert(station1);
00073     elements->insert(station2);
00074     elements->insert(station3);
00075     // create components to Enter into Stations
00076     Enter* enter1 = new Enter(model);
00077     enter1->setStation(station1);
00078     components->insert(enter1);
00079     Enter* enter2 = new Enter(model);
00080     enter2->setStation(station2);
00081     components->insert(enter2);
00082     Enter* enter3 = new Enter(model);
00083     enter3->setStation(station3);
00084     components->insert(enter3);
00085     // create components to Leave stations
00086     Leave* leave1 = new Leave(model);
```

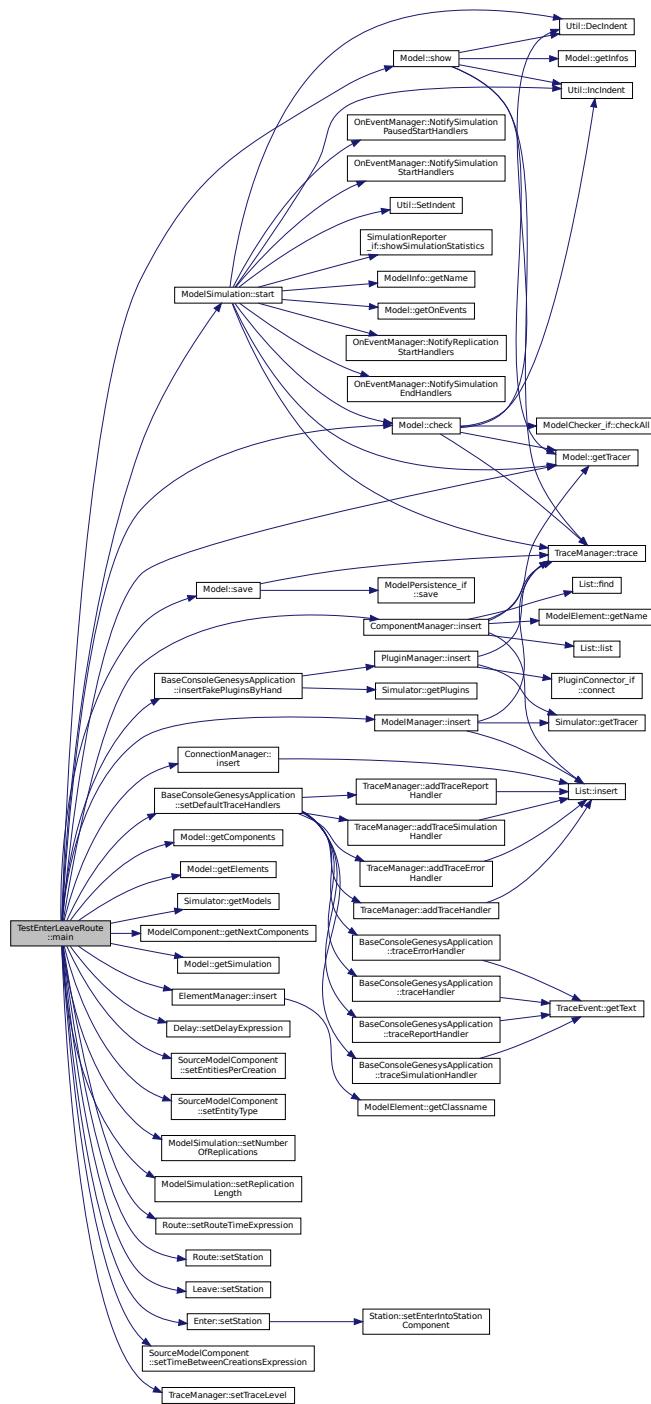
```

00087     leave1->setStation(station1);
00088     components->insert(leave1);
00089     Leave* leave2 = new Leave(model);
00090     leave2->setStation(station2);
00091     components->insert(leave2);
00092     Leave* leave3 = new Leave(model);
00093     leave3->setStation(station3);
00094     components->insert(leave3);
00095     // create route components
00096     Route* route0 = new Route(model);
00097     route0->setStation(station1);
00098     route0->setRouteTimeExpression("0.5");
00099     components->insert(route0);
00100     Route* routel = new Route(model);
00101     routel->setStation(station2);
00102     routel->setRouteTimeExpression("0.5");
00103     components->insert(routel);
00104     Route* route2 = new Route(model);
00105     route2->setStation(station3);
00106     route2->setRouteTimeExpression("0.5");
00107     components->insert(route2);
00108     // create delay components
00109     Delay* delay1 = new Delay(model);
00110     delay1->setDelayExpression("1.0");
00111     components->insert(delay1);
00112     Delay* delay2 = new Delay(model);
00113     delay2->setDelayExpression("1.0");
00114     components->insert(delay2);
00115     Delay* delay3 = new Delay(model);
00116     delay3->setDelayExpression("1.0");
00117     components->insert(delay3);
00118     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00119     Dispose* dispose1 = new Dispose(model);
00120     components->insert(dispose1);
00121     // connect model components to create a "workflow"
00122     createl->getNextComponents()->insert(route0);
00123     //
00124     enter1->getNextComponents()->insert(delay1);
00125     delay1->getNextComponents()->insert(leave1);
00126     leave1->getNextComponents()->insert(routel);
00127     //
00128     enter2->getNextComponents()->insert(delay2);
00129     delay2->getNextComponents()->insert(leave2);
00130     leave2->getNextComponents()->insert(route2);
00131     //
00132     enter3->getNextComponents()->insert(delay3);
00133     delay3->getNextComponents()->insert(leave3);
00134     leave3->getNextComponents()->insert(dispose1);
00135     // insert the model into the simulator
00136     genesys->getModels()->insert(model);
00137     // check the model
00138     model->check();
00139     // save the model into a text file
00140     model->save("./temp/testEnterLeaveRoute.txt");
00141     // show the model
00142     model->show();
00143     // execute the simulation
00144     sim->start();
00145     return 0;
00146 }

```

References Model::check(), Util::componentArrival, Model::getComponents(), Model::getElements(), Simulator::getModels(), ModelComponent::getNextComponents(), Model::getSimulation(), Model::getTracer(), ComponentManager::insert(), ModelManager::insert(), ElementManager::insert(), ConnectionManager::insert(), BaseConsoleGenesysApplication::insertFakePlugin, Model::save(), BaseConsoleGenesysApplication::setDefaultTraceHandlers(), Delay::setDelayExpression(), SourceModelComponent::setEntitiesPerCreation(), SourceModelComponent::setEntityType(), ModelSimulation::setNumberOfReplicas, ModelSimulation::setReplicationLength(), Route::setRouteTimeExpression(), Route::setStation(), Leave::setStation(), Enter::setStation(), SourceModelComponent::setTimeBetweenCreationsExpression(), TraceManager::setTraceLevel(), Model::show(), and ModelSimulation::start().

Here is the call graph for this function:



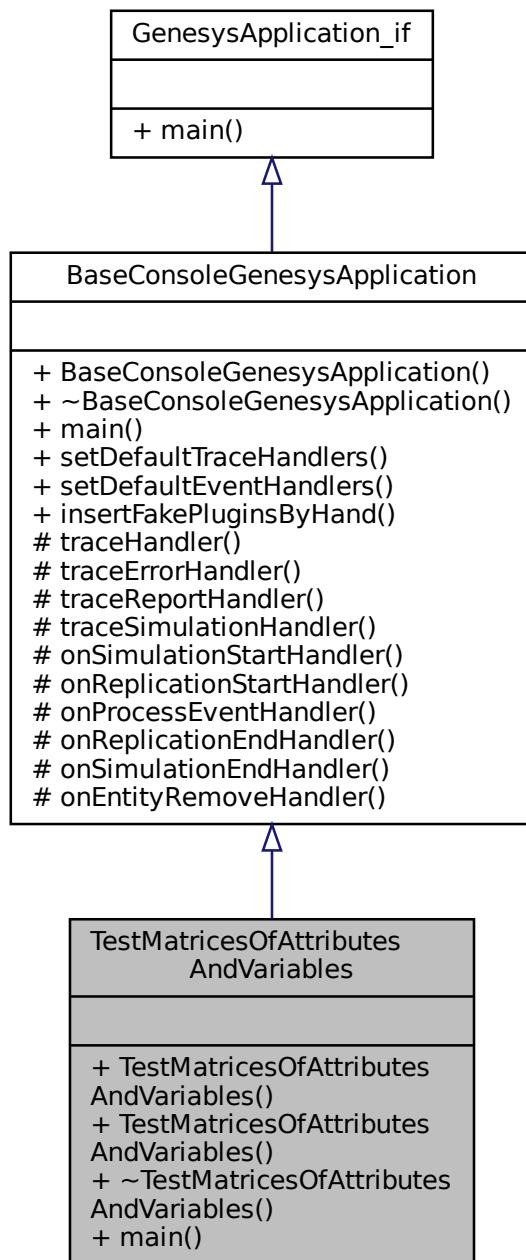
The documentation for this class was generated from the following files:

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

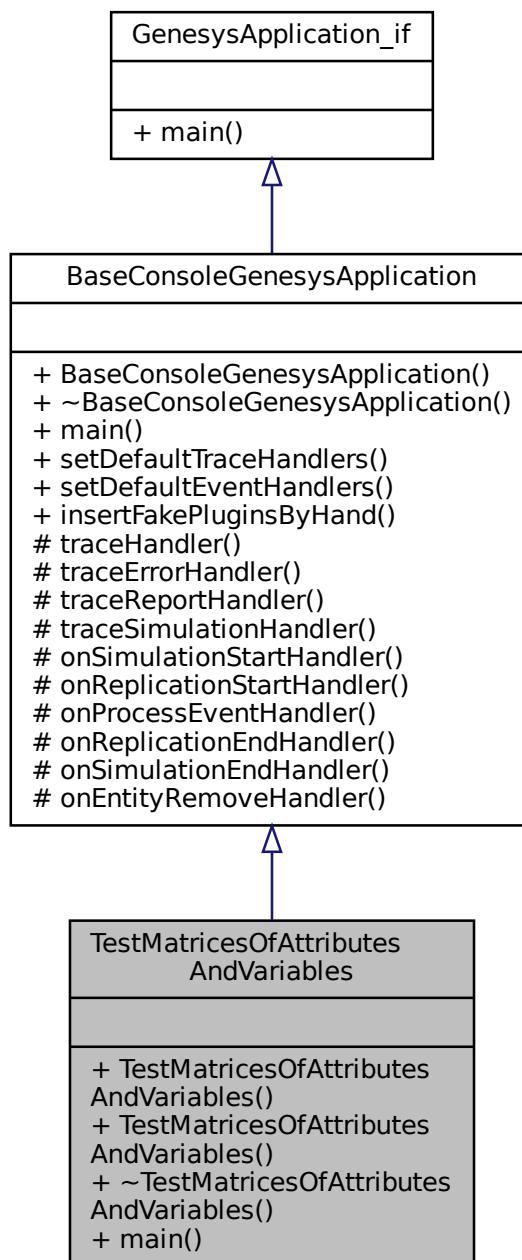
8.141 TestMatricesOfAttributesAndVariables Class Reference

```
#include <TestMatricesOfAttributesAndVariables.h>
```

Inheritance diagram for TestMatricesOfAttributesAndVariables:



Collaboration diagram for TestMatricesOfAttributesAndVariables:



Public Member Functions

- `TestMatricesOfAttributesAndVariables ()`
- `TestMatricesOfAttributesAndVariables (const TestMatricesOfAttributesAndVariables &orig)`
- `virtual ~TestMatricesOfAttributesAndVariables ()`
- `virtual int main (int argc, char **argv)`

Additional Inherited Members

8.141.1 Detailed Description

Definition at line 19 of file [TestMatricesOfAttributesAndVariables.h](#).

8.141.2 Constructor & Destructor Documentation

8.141.2.1 TestMatricesOfAttributesAndVariables() [1/2] `TestMatricesOfAttributes::TestMatricesOfAttributesAndVariables ()`

Definition at line 27 of file [TestMatricesOfAttributesAndVariables.cpp](#).

```
00027 {  
00028 }
```

8.141.2.2 TestMatricesOfAttributesAndVariables() [2/2] `TestMatricesOfAttributes::TestMatricesOfAttributesAndVariables (`
`const TestMatricesOfAttributesAndVariables & orig)`

Definition at line 30 of file [TestMatricesOfAttributesAndVariables.cpp](#).

```
00030 {  
00031 }
```

8.141.2.3 ~TestMatricesOfAttributesAndVariables() `TestMatricesOfAttributes::~TestMatricesOfAttributesAndVariables () [virtual]`

Definition at line 33 of file [TestMatricesOfAttributesAndVariables.cpp](#).

```
00033 {  
00034 }
```

8.141.3 Member Function Documentation

```
8.141.3.1 main() int TestMatricesOfAttributesAndVariables::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

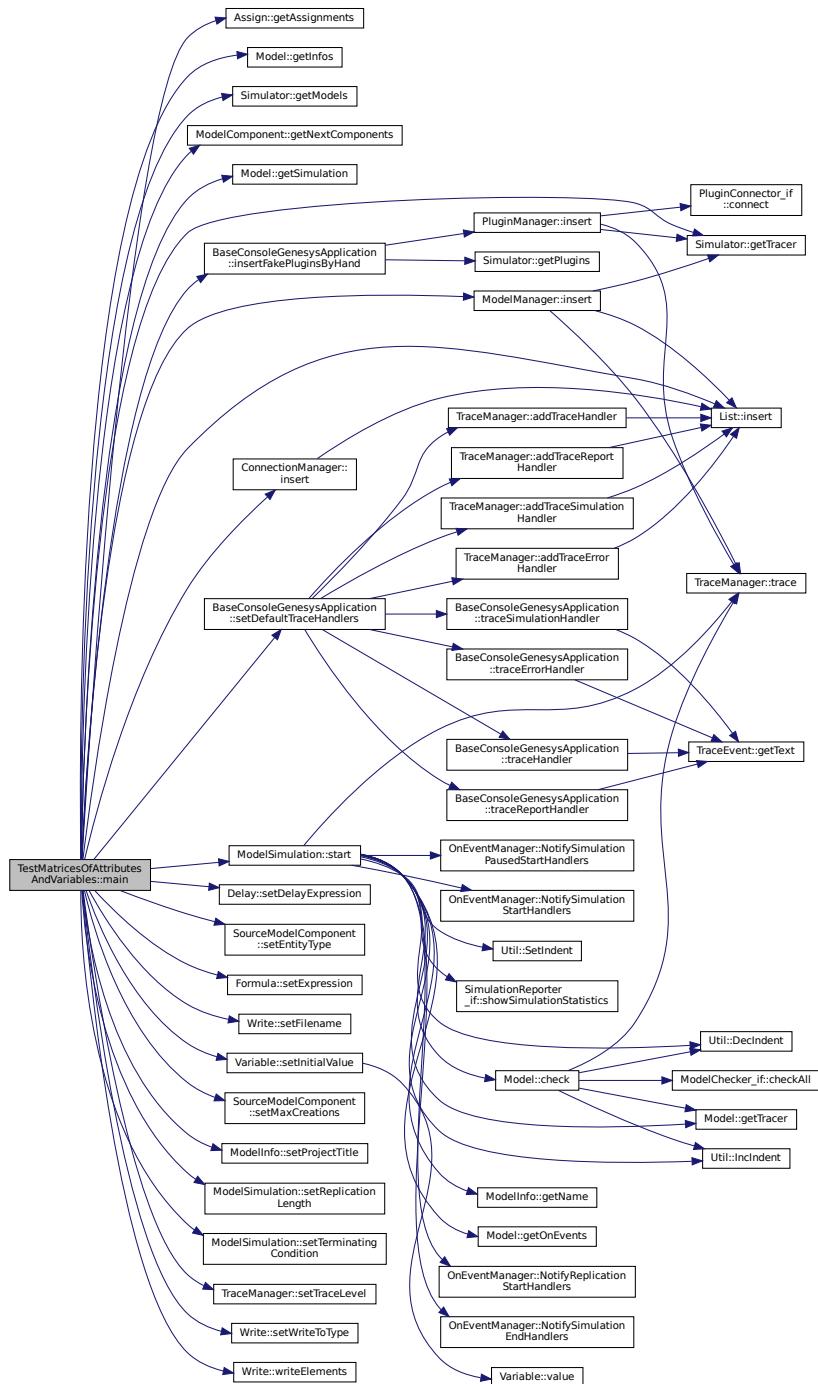
Definition at line 36 of file [TestMatricesOfAttributesAndVariables.cpp](#).

```
00036
00037     Simulator* genesys = new Simulator();
00038     setDefaultTraceHandlers(genesys->getTracer());
00039     genesys->getTracer()->setTraceLevel(Util::TraceLevel::modelSimulationInternal);
00040     insertFakePluginsByHand(genesys);
00041     Model* m = new Model(genesys);
00042     genesys->getModels()->insert(m);
00043     m->getInfos()->setProjectTitle("Stochastic Simulation of Chemical Reactions");
00044     m->getSimulation()->setReplicationLength(500);
00045     Create* crl = new Create(m);
00046     Write* wl = new Write(m);
00047     Assign* asl = new Assign(m, "Define próxima reação a ocorrer");
00048     Delay* del = new Delay(m, "Aguarda tempo em que a reação ocorre");
00049     Dispose* dil = new Dispose(m);
00050     crl->getNextComponents()->insert(wl);
00051     wl->getNextComponents()->insert(asl);
00052     asl->getNextComponents()->insert(del);
00053     del->getNextComponents()->insert(wl);
00054     del->getNextComponents()->insert(dil); // trick to be connected
00055     crl->setEntityType(new EntityType(m));
00056     crl->setMaxCreations("1");
00057     Variable* s = new Variable(m, "s");
00058     Variable* k = new Variable(m, "k");
00059     Variable* N = new Variable(m, "N");
00060     new Variable(m, "temp");
00061     Formula* prop = new Formula(m, "prop");
00062     s->setInitialValue("1", -1);
00063     s->setInitialValue("1,2", -1);
00064     s->setInitialValue("1,3", 1);
00065     s->setInitialValue("2,1", 1);
00066     s->setInitialValue("2,2", 1);
00067     s->setInitialValue("2,3", -1);
00068     k->setInitialValue("1", 0.1);
00069     k->setInitialValue("2", 0.2);
00070     N->setInitialValue("1", 100);
00071     N->setInitialValue("2", 100);
00072     N->setInitialValue("3", 0);
00073     prop->setExpression("1", "k[1]*N[1]*N[2]");
00074     prop->setExpression("2", "k[2]*N[3]");
00075     wl->setWriteToType(Write::WriteToType::FILE);
00076     wl->setFilename("./temp/molecules.txt");
00077     wl->writeElements()->insert(new WriteElement("tnow", true));
00078     wl->writeElements()->insert(new WriteElement(" "));
00079     wl->writeElements()->insert(new WriteElement("N[1]", true));
00080     wl->writeElements()->insert(new WriteElement(" "));
00081     wl->writeElements()->insert(new WriteElement("N[2]", true));
00082     wl->writeElements()->insert(new WriteElement(" "));
00083     wl->writeElements()->insert(new WriteElement("N[3]", true, true));
00084     //wl->writeElements()->insert(new WriteElement("temp[6]",true, true));
00085     //wl->writeElements()->insert(new WriteElement("tnow",true, true));
00086     asl->getAssignments()->insert(new Assign::Assignment("temp[1]", "k[1]*N[1]*N[2]"));
00087     asl->getAssignments()->insert(new Assign::Assignment("temp[2]", "k[2]*N[3]"));
00088     asl->getAssignments()->insert(new Assign::Assignment("temp[3]", "temp[1]+temp[2]"));
00089     asl->getAssignments()->insert(new Assign::Assignment("temp[4]", "if (temp[3]>0) temp[1]/temp[3]
00090     else 0"));
00091     asl->getAssignments()->insert(new Assign::Assignment("temp[5]", "if (temp[3]>0) (if (rnd<temp[4])
00092     1 else 2) else 0"));
00093     asl->getAssignments()->insert(new Assign::Assignment("N[1]", "N[1]+s[temp[5],1]"));
00094     asl->getAssignments()->insert(new Assign::Assignment("N[2]", "N[2]+s[temp[5],2]"));
00095     asl->getAssignments()->insert(new Assign::Assignment("N[3]", "N[3]+s[temp[5],3]"));
00096     asl->getAssignments()->insert(new Assign::Assignment("temp[6]", "1-exp(-temp[3]))"));
00097     asl->getAssignments()->insert(new Assign::Assignment("temp[7]", "expo(temp[6]))"));
00098     del->setDelayExpression("temp[7]");
00099     m->getSimulation()->setTerminatingCondition("(N[1]+N[2]+N[3])==0");
00100     m->getSimulation()->start();
00101     return 0;
00102
00103     /*
00104     Create* createl = new Create(m);
00105     Assign* assignl = new Assign(m);
00106     //Separate* sep1 = new Separate(m);
00107     Dispose* dispose1 = new Dispose(m);
00108     createl->nextComponents()->insert(assignl);
00109     assignl->nextComponents()->insert(dispose1);
00110     //sep1->nextComponents()->insert(dispose1);
00111     //sep1->->nextComponents()->insert(assignl);
```

```
00111     createl->setEntityType(new EntityType(m));
00112     createl->setMaxCreations("1");
00113     new Attribute(m, "attr1");
00114     Variable* varl = new Variable(m, "varl");
00115     std::string expression, index;
00116     for (int i = 1; i < 3; i++) {
00117         for (int j = 1; j < 3; j++) {
00118             index = std::to_string(i) + "," + std::to_string(j);
00119             varl->setInitialValue(index, 1.0*i + j / 10.0);
00120             expression = "attr1[" + index + "] + varl[" + index + "]";
00121             assign1->assignments()->insert(new Assign::Assignment("attr1[" + index + "]", expression));
00122         }
00123     }
00124     m->simulation()->start();
00125     return 0;
00126     */
00127 }
```

References [Write::FILE](#), [Assign::getAssignments\(\)](#), [Model::getInfos\(\)](#), [Simulator::getModels\(\)](#), [ModelComponent::getNextComponents\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [ModelManager::insert\(\)](#), [ConnectionManager::insert\(\)](#), [List< T >::insert\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [Util::modelSimulationInternal](#), [BaseConsoleGenesysApplication::setDelay\(\)](#), [Delay::setDelayExpression\(\)](#), [SourceModelComponent::setEntityType\(\)](#), [Formula::setExpression\(\)](#), [Write::setFilename\(\)](#), [Variable::setInitialValue\(\)](#), [SourceModelComponent::setMaxCreations\(\)](#), [ModellInfo::setProjectTitle\(\)](#), [ModelSimulation::setReplication\(\)](#), [ModelSimulation::setTerminatingCondition\(\)](#), [TraceManager::setTraceLevel\(\)](#), [Write::setWriteToType\(\)](#), [ModelSimulation::start\(\)](#), and [Write::writeElements\(\)](#).

Here is the call graph for this function:



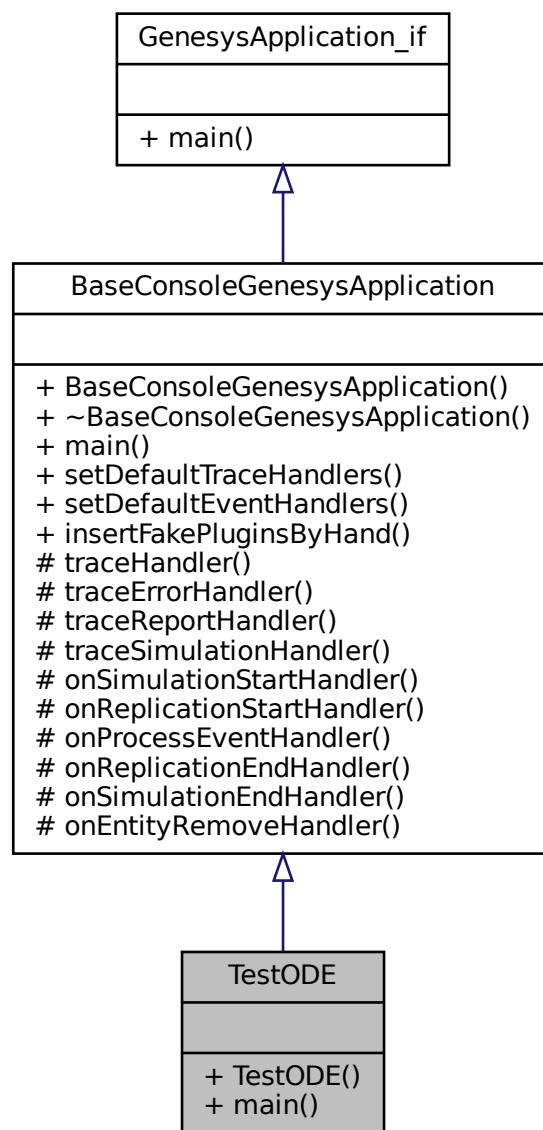
The documentation for this class was generated from the following files:

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

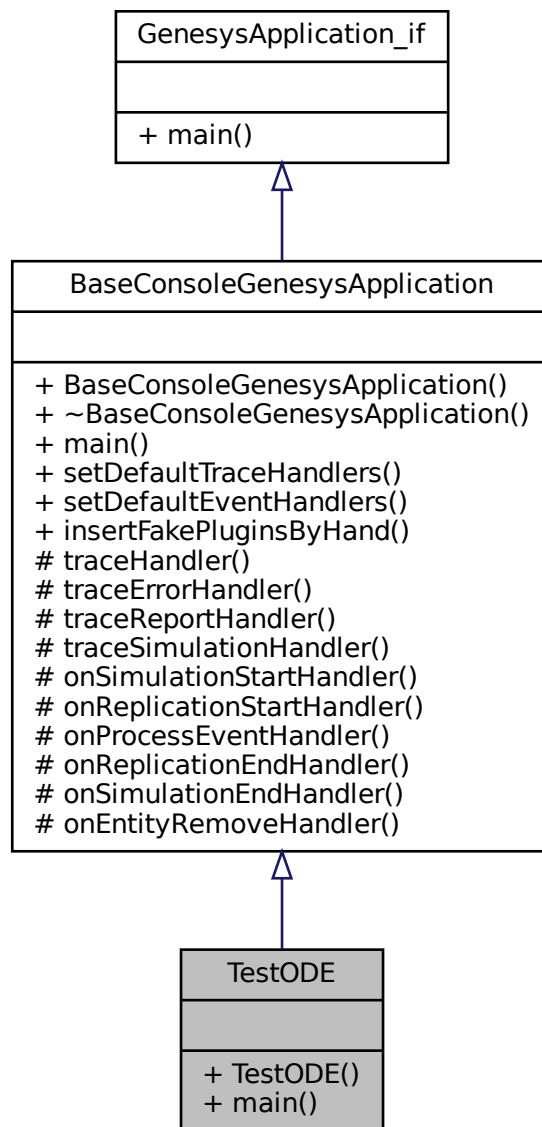
8.142 TestODE Class Reference

```
#include <TestFunctions.h>
```

Inheritance diagram for TestODE:



Collaboration diagram for TestODE:



Public Member Functions

- `TestODE ()`
- virtual int `main` (int argc, char **argv)

Additional Inherited Members

8.142.1 Detailed Description

Definition at line 19 of file [TestFunctions.h](#).

8.142.2 Constructor & Destructor Documentation

8.142.2.1 TestODE() TestODE::TestODE ()

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

```
00031             {
00032 }
```

8.142.3 Member Function Documentation

8.142.3.1 main() int TestODE::main (

```
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

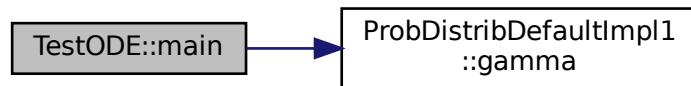
Definition at line 34 of file [TestFunctions.cpp](#).

```
00034             {
00035     using namespace std;
00036     //using namespace boost::numeric::odeint;
00037     //using namespace boost::math;
00038
00039     //beta_distribution<>* betadist = new beta_distribution<>(2.0, 2.0);
00040     //betadist->alpha();
00041     ProbDistribDefaultImpl pd;
00042     for (double x = -2.0; x <= 2.0; x += 0.1) {
00043         std::cout << x << ":" << pd.gamma(x, 2.0, 2.0) << std::endl;
00044     }
00045     //normdist->mean();
00046     //beta<double,double>* b = new beta<double,double>(2.0, 2.0);
00047     //b->
00048
00049     state_type x(2);
00050
00051     /*
00052     x[0] = 1.0; // start at x=1.0, p=0.0
00053     x[1] = 0.0;
00054     runge_kutta4< state_type > stepper;
00055     integrate_const(stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00056     std::cout << x[0] << " ; " << x[1] << std::endl;
00057     */
00058
00059     /*
00060     x[0] = 1.0; // start at x=1.0, p=0.0
00061     x[1] = 0.0;
00062     const double dt = 0.01;
00063     double t;
00064     for (t = 0.0; t < 0.2; t += dt) {
00065         std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00066         stepper.do_step(harmonic_oscillator, x, t, dt);
00067     }
00068     std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00069     */
00070
00071     /*
00072     x[0] = 1.0; // start at x=1.0, p=0.0
00073     x[1] = 0.0;
00074     typedef runge_kutta_cash_karp54< state_type > error_stepper_type;
00075     typedef controlled_runge_kutta< error_stepper_type > controlled_stepper_type;
00076     controlled_stepper_type controlled_stepper;
00077     //integrate_adaptive(controlled_stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00078     integrate_adaptive(make_controlled< error_stepper_type >(1.0e-20, 1.0e-16), harmonic_oscillator,
00079     x, 0.0, 0.2, 0.01);
00080     std::cout << x[0] << " ; " << x[1] << std::endl;
00081     */
00082     return 0;
```

```
00083 }
```

References [ProbDistribDefaultImpl1::gamma\(\)](#).

Here is the call graph for this function:



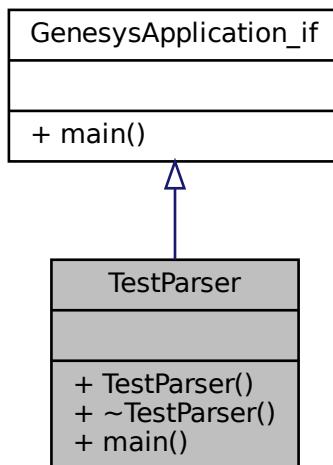
The documentation for this class was generated from the following files:

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

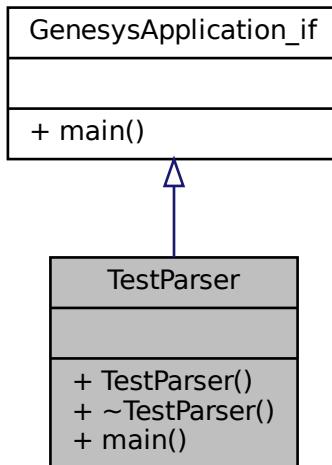
8.143 TestParser Class Reference

```
#include <TestParser.h>
```

Inheritance diagram for TestParser:



Collaboration diagram for TestParser:



Public Member Functions

- `TestParser ()`
- virtual `~TestParser ()=default`
- virtual int `main (int argc, char **argv)`

8.143.1 Detailed Description

Definition at line 19 of file [TestParser.h](#).

8.143.2 Constructor & Destructor Documentation

8.143.2.1 `TestParser()` `TestParser::TestParser ()`

Definition at line 19 of file [TestParser.cpp](#).

```
00019          {
00020 }
```

8.143.2.2 `~TestParser()` `virtual TestParser::~TestParser () [virtual], [default]`

8.143.3 Member Function Documentation

8.143.3.1 main() int TestParser::main (

```
int argc,
char ** argv ) [virtual]
```

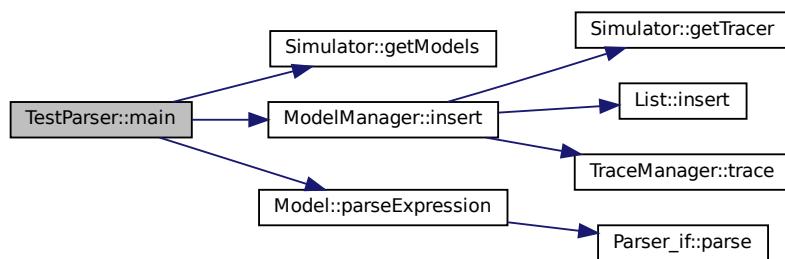
Implements [GenesysApplication_if](#).

Definition at line 22 of file [TestParser.cpp](#).

```
00022     Simulator* simulator = new Simulator();
00023     Model* model = new Model(simulator);
00024     simulator->getModels()->insert(model);
00025     double value;
00026     bool success;
00027     std::string errorMsg = "";
00028     value = model->parseExpression("NORM(20,10)", &success, &errorMsg);
00029     std::cout << value << std::endl;
00030     value = model->parseExpression("-10+1+2+3", &success, &errorMsg);
00031     std::cout << value << std::endl;
00032     value = model->parseExpression("1+2+3-10", &success, &errorMsg);
00033     std::cout << value << std::endl;
00034     return 0;
00035 }
00036 }
```

References [Simulator::getModels\(\)](#), [ModelManager::insert\(\)](#), and [Model::parseExpression\(\)](#).

Here is the call graph for this function:



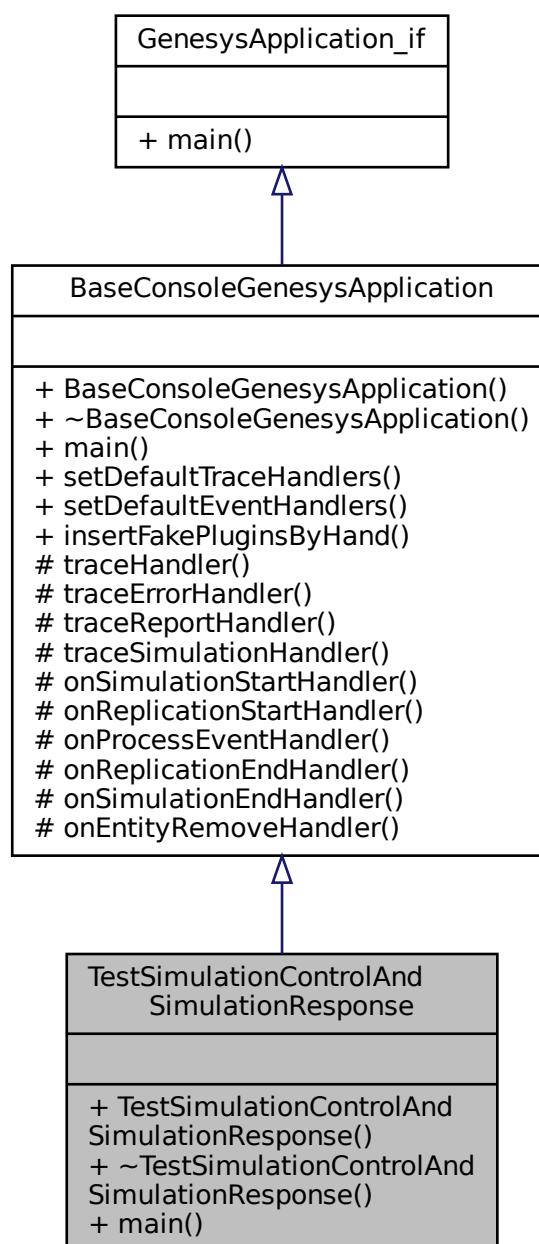
The documentation for this class was generated from the following files:

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

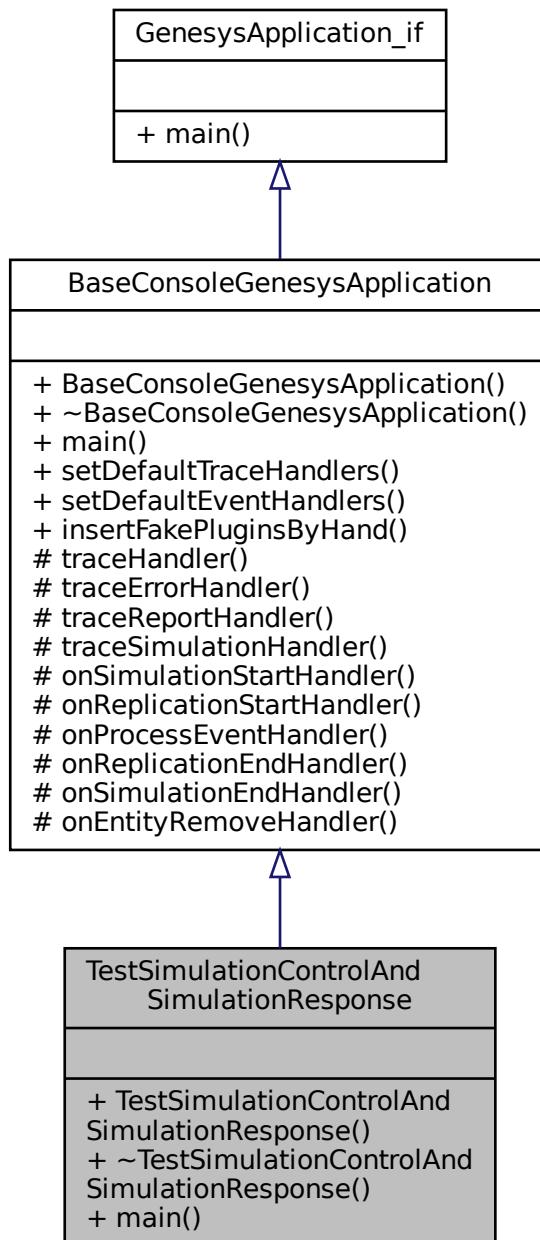
8.144 TestSimulationControlAndSimulationResponse Class Reference

```
#include <TestSimulationControlAndSimulationResponse.h>
```

Inheritance diagram for TestSimulationControlAndSimulationResponse:



Collaboration diagram for TestSimulationControlAndSimulationResponse:



Public Member Functions

- `TestSimulationControlAndSimulationResponse ()`
- virtual `~TestSimulationControlAndSimulationResponse ()=default`
- virtual int `main (int argc, char **argv)`

Additional Inherited Members

8.144.1 Detailed Description

Definition at line 19 of file [TestSimulationControlAndSimulationResponse.h](#).

8.144.2 Constructor & Destructor Documentation

8.144.2.1 TestSimulationControlAndSimulationResponse() `TestSimulationControlAndSimulation<→Response::TestSimulationControlAndSimulationResponse ()`

Definition at line 21 of file [TestSimulationControlAndSimulationResponse.cpp](#).

```
00021
00022 }
```

8.144.2.2 ~TestSimulationControlAndSimulationResponse() `virtual TestSimulationControlAnd←SimulationResponse::~TestSimulationControlAndSimulationResponse () [virtual], [default]`

8.144.3 Member Function Documentation

8.144.3.1 main() `int TestSimulationControlAndSimulationResponse::main (int argc, char ** argv) [virtual]`

Implements [BaseConsoleGenesysApplication](#).

Definition at line 24 of file [TestSimulationControlAndSimulationResponse.cpp](#).

```
00024
00025     Simulator* simulator = new Simulator();
00026     TraceManager* tm = simulator->getTracer();
00027     this->setDefaultTraceHandlers(tm);
00028     tm->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00029     this->insertFakePluginsByHand(simulator);
00030
00031     simulator->getModels()->loadModel("./temp/forthExampleOfSimulation.txt");
00032     Model* model = simulator->getModels()->current();
00033     model->getSimulation()->start();
00034     tm->trace("\nResponses:");
00035     for (std::list<SimulationResponse*>::iterator it = model->getResponses()->list()->begin(); it != model->getResponses()->list()->end(); it++) {
00036         tm->trace((*it)->getName() + ": " + std::to_string((*it)->getValue()));
00037     }
00038
00039     /*
00040     Model* model = new Model(simulator);
00041     model->show();
00042
00043     std::cout << "NumRepl antes: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00044     model->getInfos()->setNumberOfReplications(10);
00045     std::cout << "NumRepl depois: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00046     SimulationControl* control = model->getControls()->front();
00047     std::cout << control->getName() << " antes: " << control->getValue() << std::endl;
00048     control->setValue(20);
00049     std::cout << control->getName() << " depois: " << control->getValue() << std::endl;
```

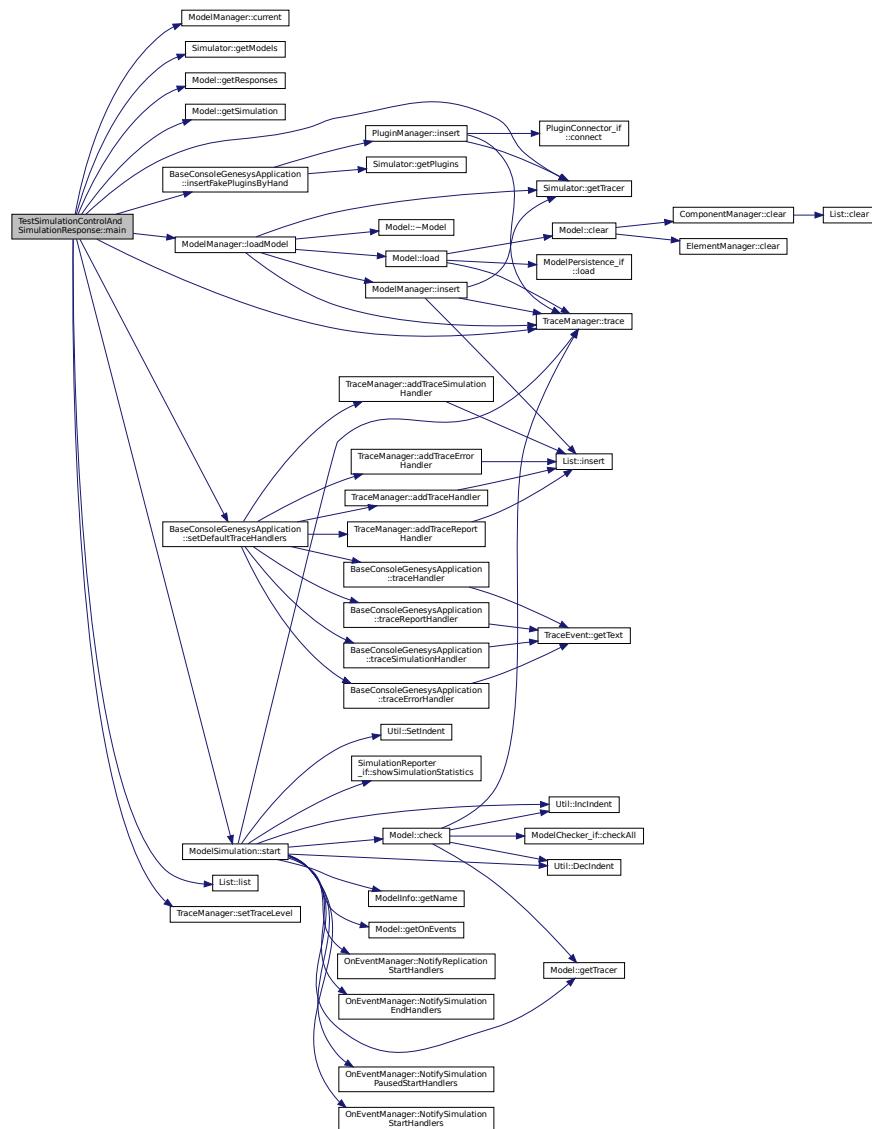
```

00050     std::cout << "NumRepl depois: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00051     */
00052     return 0;
00053 }

```

References [ModelManager::current\(\)](#), [Util::everythingMostDetailed](#), [Simulator::getModels\(\)](#), [Model::getResponses\(\)](#), [Model::getSimulation\(\)](#), [Simulator::getTracer\(\)](#), [BaseConsoleGenesysApplication::insertFakePluginsByHand\(\)](#), [List< T >::list\(\)](#), [ModelManager::loadModel\(\)](#), [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#), [TraceManager::setTraceLevel\(\)](#), [ModelSimulation::start\(\)](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



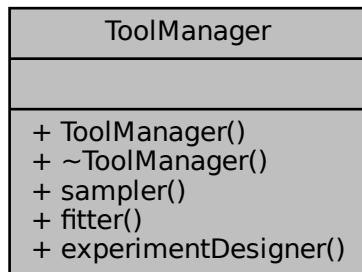
The documentation for this class was generated from the following files:

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

8.145 ToolManager Class Reference

```
#include <ToolManager.h>
```

Collaboration diagram for ToolManager:



Public Member Functions

- `ToolManager (Simulator *_simulator)`
- `virtual ~ToolManager ()=default`
- `Sampler_if * sampler () const`

Returns the Sampler, used to generate samples accordingly to a probability distribution.

- `Fitter_if * fitter () const`
- `ExperimentManager_if * experimentDesigner () const`

Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

8.145.1 Detailed Description

Definition at line 23 of file [ToolManager.h](#).

8.145.2 Constructor & Destructor Documentation

8.145.2.1 ToolManager() ToolManager::ToolManager (Simulator * _simulator)

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

```

00017
00018     _simulator = simulator;
00019     //{
00020     _fitter = new Traits<Fitter_if>::Implementation();
00021     _sampler = new Traits<Sampler_if>::Implementation();
00022     _processAnalyser = new Traits<ExperimentManager_if>::Implementation();
00023 }
```

8.145.2.2 ~ToolManager() virtual ToolManager::~ToolManager () [virtual], [default]

8.145.3 Member Function Documentation

8.145.3.1 experimentDesigner() ExperimentManager_if * ToolManager::experimentDesigner () const

Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

Definition at line 33 of file [ToolManager.cpp](#).

```
00033
00034     return _processAnalyser;
00035 }
```

8.145.3.2 fitter() Fitter_if * ToolManager::fitter () const

Definition at line 29 of file [ToolManager.cpp](#).

```
00029
00030     return _fitter;
00031 }
```

8.145.3.3 sampler() Sampler_if * ToolManager::sampler () const

Returns the Sampler, used to generate samples accordingly to a probability distribution.

Definition at line 25 of file [ToolManager.cpp](#).

```
00025
00026     return _sampler;
00027 }
```

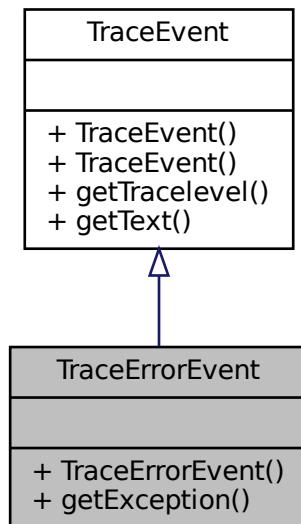
The documentation for this class was generated from the following files:

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

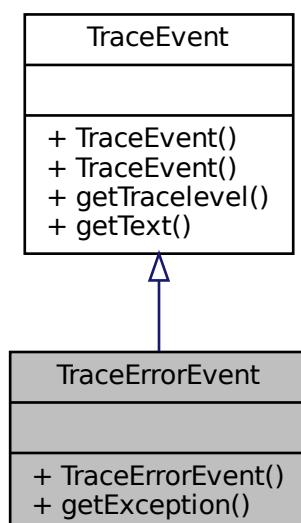
8.146 TraceErrorEvent Class Reference

```
#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`

8.146.1 Detailed Description

Definition at line 50 of file [TraceManager.h](#).

8.146.2 Constructor & Destructor Documentation

8.146.2.1 `TraceErrorEvent()`

```
TraceErrorEvent::TraceErrorEvent (
    std::string text,
    std::exception e) [inline]
```

Definition at line 53 of file [TraceManager.h](#).

```
00053     : TraceEvent(text,
00054         Util::TraceLevel::errorFatal) {
00055         _e = e;
}
```

8.146.3 Member Function Documentation

8.146.3.1 `getException()`

```
std::exception TraceErrorEvent::getException () const [inline]
```

Definition at line 57 of file [TraceManager.h](#).

```
00057 {
00058     return _e;
00059 }
```

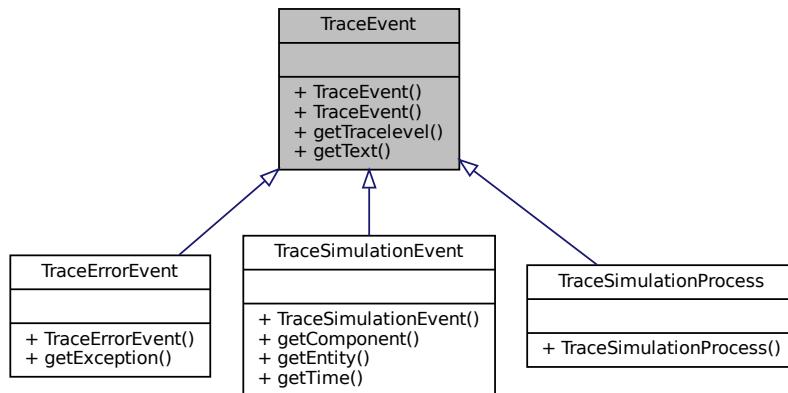
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

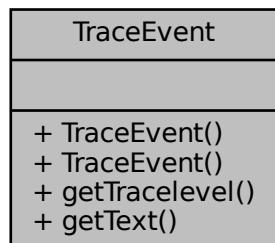
8.147 TraceEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceEvent:



Collaboration diagram for TraceEvent:



Public Member Functions

- `TraceEvent (Util::TraceLevel level, std::string text)`
- `TraceEvent (std::string text, Util::TraceLevel level=Util::TraceLevel::componentInternal)`
- `Util::TraceLevel getTracelevel () const`
- `std::string getText () const`

8.147.1 Detailed Description

Definition at line 25 of file [TraceManager.h](#).

8.147.2 Constructor & Destructor Documentation

8.147.2.1 TraceEvent() [1/2] `TraceEvent::TraceEvent (`

```
    Util::TraceLevel level,
    std::string text ) [inline]
```

Definition at line 28 of file [TraceManager.h](#).

```
00028     _tracelevel = level;
00029     _text = text;
00030 }
00031 }
```

8.147.2.2 TraceEvent() [2/2] `TraceEvent::TraceEvent (`

```
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::componentInternal ) [inline]
```

Definition at line 33 of file [TraceManager.h](#).

```
00033     _tracelevel = level;
00034     _text = text;
00035 }
00036 }
```

8.147.3 Member Function Documentation

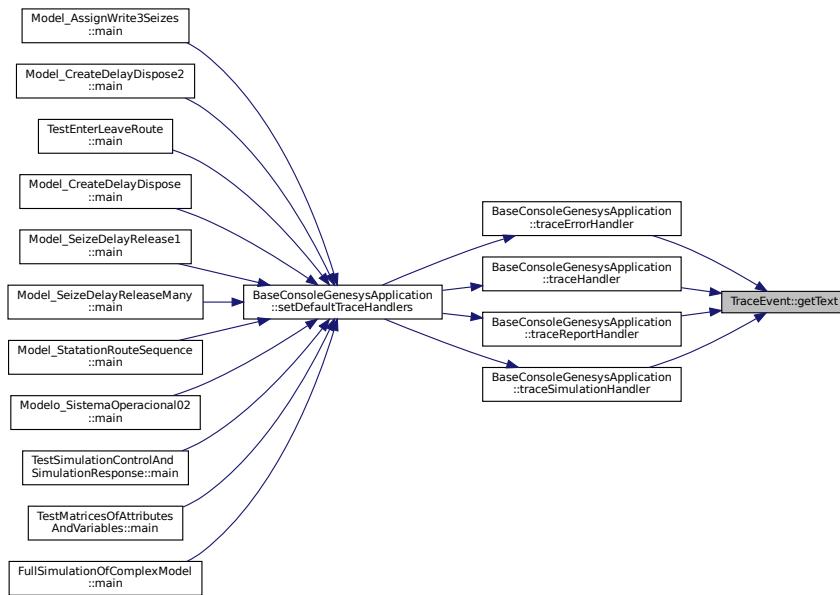
8.147.3.1 getText() `std::string TraceEvent::getText () const [inline]`

Definition at line 42 of file [TraceManager.h](#).

```
00042     {
00043         return _text;
00044     }
```

Referenced by [BaseConsoleGenesysApplication::traceErrorHandler\(\)](#), [BaseConsoleGenesysApplication::traceHandler\(\)](#), [BaseConsoleGenesysApplication::traceReportHandler\(\)](#), and [BaseConsoleGenesysApplication::traceSimulationHandler\(\)](#).

Here is the caller graph for this function:



8.147.3.2 getTracelevel() [Util::TraceLevel](#) TraceEvent::getTracelevel () const [inline]

Definition at line 38 of file [TraceManager.h](#).

```

00038
00039     return _tracelevel;
00040 }
```

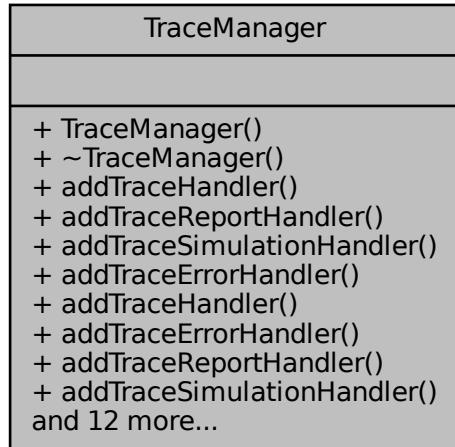
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

8.148 TraceManager Class Reference

```
#include <TraceManager.h>
```

Collaboration diagram for TraceManager:



Public Member Functions

- `TraceManager (Simulator *simulator)`
- `virtual ~TraceManager ()=default`
- `void addTraceHandler (traceListener traceListener)`
- `void addTraceReportHandler (traceListener traceReportListener)`
- `void addTraceSimulationHandler (traceSimulationListener traceSimulationListener)`
- `void addTraceErrorHandler (traceErrorListener traceErrorListener)`
- template<typename Class >
 `void addTraceHandler (Class *object, void(Class::*function)(TraceEvent))`
- template<typename Class >
 `void addTraceErrorHandler (Class *object, void(Class::*function)(TraceErrorEvent))`
- template<typename Class >
 `void addTraceReportHandler (Class *object, void(Class::*function)(TraceEvent))`
- template<typename Class >
 `void addTraceSimulationHandler (Class *object, void(Class::*function)(TraceSimulationEvent))`
- `void trace (Util::TraceLevel level, std::string text)`
- `void traceError (std::exception e, std::string text)`
- `void traceReport (Util::TraceLevel level, std::string text)`
- `void traceSimulation (Util::TraceLevel level, double time, Entity *entity, ModelComponent *component, std::string text)`
- `void trace (std::string text, Util::TraceLevel level=Util::TraceLevel::componentInternal)`
- `void traceError (std::string text, std::exception e)`
- `void traceReport (std::string text, Util::TraceLevel level=Util::TraceLevel::report)`
- `void traceSimulation (double time, Entity *entity, ModelComponent *component, std::string text, Util::TraceLevel level=Util::TraceLevel::componentInternal)`
- `List< std::string > * errorMessages () const`
- `void setTraceLevel (Util::TraceLevel _traceLevel)`
- `Util::TraceLevel getTraceLevel () const`
- `Simulator * getParentSimulator () const`

8.148.1 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.

Definition at line 117 of file [TraceManager.h](#).

8.148.2 Constructor & Destructor Documentation

8.148.2.1 TraceManager() [TraceManager::TraceManager](#) (Simulator * simulator)

Definition at line 19 of file [TraceManager.cpp](#).

```
00019                                         { // (Model* model) {
00020     _simulator = simulator;
00021     _debugged = Traits<Model>::debugged;
00022     _traceLevel = Traits<Model>::traceLevel;
00023 }
```

8.148.2.2 ~TraceManager() [virtual TraceManager::~TraceManager](#) () [virtual], [default]

8.148.3 Member Function Documentation

8.148.3.1 addTraceErrorHandler() [1/2] [template<typename Class > void TraceManager::addTraceErrorHandler](#) (

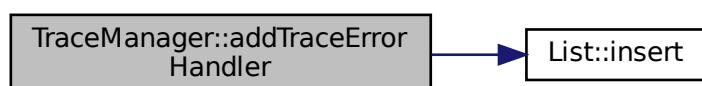
```
Class * object,
void(Class::*)(TraceErrorEvent) function )
```

Definition at line 180 of file [TraceManager.h](#).

```
00180
00181     {
00182         this->_traceErrorHandlerMethod->insert(std::bind(function, object, std::placeholders::_1));
00183     }
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



8.148.3.2 addTraceErrorHandler() [2/2] void TraceManager::addTraceErrorHandler (traceErrorListener traceErrorListener)

Definition at line 45 of file [TraceManager.cpp](#).

```
00045
00046     this->_traceErrorHandlers->insert(traceErrorListener);
00047 }
```

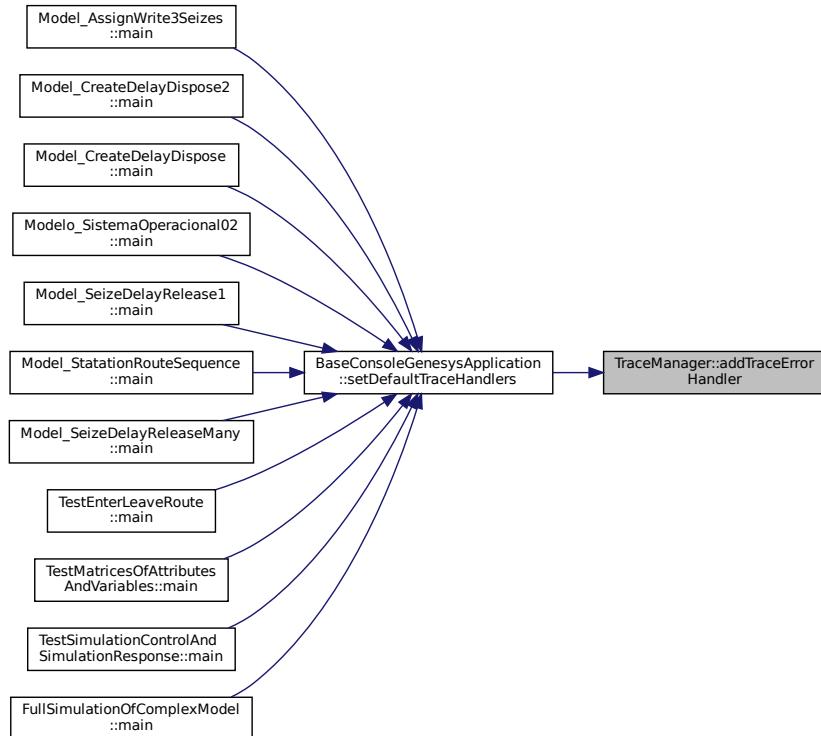
References [List< T >::insert\(\)](#).

Referenced by [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.148.3.3 addTraceHandler() [1/2] `template<typename Class >`
void TraceManager::addTraceHandler (
 Class * object,
 void(Class::*)(TraceEvent) function)

Definition at line 176 of file [TraceManager.h](#).

```
00176     {  
00177         this->_traceHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));  
00178     }
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



8.148.3.4 addTraceHandler() [2/2] void TraceManager::addTraceHandler (
 [traceListener](#) traceListener)

Definition at line 37 of file [TraceManager.cpp](#).

```
00037 {  
00038     this->_traceHandlers->insert(traceListener);  
00039 }
```

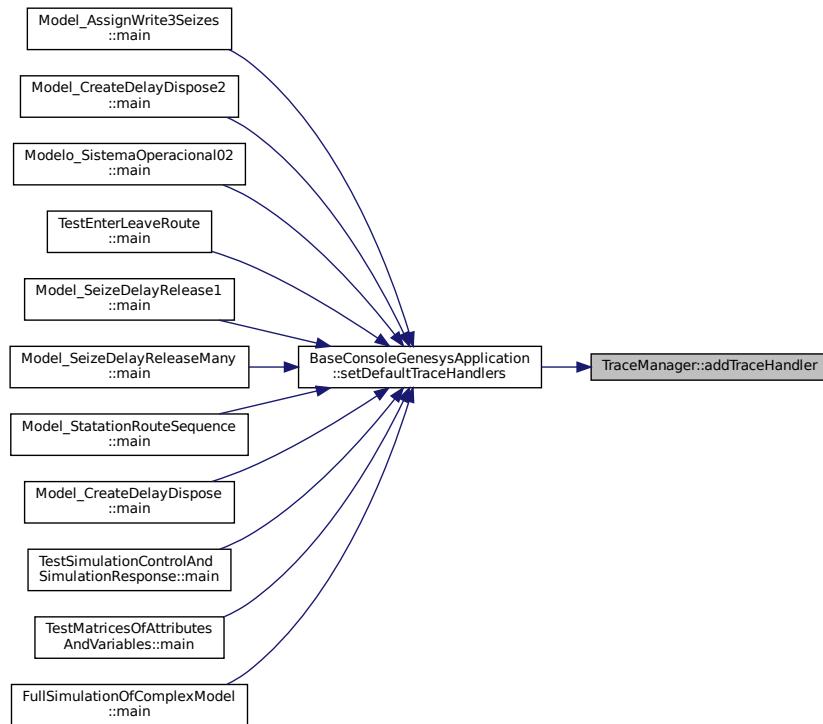
References [List< T >::insert\(\)](#).

Referenced by [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.148.3.5 addTraceReportHandler() [1/2] template<typename Class >
void TraceManager::addTraceReportHandler (
    Class * object,
    void(Class::*)(TraceEvent) function )
```

Definition at line 184 of file [TraceManager.h](#).

```
00184
00185     {
00186     this->_traceReportHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
}
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



```
8.148.3.6 addTraceReportHandler() [2/2] void TraceManager::addTraceReportHandler (
    traceListener traceReportListener )
```

Definition at line 49 of file [TraceManager.cpp](#).

```
00049
00050     this->_traceReportHandlers->insert(traceReportListener);
00051 }
```

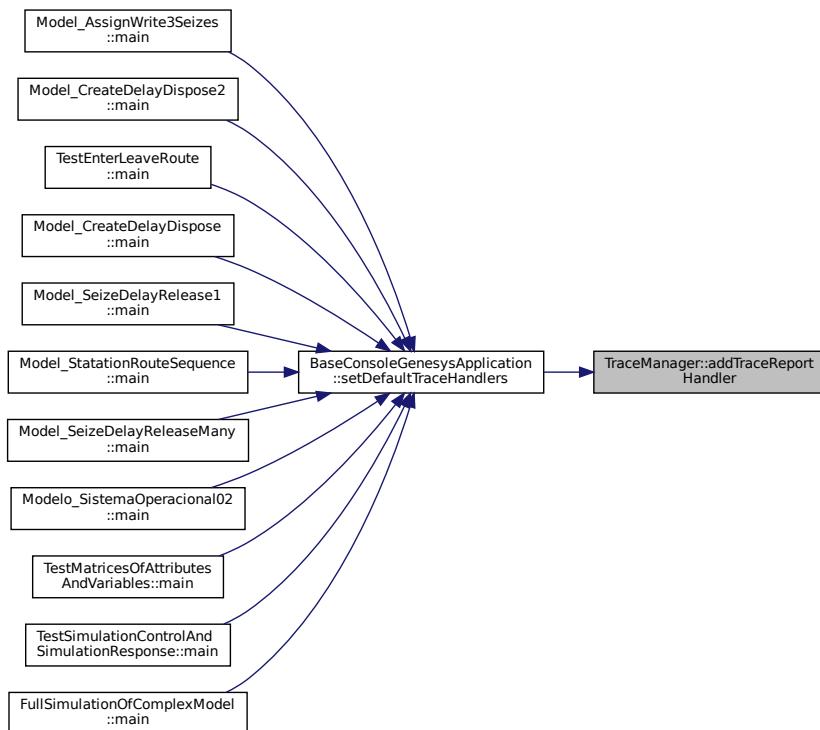
References [List< T >::insert\(\)](#).

Referenced by [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.148.3.7 addTraceSimulationHandler() [1/2] template<typename Class >
void TraceManager::addTraceSimulationHandler (

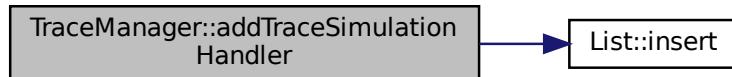
```
    Class * object,
    void(Class::*)(TraceSimulationEvent) function )
```

Definition at line 188 of file [TraceManager.h](#).

```
00188
00189     {
00190         this->_traceSimulationHandlersMethod->insert(std::bind(function, object,
00190             std::placeholders::_1));
00190     }
```

References [List< T >::insert\(\)](#).

Here is the call graph for this function:



8.148.3.8 addTraceSimulationHandler() [2/2] void TraceManager::addTraceSimulationHandler (

```
    traceSimulationListener traceSimulationListener )
```

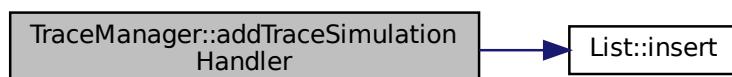
Definition at line 41 of file [TraceManager.cpp](#).

```
00041
00042     this->_traceSimulationHandlers->insert(traceSimulationListener);
00043 }
```

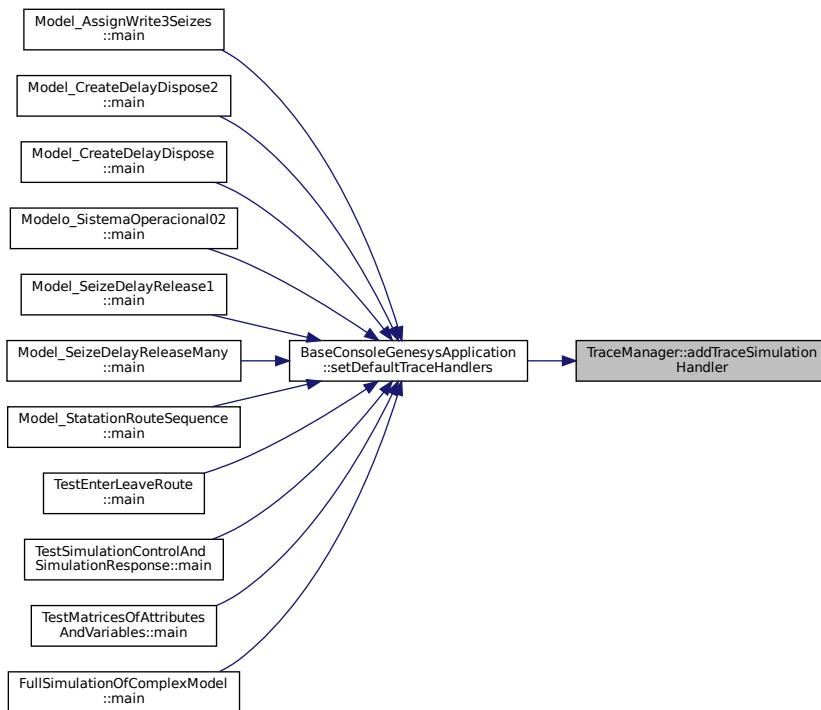
References [List< T >::insert\(\)](#).

Referenced by [BaseConsoleGenesysApplication::setDefaultTraceHandlers\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.148.3.9 errorMessages() `List< std::string > * TraceManager::errorMessages () const`

Definition at line 123 of file [TraceManager.cpp](#).

```
00123
00124     return _errorMessages;
00125 }
```

8.148.3.10 getParentSimulator() `Simulator * TraceManager::getParentSimulator () const`

Definition at line 33 of file [TraceManager.cpp](#).

```
00033
00034     return _simulator;
00035 }
```

8.148.3.11 getTraceLevel() `Util::TraceLevel TraceManager::getTraceLevel () const`

Definition at line 29 of file [TraceManager.cpp](#).

```
00029
00030     return _traceLevel;
00031 }
```

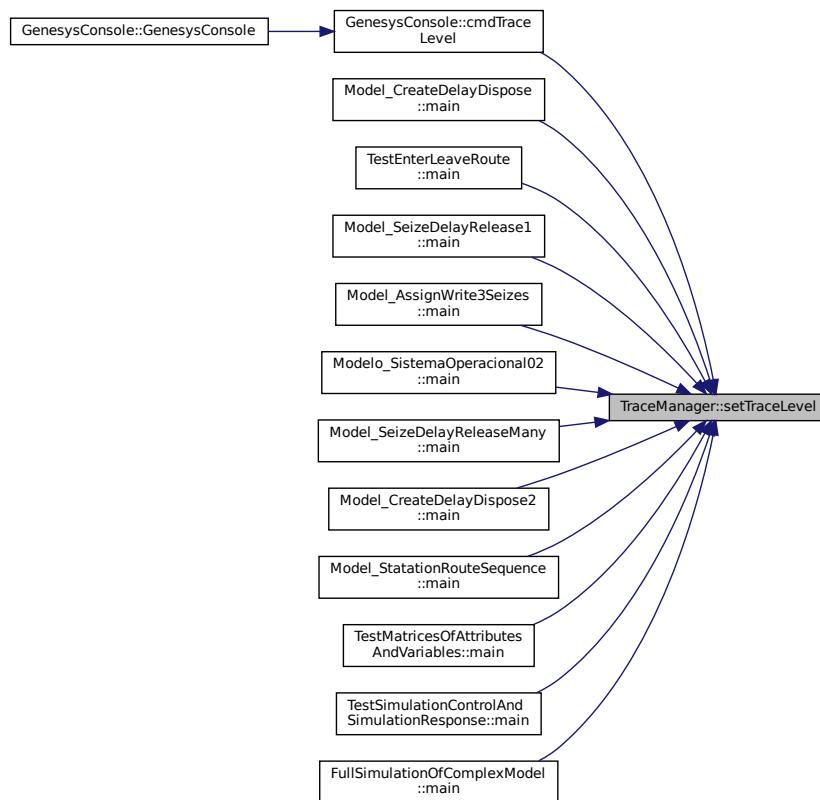
```
8.148.3.12 setTraceLevel() void TraceManager::setTraceLevel (
    Util::TraceLevel _traceLevel )
```

Definition at line 25 of file [TraceManager.cpp](#).

```
00025
00026     this->_traceLevel = _traceLevel;
00027 }
```

Referenced by [GenesysConsole::cmdTraceLevel\(\)](#), [Model_CreateDelayDispose::main\(\)](#), [Model_StationRouteSequence::main\(\)](#), [TestEnterLeaveRoute::main\(\)](#), [Modelo_SistemaOperacional02::main\(\)](#), [Model_CreateDelayDispose2::main\(\)](#), [Model_SeizeDelayReleaseMany::main\(\)](#), [Model_SeizeDelayRelease1::main\(\)](#), [Model_AssignWrite3Seizes::main\(\)](#), [TestMatricesOfAttributesAndVariables::main\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), and [FullSimulationOfComplexModel::main\(\)](#).

Here is the caller graph for this function:



```
8.148.3.13 trace() [1/2] void TraceManager::trace (
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::componentInternal )
```

Definition at line 57 of file [TraceManager.cpp](#).

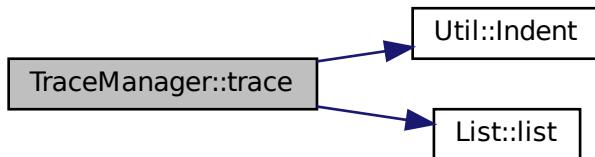
```
00057
00058     if (_traceConditionPassed(level)) {
00059         //text = std::to_string(static_cast<int> (level)) + ". " + Util::Indent() + text;
00060         text = Util::Indent() + text;
00061         TraceEvent e = TraceEvent(text, level);
00062         /* \todo:--: somewhere in future it should be interesting to use "auto" and c++17 at least */
```

```

00063     for (std::list<traceListener>::iterator it = this->_traceHandlers->list()->begin(); it != _traceHandlers->list()->end(); it++) {
00064         (*it)(e);
00065     }
00066     for (std::list<traceListenerMethod>::iterator it =
00067         this->_traceHandlersMethod->list()->begin(); it != _traceHandlersMethod->list()->end(); it++) {
00068         (*it)(e);
00069     }
00070 }
00071 }
```

References [Util::Indent\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



8.148.3.14 trace() [2/2] void TraceManager::trace (

```

        Util::TraceLevel level,
        std::string text )
```

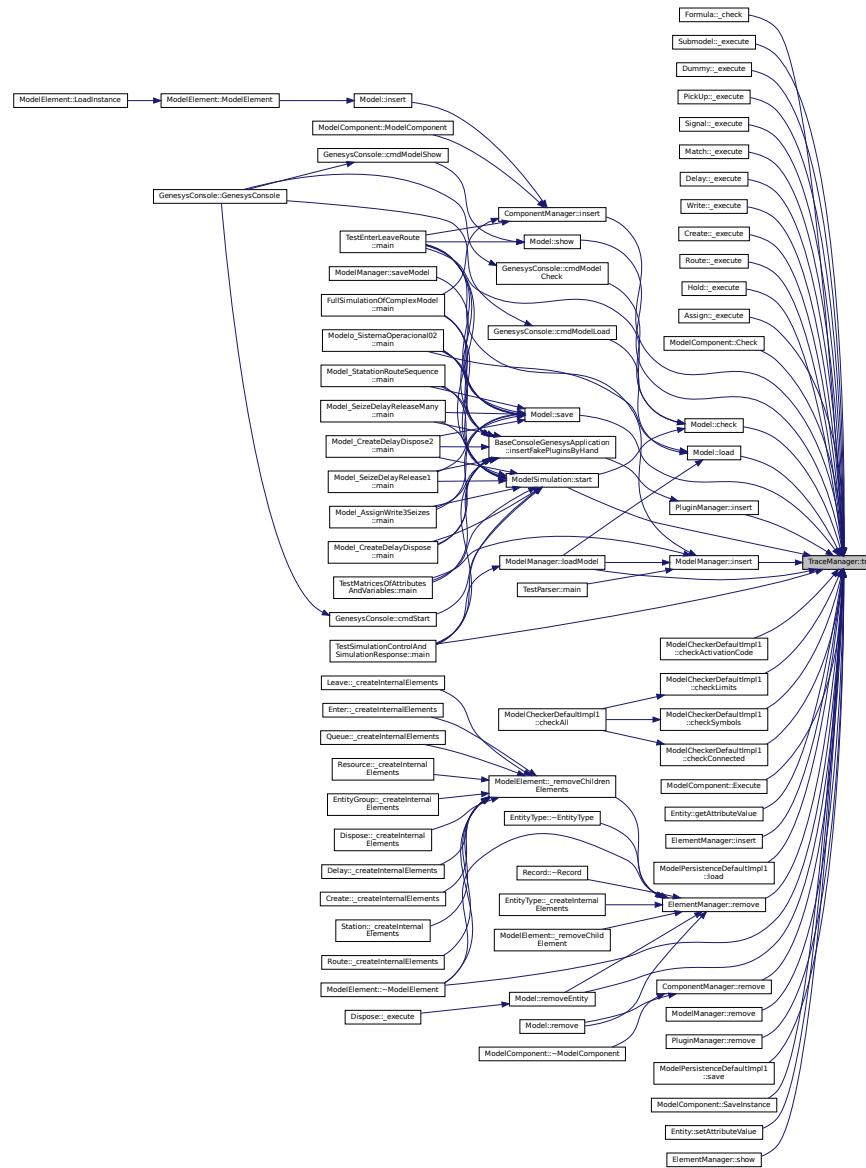
Definition at line 53 of file [TraceManager.cpp](#).

```

00053
00054     trace(text, level);
00055 }
```

Referenced by [Formula::_check\(\)](#), [Submodel::_execute\(\)](#), [Dummy::_execute\(\)](#), [PickUp::_execute\(\)](#), [Signal::_execute\(\)](#), [Match::_execute\(\)](#), [Delay::_execute\(\)](#), [Write::_execute\(\)](#), [Create::_execute\(\)](#), [Route::_execute\(\)](#), [Hold::_execute\(\)](#), [Assign::_execute\(\)](#), [ModelComponent::Check\(\)](#), [Model::check\(\)](#), [ModelCheckerDefaultImpl1::checkActivationCode\(\)](#), [ModelCheckerDefaultImpl1::checkConnected\(\)](#), [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [ModelCheckerDefaultImpl1::checkSymbol\(\)](#), [ModelComponent::Execute\(\)](#), [Entity::getAttributeValue\(\)](#), [ComponentManager::insert\(\)](#), [ModelManager::insert\(\)](#), [PluginManager::insert\(\)](#), [ElementManager::insert\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [Model::load\(\)](#), [ModelManager::loadModel\(\)](#), [TestSimulationControlAndSimulationResponse::main\(\)](#), [ComponentManager::remove\(\)](#), [ModelManager::remove\(\)](#), [PluginManager::remove\(\)](#), [ElementManager::remove\(\)](#), [Model::removeEntity\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [Model::save\(\)](#), [ModelComponent::SaveInstance\(\)](#), [Entity::setAttributeValue\(\)](#), [Model::show\(\)](#), [ElementManager::show\(\)](#), [ModelSimulation::start\(\)](#), and [ModelElement::~ModelElement\(\)](#).

Here is the caller graph for this function:



```

8.148.3.15 traceError() [1/2] void TraceManager::traceError (
    std::exception e,
    std::string text )
  
```

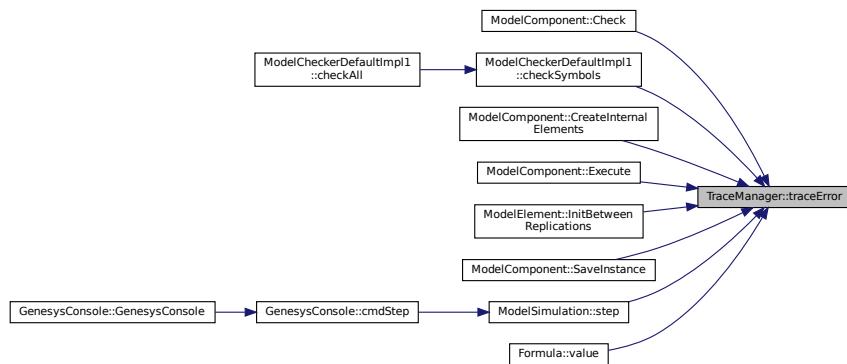
Definition at line 73 of file [TraceManager.cpp](#).

```

00073
00074     traceError(text, e);
00075 }
  
```

Referenced by [ModelComponent::Check\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [ModelComponent::CreateInternalElements\(\)](#), [ModelComponent::Execute\(\)](#), [ModelElement::InitBetweenReplications\(\)](#), [ModelComponent::SaveInstance\(\)](#), [ModelSimulation::step\(\)](#), and [Formula::value\(\)](#).

Here is the caller graph for this function:



8.148.3.16 traceError() [2/2] void TraceManager::traceError (std::string text, std::exception e)

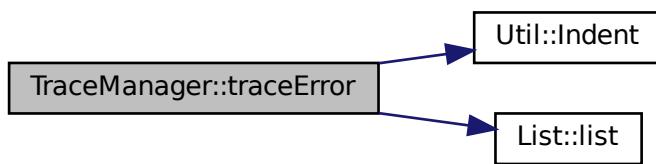
Definition at line 77 of file [TraceManager.cpp](#).

```

00077
00078     text = Util::Indent() + text;
00079     TraceErrorEvent exceptEvent = TraceErrorEvent(text, e);
00080     /* \todo--- somewhere in future it should be interesting to use "auto" and c++17 at least */
00081     for (std::list<traceErrorListener>::iterator it = this->_traceErrorHandlers->list()->begin(); it
00082         != _traceErrorHandlers->list()->end(); it++) {
00083         (*it)(exceptEvent);
00084     }
00085     for (std::list<traceErrorListenerMethod>::iterator it =
00086           this->_traceErrorHandlersMethod->list()->begin(); it != _traceErrorHandlersMethod->list()->end();
00087         it++) {
00088         (*it)(exceptEvent);
00089     }
00090 }
```

References [Util::Indent\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



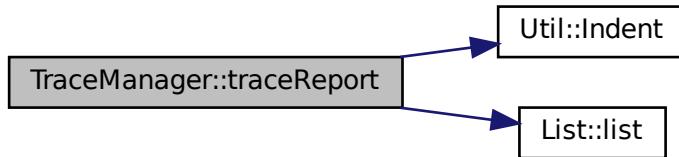
```
8.148.3.17 traceReport() [1/2] void TraceManager::traceReport (
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::report )
```

Definition at line 110 of file [TraceManager.cpp](#).

```
00110     if (_traceConditionPassed(level)) {
00111         text = Util::Indent() + text;
00112         TraceEvent e = TraceEvent(text, level);
00113         for (std::list<traceListener>::iterator it = this->_traceReportHandlers->list()->begin(); it
00114             != _traceReportHandlers->list()->end(); it++) {
00115             (*it)(e);
00116         }
00117         for (std::list<traceListenerMethod>::iterator it =
00118             this->_traceReportHandlersMethod->list()->begin(); it != _traceReportHandlersMethod->list()->end();
00119             it++) {
00120             (*it)(e);
00121         }
00121 }
```

References [Util::Indent\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



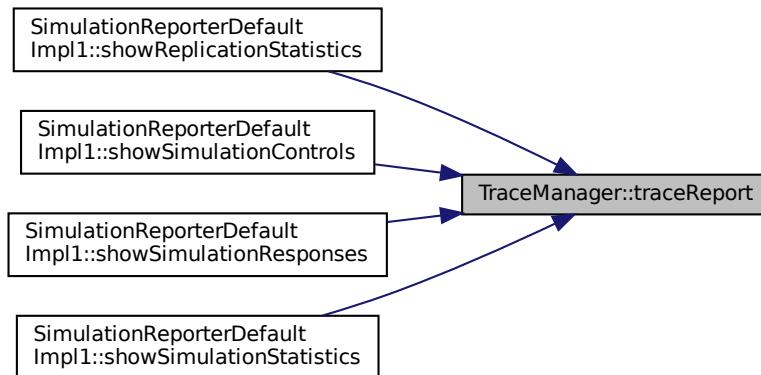
```
8.148.3.18 traceReport() [2/2] void TraceManager::traceReport (
    Util::TraceLevel level,
    std::string text )
```

Definition at line 106 of file [TraceManager.cpp](#).

```
00106     traceReport(text, level);
00107
00108 }
```

Referenced by [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [SimulationReporterDefaultImpl1::showSimulationControl\(\)](#), [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.148.3.19 traceSimulation() [1/2] void TraceManager::traceSimulation (

```

        double time,
        Entity * entity,
        ModelComponent * component,
        std::string text,
        Util::TraceLevel level = Util::TraceLevel::componentInternal )
  
```

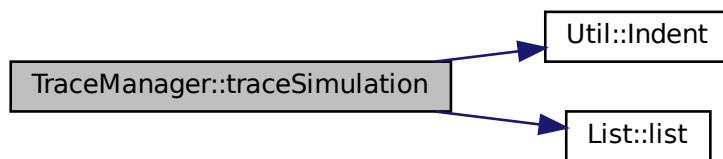
Definition at line 93 of file [TraceManager.cpp](#).

```

00093 {
00094     if (_traceConditionPassed(level)) {
00095         text = Util::Indent() + text;
00096         TraceSimulationEvent e = TraceSimulationEvent(level, time, entity, component, text);
00097         for (std::list<traceSimulationListener>::iterator it =
00098             this->_traceSimulationHandlers->list()->begin(); it != _traceSimulationHandlers->list()->end(); it++)
00099         {
00100             (*it)(e);
00101         }
00102     }
00103 }
00104 }
```

References [Util::Indent\(\)](#), and [List< T >::list\(\)](#).

Here is the call graph for this function:



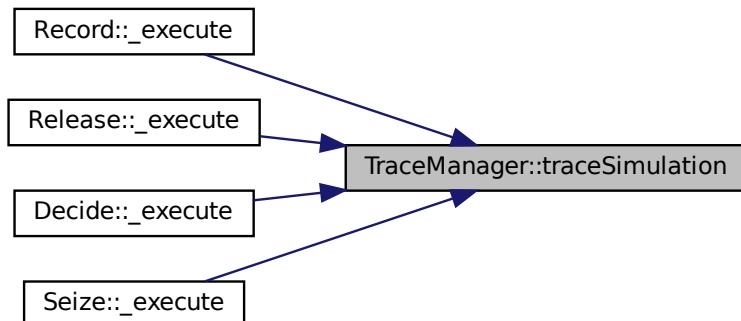
```
8.148.3.20 traceSimulation() [2/2] void TraceManager::traceSimulation (
    Util::TraceLevel level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text )
```

Definition at line 89 of file [TraceManager.cpp](#).

```
00089
00090     {
00090     traceSimulation(time, entity, component, text, level);
00091 }
```

Referenced by [Record::_execute\(\)](#), [Release::_execute\(\)](#), [Decide::_execute\(\)](#), and [Seize::_execute\(\)](#).

Here is the caller graph for this function:



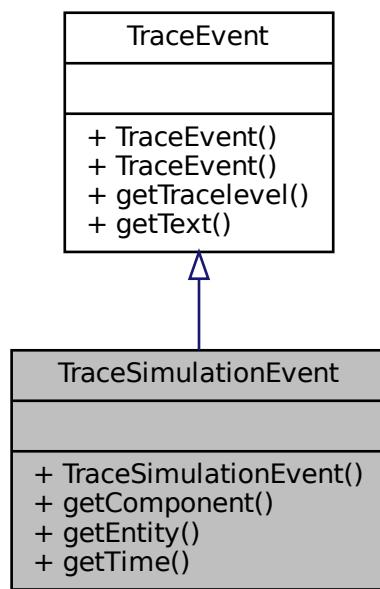
The documentation for this class was generated from the following files:

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

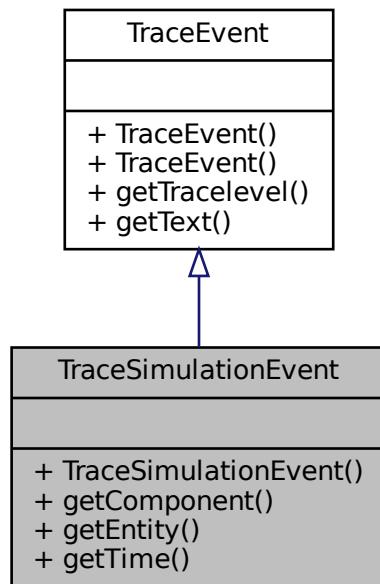
8.149 TraceSimulationEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationEvent:



Collaboration diagram for TraceSimulationEvent:



Public Member Functions

- `TraceSimulationEvent (Util::TraceLevel level, double time, Entity *entity, ModelComponent *component, std::string text)`
- `ModelComponent * getComponent () const`
- `Entity * getEntity () const`
- `double getTime () const`

8.149.1 Detailed Description

Definition at line 64 of file [TraceManager.h](#).

8.149.2 Constructor & Destructor Documentation

8.149.2.1 TraceSimulationEvent()

```
TraceSimulationEvent::TraceSimulationEvent (
    Util::TraceLevel level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text) [inline]
```

Definition at line 67 of file [TraceManager.h](#).

```
00067             : TraceEvent(text, level) {
00068     _time = time;
00069     _entity = entity;
00070     _component = component;
00071 }
```

8.149.3 Member Function Documentation

8.149.3.1 getComponent()

```
ModelComponent* TraceSimulationEvent::getComponent () const [inline]
```

Definition at line 73 of file [TraceManager.h](#).

```
00073             {
00074     return _component;
00075 }
```

8.149.3.2 getEntity()

```
Entity* TraceSimulationEvent::getEntity () const [inline]
```

Definition at line 77 of file [TraceManager.h](#).

```
00077             {
00078     return _entity;
00079 }
```

8.149.3.3 getTime() double TraceSimulationEvent::getTime () const [inline]

Definition at line 81 of file [TraceManager.h](#).

```
00081 {  
00082     return _time;  
00083 }
```

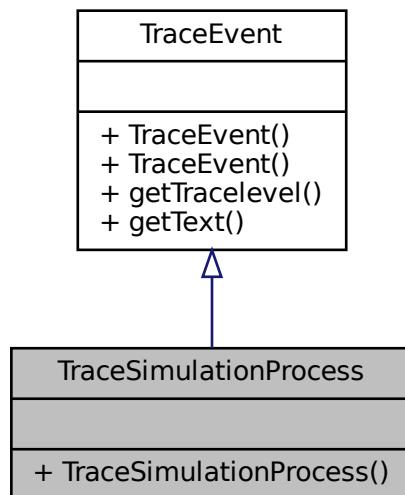
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

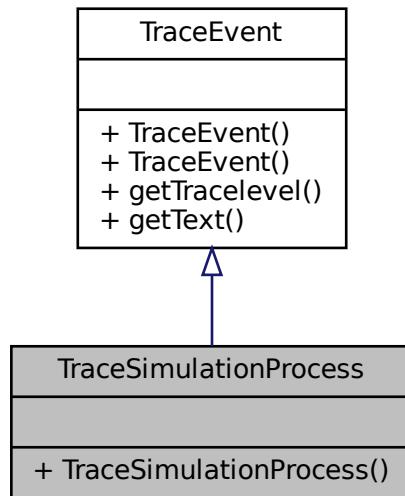
8.150 TraceSimulationProcess Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationProcess:



Collaboration diagram for TraceSimulationProcess:



Public Member Functions

- [TraceSimulationProcess \(std::string text, Util::TraceLevel level=Util::TraceLevel::modelSimulationEvent\)](#)

8.150.1 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 FULLY IMPLEMENTED (to be implemented for process analyser)

Definition at line [95](#) of file [TraceManager.h](#).

8.150.2 Constructor & Destructor Documentation

8.150.2.1 TraceSimulationProcess() [TraceSimulationProcess::TraceSimulationProcess \(](#)

```
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::modelSimulationEvent ) [inline]
```

Definition at line [98](#) of file [TraceManager.h](#).

```
00098   : TraceEvent(text, level) {
00099 }
```

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

- [TraceManager.h](#)

8.151 Traits< T > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< T >:



8.151.1 Detailed Description

```
template<typename T>
struct Traits< T >
```

Definition at line 83 of file [Traits.h](#).

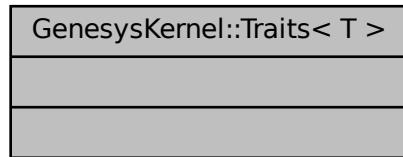
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.152 GenesysKernel::Traits< T > Struct Template Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< T >:



8.152.1 Detailed Description

```
template<typename T>
struct GenesysKernel::Traits< T >
```

Definition at line 43 of file [TraitsKernel.h](#).

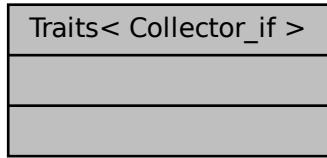
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.153 Traits< Collector_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Collector_if >:



Public Types

- [typedef CollectorDatafileDefaultImpl1 Implementation](#)

8.153.1 Detailed Description

Definition at line 149 of file [Traits.h](#).

8.153.2 Member Typedef Documentation

8.153.2.1 Implementation [typedef CollectorDatafileDefaultImpl1 Traits< Collector_if >::Implementation](#)

Definition at line 150 of file [Traits.h](#).

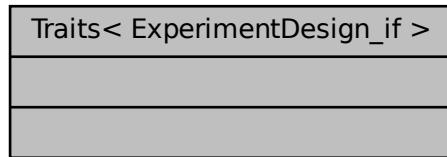
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.154 Traits< ExperimentDesign_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ExperimentDesign_if >:



Public Types

- [typedef ExperimentDesignDefaultImpl1 Implementation](#)

8.154.1 Detailed Description

Definition at line 187 of file [Traits.h](#).

8.154.2 Member Typedef Documentation

8.154.2.1 Implementation [typedef ExperimentDesignDefaultImpl1 Traits< ExperimentDesign_if >::Implementation](#)

Definition at line 188 of file [Traits.h](#).

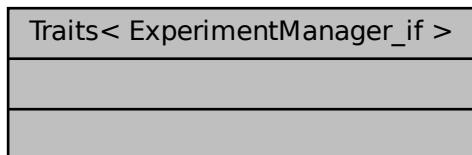
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.155 Traits< ExperimentManager_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ExperimentManager_if >:



Public Types

- [typedef ExperimentManagerDefaultImpl1 Implementation](#)

8.155.1 Detailed Description

Definition at line 191 of file [Traits.h](#).

8.155.2 Member Typedef Documentation

8.155.2.1 Implementation [typedef ExperimentManagerDefaultImpl1 Traits< ExperimentManager_if >::Implementation](#)

Definition at line 192 of file [Traits.h](#).

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

- [Traits.h](#)

8.156 Traits< Fitter_if > Struct Reference

#include <Traits.h>

Collaboration diagram for Traits< Fitter_if >:



Public Types

- [typedef FitterDefaultImpl1 Implementation](#)

8.156.1 Detailed Description

Definition at line 174 of file [Traits.h](#).

8.156.2 Member Typedef Documentation

8.156.2.1 Implementation `typedef FitterDefaultImpl1 Traits< Fitter_if >::Implementation`

Definition at line 175 of file [Traits.h](#).

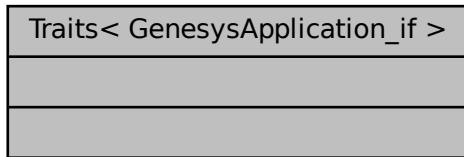
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.157 Traits< GenesysApplication_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< GenesysApplication_if >:



Public Types

- `typedef Modelo_SistemaOperacional02 Application`

8.157.1 Detailed Description

Definition at line 94 of file [Traits.h](#).

8.157.2 Member Typedef Documentation

8.157.2.1 Application `typedef Modelo_SistemaOperacional02 Traits< GenesysApplication_if >::Application`

Definition at line 98 of file [Traits.h](#).

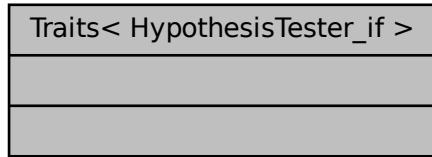
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.158 Traits< HypothesisTester_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< HypothesisTester_if >:



Public Types

- `typedef IntegratorDefaultImpl1 IntegratorImplementation`
- `typedef HypothesisTesterDefaultImpl1 Implementation`

8.158.1 Detailed Description

Definition at line 178 of file [Traits.h](#).

8.158.2 Member Typedef Documentation

8.158.2.1 Implementation `typedef HypothesisTesterDefaultImpl1 Traits< HypothesisTester_if >::Implementation`

Definition at line 180 of file [Traits.h](#).

8.158.2.2 IntegratorImplementation `typedef IntegratorDefaultImpl1 Traits< HypothesisTester_if >::IntegratorImplementation`

Definition at line 179 of file [Traits.h](#).

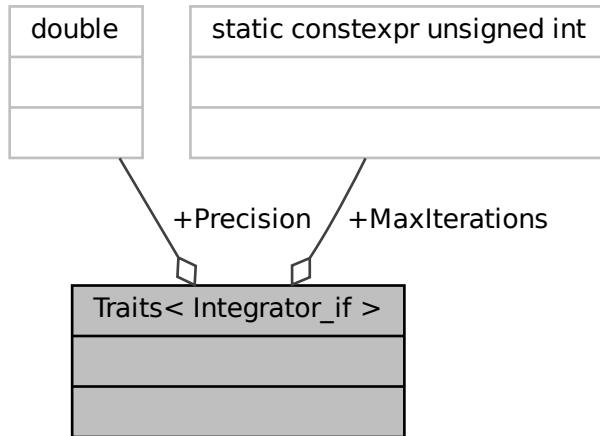
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.159 Traits< Integrator_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Integrator_if >:



Public Types

- `typedef IntegratorDefaultImpl1 Implementation`

Static Public Attributes

- `static constexpr unsigned int MaxIterations = 1e3`
- `static constexpr double Precision = 1e-9`

8.159.1 Detailed Description

Definition at line 159 of file [Traits.h](#).

8.159.2 Member Typedef Documentation

8.159.2.1 Implementation `typedef IntegratorDefaultImpl1 Traits< Integrator_if >::Implementation`

Definition at line 160 of file [Traits.h](#).

8.159.3 Member Data Documentation

8.159.3.1 MaxIterations `constexpr unsigned int Traits< Integrator_if >::MaxIterations = 1e3 [static], [constexpr]`

Definition at line 161 of file [Traits.h](#).

8.159.3.2 Precision `constexpr double Traits< Integrator_if >::Precision = 1e-9 [static], [constexpr]`

Definition at line 162 of file [Traits.h](#).

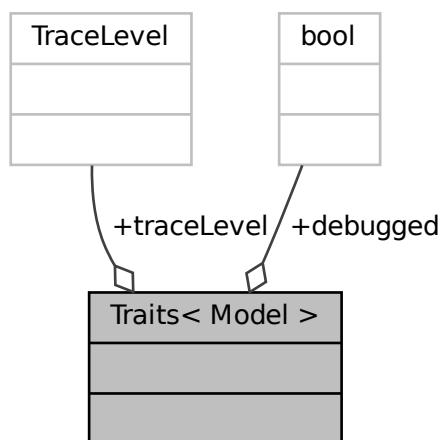
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.160 Traits< Model > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Model >:



Static Public Attributes

- static const bool `debugged = true`
- static const `Util::TraceLevel traceLevel = Util::TraceLevel::modelSimulationEvent`

8.160.1 Detailed Description

Definition at line 118 of file [Traits.h](#).

8.160.2 Member Data Documentation

8.160.2.1 **debugged** const bool [Traits< Model >](#)::debugged = true [static]

Definition at line 119 of file [Traits.h](#).

8.160.2.2 **traceLevel** const Util::TraceLevel [Traits< Model >](#)::traceLevel = Util::TraceLevel::modelSimulationEvent

[static]

Definition at line 120 of file [Traits.h](#).

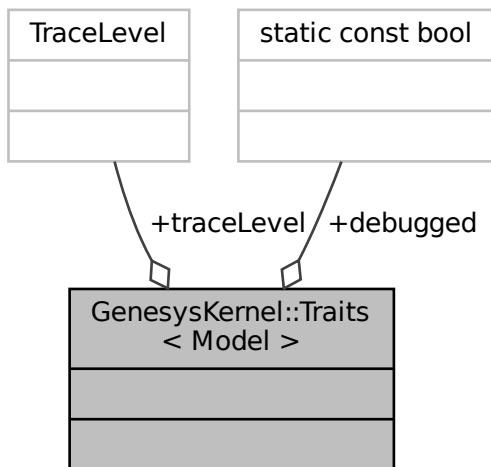
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.161 GenesysKernel::Traits< Model > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< Model >:



Static Public Attributes

- static const bool `debugged` = true
- static const `Util::TraceLevel traceLevel` = `Util::TraceLevel::modelSimulationEvent`

8.161.1 Detailed Description

Definition at line 50 of file [TraitsKernel.h](#).

8.161.2 Member Data Documentation

8.161.2.1 `debugged` const bool `GenesysKernel::Traits< Model >::debugged` = true [static]

Definition at line 51 of file [TraitsKernel.h](#).

8.161.2.2 `traceLevel` const `Util::TraceLevel GenesysKernel::Traits< Model >::traceLevel` = `Util::TraceLevel::mo` [static]

Definition at line 52 of file [TraitsKernel.h](#).

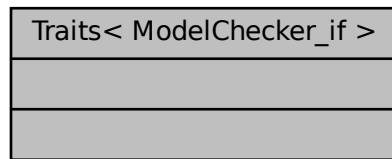
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.162 Traits< ModelChecker_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelChecker_if >:



Public Types

- [typedef ModelCheckerDefaultImpl1 Implementation](#)

8.162.1 Detailed Description

Definition at line 132 of file [Traits.h](#).

8.162.2 Member Typedef Documentation

8.162.2.1 Implementation [typedef ModelCheckerDefaultImpl1 Traits< ModelChecker_if >::Implementation](#)

Definition at line 133 of file [Traits.h](#).

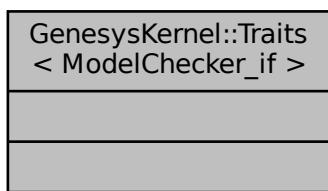
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.163 GenesysKernel::Traits< ModelChecker_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< ModelChecker_if >:



Public Types

- [typedef ModelCheckerDefaultImpl1 Implementation](#)

8.163.1 Detailed Description

Definition at line 63 of file [TraitsKernel.h](#).

8.163.2 Member Typedef Documentation

8.163.2.1 Implementation `typedef ModelCheckerDefaultImpl1 GenesysKernel::Traits< ModelChecker_if >::Implementation`

Definition at line 64 of file [TraitsKernel.h](#).

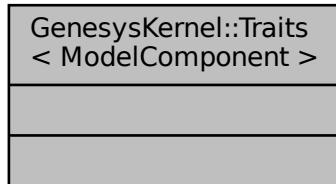
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.164 GenesysKernel::Traits< ModelComponent > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< ModelComponent >:



Public Types

- `typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation`
- `typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation`

8.164.1 Detailed Description

Definition at line 67 of file [TraitsKernel.h](#).

8.164.2 Member Typedef Documentation

8.164.2.1 StatisticsCollector_CollectorImplementation `typedef CollectorDefaultImpl1 GenesysKernel::Traits< ModelComponent >::StatisticsCollector_CollectorImplementation`

Definition at line 69 of file [TraitsKernel.h](#).

8.164.2.2 StatisticsCollector_StatisticsImplementation `typedef StatisticsDefaultImpl1 GenesysKernel::Traits< ModelComponent >::StatisticsCollector_StatisticsImplementation`

Definition at line 68 of file [TraitsKernel.h](#).

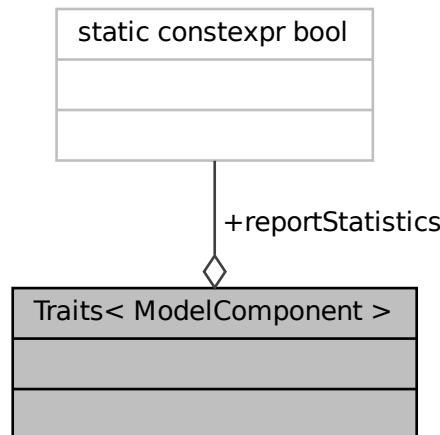
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.165 Traits< ModelComponent > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelComponent >:



Public Types

- `typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation`
- `typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation`

Static Public Attributes

- `static constexpr bool reportStatistics = true`

8.165.1 Detailed Description

Definition at line 136 of file [Traits.h](#).

8.165.2 Member Typedef Documentation

8.165.2.1 StatisticsCollector_CollectorImplementation `typedef CollectorDefaultImpl1 Traits< ModelComponent >::StatisticsCollector_CollectorImplementation`

Definition at line 138 of file [Traits.h](#).

8.165.2.2 StatisticsCollector_StatisticsImplementation `typedef StatisticsDefaultImpl1 Traits< ModelComponent >::StatisticsCollector_StatisticsImplementation`

Definition at line 137 of file [Traits.h](#).

8.165.3 Member Data Documentation

8.165.3.1 reportStatistics `constexpr bool Traits< ModelComponent >::reportStatistics = true [static], [constexpr]`

Definition at line 139 of file [Traits.h](#).

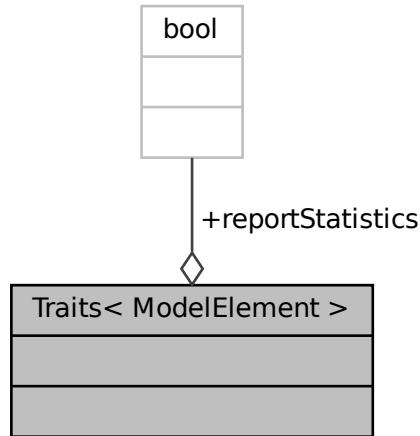
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.166 Traits< ModelElement > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelElement >:



Static Public Attributes

- static constexpr bool [reportStatistics](#) = true

8.166.1 Detailed Description

Definition at line 141 of file [Traits.h](#).

8.166.2 Member Data Documentation

8.166.2.1 reportStatistics constexpr bool [Traits< ModelElement >::reportStatistics](#) = true [static], [constexpr]

Definition at line 142 of file [Traits.h](#).

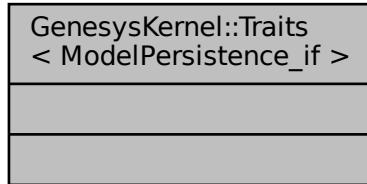
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.167 GenesysKernel::Traits< ModelPersistence_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< ModelPersistence_if >:



Public Types

- [typedef ModelPersistenceDefaultImpl1 Implementation](#)

8.167.1 Detailed Description

Definition at line [55](#) of file [TraitsKernel.h](#).

8.167.2 Member Typedef Documentation

8.167.2.1 Implementation [typedef ModelPersistenceDefaultImpl1 GenesysKernel::Traits< ModelPersistence_if >::Implementation](#)

Definition at line [56](#) of file [TraitsKernel.h](#).

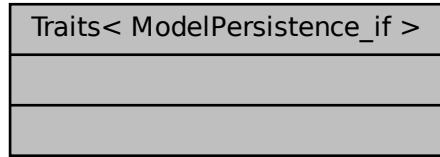
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.168 Traits< ModelPersistence_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelPersistence_if >:



Public Types

- [typedef ModelPersistenceDefaultImpl1 Implementation](#)

8.168.1 Detailed Description

Definition at line [123](#) of file [Traits.h](#).

8.168.2 Member Typedef Documentation

8.168.2.1 Implementation [typedef ModelPersistenceDefaultImpl1 Traits< ModelPersistence_if >::Implementation](#)

Definition at line [124](#) of file [Traits.h](#).

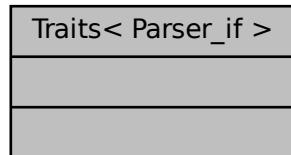
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.169 Traits< Parser_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Parser_if >:



Public Types

- `typedef ParserDefaultImpl2 Implementation`

8.169.1 Detailed Description

Definition at line 110 of file [Traits.h](#).

8.169.2 Member Typedef Documentation

8.169.2.1 Implementation `typedef ParserDefaultImpl2 Traits< Parser_if >::Implementation`

Definition at line 111 of file [Traits.h](#).

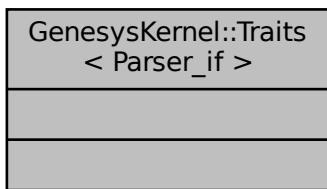
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.170 GenesysKernel::Traits< Parser_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< Parser_if >:



Public Types

- `typedef ParserDefaultImpl2 Implementation`

8.170.1 Detailed Description

Definition at line 77 of file [TraitsKernel.h](#).

8.170.2 Member Typedef Documentation

8.170.2.1 Implementation `typedef ParserDefaultImpl2 GenesysKernel::Traits< Parser_if >::Implementation`

Definition at line 78 of file [TraitsKernel.h](#).

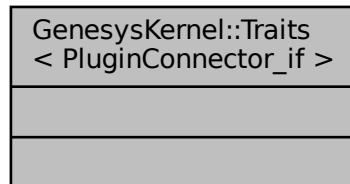
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.171 GenesysKernel::Traits< PluginConnector_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< PluginConnector_if >:



Public Types

- `typedef PluginConnectorDummyImpl1 Implementation`

8.171.1 Detailed Description

Definition at line 81 of file [TraitsKernel.h](#).

8.171.2 Member Typedef Documentation

8.171.2.1 Implementation `typedef PluginConnectorDummyImpl1 GenesysKernel::Traits< PluginConnector_if >::Implementation`

Definition at line 82 of file [TraitsKernel.h](#).

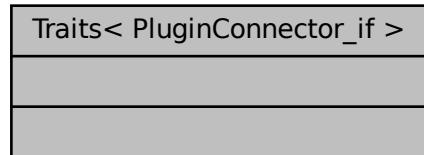
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.172 Traits< PluginConnector_if > Struct Reference

#include <Traits.h>

Collaboration diagram for Traits< PluginConnector_if >:



Public Types

- `typedef PluginConnectorDummyImpl1 Implementation`

8.172.1 Detailed Description

Definition at line 106 of file [Traits.h](#).

8.172.2 Member Typedef Documentation

8.172.2.1 Implementation `typedef PluginConnectorDummyImpl1 Traits< PluginConnector_if >::Implementation`

Definition at line 107 of file [Traits.h](#).

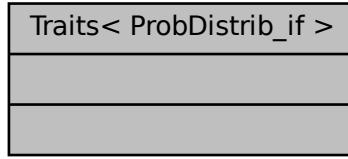
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.173 Traits< ProbDistrib_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ProbDistrib_if >:



Public Types

- [typedef ProbDistribDefaultImpl1 Implementation](#)

8.173.1 Detailed Description

Definition at line [170](#) of file [Traits.h](#).

8.173.2 Member Typedef Documentation

8.173.2.1 Implementation [typedef ProbDistribDefaultImpl1 Traits< ProbDistrib_if >::Implementation](#)

Definition at line [171](#) of file [Traits.h](#).

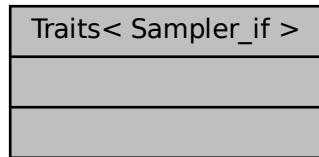
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.174 Traits< Sampler_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Sampler_if >:



Public Types

- `typedef SamplerDefaultImpl1 Implementation`
- `typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters`

8.174.1 Detailed Description

Definition at line 165 of file [Traits.h](#).

8.174.2 Member Typedef Documentation

8.174.2.1 Implementation `typedef SamplerDefaultImpl1 Traits< Sampler_if >::Implementation`

Definition at line 166 of file [Traits.h](#).

8.174.2.2 Parameters `typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Traits< Sampler_if >::Parameters`

Definition at line 167 of file [Traits.h](#).

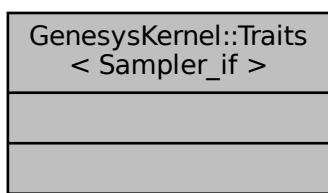
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.175 GenesysKernel::Traits< Sampler_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< Sampler_if >:



Public Types

- `typedef SamplerDefaultImpl1 Implementation`
- `typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters`

8.175.1 Detailed Description

Definition at line 72 of file [TraitsKernel.h](#).

8.175.2 Member Typedef Documentation

8.175.2.1 Implementation `typedef SamplerDefaultImpl1 GenesysKernel::Traits< Sampler_if >::Implementation`

Definition at line 73 of file [TraitsKernel.h](#).

8.175.2.2 Parameters `typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters GenesysKernel::Traits< Sampler_if >::Parameters`

Definition at line 74 of file [TraitsKernel.h](#).

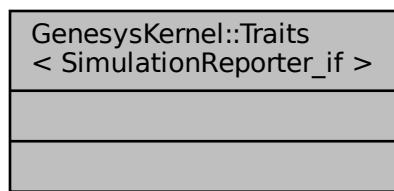
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.176 GenesysKernel::Traits< SimulationReporter_if > Struct Reference

```
#include <TraitsKernel.h>
```

Collaboration diagram for GenesysKernel::Traits< SimulationReporter_if >:



Public Types

- [typedef SimulationReporterDefaultImpl1 Implementation](#)

8.176.1 Detailed Description

Definition at line [59](#) of file [TraitsKernel.h](#).

8.176.2 Member Typedef Documentation

8.176.2.1 Implementation [typedef SimulationReporterDefaultImpl1 GenesysKernel::Traits< SimulationReporter_if >::Implementation](#)

Definition at line [60](#) of file [TraitsKernel.h](#).

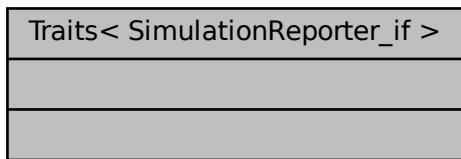
The documentation for this struct was generated from the following file:

- [TraitsKernel.h](#)

8.177 Traits< SimulationReporter_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< SimulationReporter_if >:



Public Types

- [typedef SimulationReporterDefaultImpl1 Implementation](#)
- [typedef Counter CounterImplementation](#)

8.177.1 Detailed Description

Definition at line [127](#) of file [Traits.h](#).

8.177.2 Member Typedef Documentation

8.177.2.1 CounterImplementation `typedef Counter Traits< SimulationReporter_if >::CounterImplementation`

Definition at line 129 of file [Traits.h](#).

8.177.2.2 Implementation `typedef SimulationReporterDefaultImpl1 Traits< SimulationReporter_if >::Implementation`

Definition at line 128 of file [Traits.h](#).

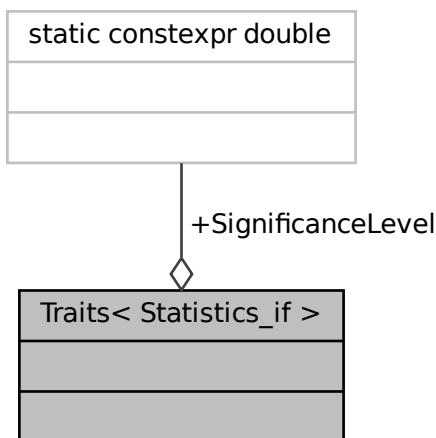
The documentation for this struct was generated from the following file:

- [Traits.h](#)

8.178 Traits< Statistics_if > Struct Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Statistics_if >:



Public Types

- `typedef StatisticsDefaultImpl1 Implementation`
- `typedef CollectorDefaultImpl1 CollectorImplementation`

Static Public Attributes

- static constexpr double `SignificanceLevel` = 0.05

8.178.1 Detailed Description

Definition at line 153 of file [Traits.h](#).

8.178.2 Member Typedef Documentation

8.178.2.1 CollectorImplementation `typedef CollectorDefaultImpl1 Traits< Statistics_if >::CollectorImplementation`

Definition at line 155 of file [Traits.h](#).

8.178.2.2 Implementation `typedef StatisticsDefaultImpl1 Traits< Statistics_if >::Implementation`

Definition at line 154 of file [Traits.h](#).

8.178.3 Member Data Documentation

8.178.3.1 SignificanceLevel `constexpr double Traits< Statistics_if >::SignificanceLevel = 0.05`
[static], [constexpr]

Definition at line 156 of file [Traits.h](#).

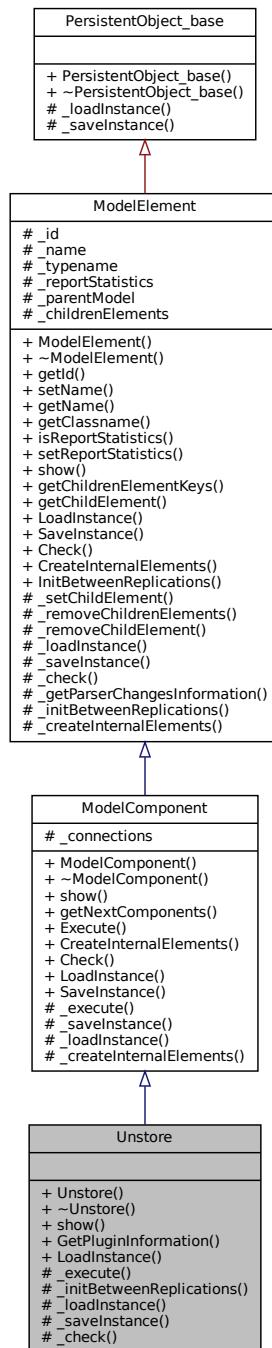
The documentation for this struct was generated from the following file:

- [Traits.h](#)

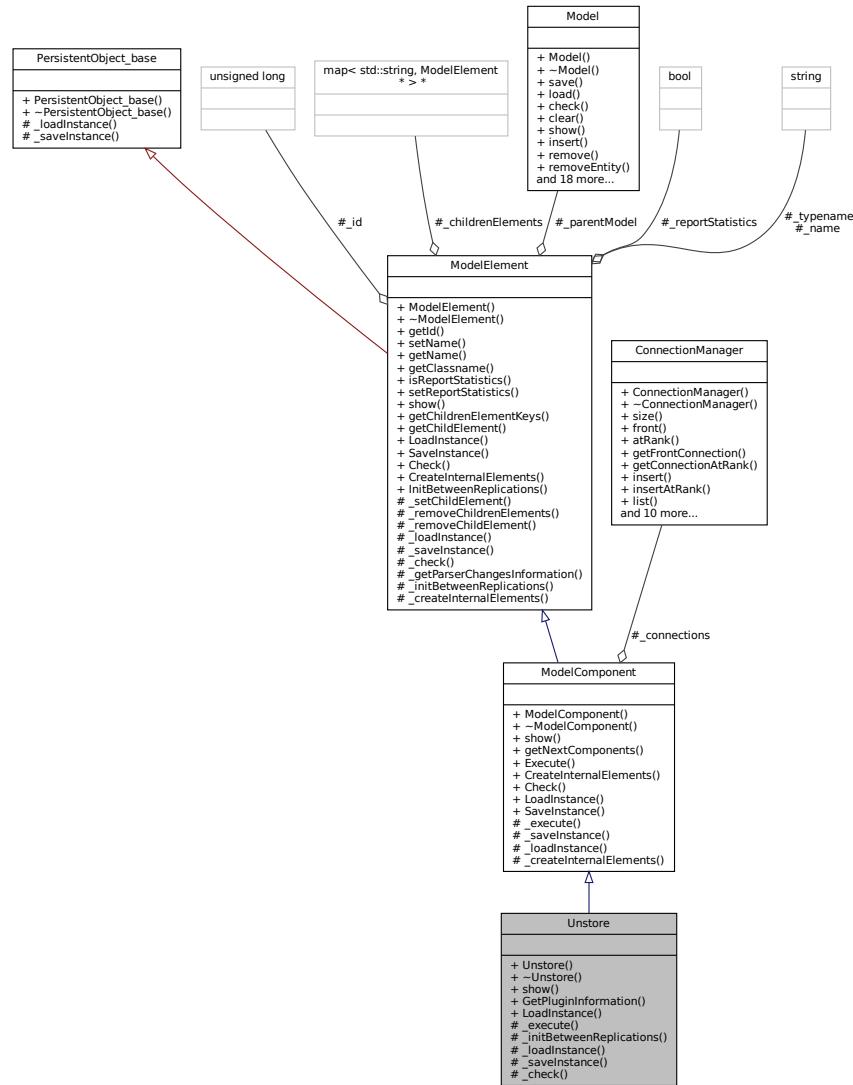
8.179 Unstore Class Reference

```
#include <Unstore.h>
```

Inheritance diagram for Unstore:



Collaboration diagram for Unstore:



Public Member Functions

- `Unstore (Model *model, std::string name="")`
- virtual `~Unstore ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.179.1 Detailed Description

Unstore module DESCRIPTION The [Unstore](#) module removes an entity from storage. When an entity arrives at the [Unstore](#) module, the storage specified is decreased and the entity immediately moves to the next module in the model. TYPICAL USES Removing the entity from an animation location when processing is complete Tracking the number of customers within a grocery store (unstore upon exit) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of specifying the storage name as a [Storage](#), [Set](#), [Attribute](#), or [Expression](#). Default will remove an entity from the last storage that it entered. [Storage](#) Name Name of the storage to which the entity will be added. Applies only when the Type is [Storage](#). [Set](#) Name Name of the storage set from which the storage is to be selected. Applies only when the Type is [Set](#). [Set](#) Index Index into the defined storage set that contains the desired storage name. Applies only when the Type is [Set](#). [Attribute](#) Name Name of the attribute whose value contains the storage. Applies only when the Type is [Attribute](#). [Expression](#) Expression that is evaluated to the storage into which the entity is placed. Applies only when the Type is [Expression](#).

Definition at line 45 of file [Unstore.h](#).

8.179.2 Constructor & Destructor Documentation

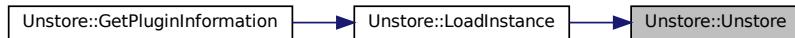
```
8.179.2.1 Unstore() Unstore::Unstore (
    Model * model,
    std::string name = "")
```

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

```
00017 : ModelComponent(model, Util::TypeOf<Unstore>(), name) {  
00018 }
```

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

Here is the caller graph for this function:



```
8.179.2.2 ~Unstore() virtual Unstore::~Unstore ( ) [virtual], [default]
```

8.179.3 Member Function Documentation

8.179.3.1 `_check()` `bool Unstore::_check (std::string * errorMessage) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 56 of file [Unstore.cpp](#).

```
00056 {  
00057     bool resultAll = true;  
00058     //...  
00059     return resultAll;  
00060 }
```

8.179.3.2 `_execute()` `void Unstore::_execute (Entity * entity) [protected], [virtual]`

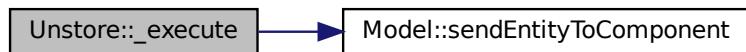
Implements [ModelComponent](#).

Definition at line 34 of file [Unstore.cpp](#).

```
00034 {  
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");  
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);  
00037 }
```

References [ModelElement::_parentModel](#), and [Model::sendEntityToComponent\(\)](#).

Here is the call graph for this function:



8.179.3.3 `_initBetweenReplications()` `void Unstore::_initBetweenReplications () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Unstore.cpp](#).

```
00047 {  
00048 }
```

```
8.179.3.4 _loadInstance() bool Unstore::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

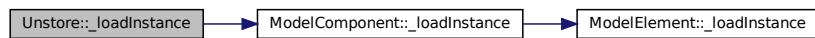
Definition at line 39 of file [Unstore.cpp](#).

```
00039
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
```

References [ModelComponent::_loadInstance\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.179.3.5 _saveInstance() std::map< std::string, std::string > * Unstore::_saveInstance ( )
[protected], [virtual]
```

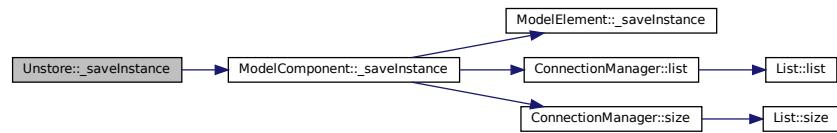
Reimplemented from [ModelComponent](#).

Definition at line 50 of file [Unstore.cpp](#).

```
00050
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
```

References [ModelComponent::_saveInstance\(\)](#).

Here is the call graph for this function:



8.179.3.6 GetPluginInformation() `PluginInformation * Unstore::GetPluginInformation () [static]`

Definition at line 62 of file [Unstore.cpp](#).

```
00062 {  
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Unstore>(), &Unstore::LoadInstance);  
00064 // ...  
00065     return info;  
00066 }
```

References [LoadInstance\(\)](#).

Here is the call graph for this function:



8.179.3.7 LoadInstance() `ModelComponent * Unstore::LoadInstance (`

```
    Model * model,  
    std::map< std::string, std::string > * fields ) [static]
```

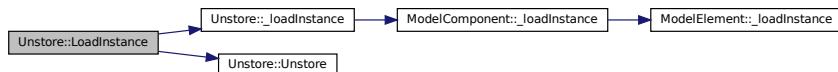
Definition at line 24 of file [Unstore.cpp](#).

```
00024 {  
00025     Unstore* newComponent = new Unstore(model);  
00026     try {  
00027         newComponent->_loadInstance(fields);  
00028     } catch (const std::exception& e) {  
00029     }  
00031     return newComponent;  
00032 }
```

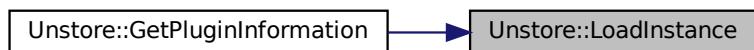
References [_loadInstance\(\)](#), and [Unstore\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.179.3.8 show() std::string Unstore::show () [virtual]

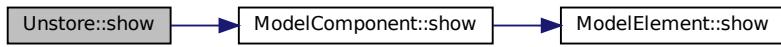
Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Unstore.cpp](#).

```
00020           {  
00021     return ModelComponent::show() + "";  
00022 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



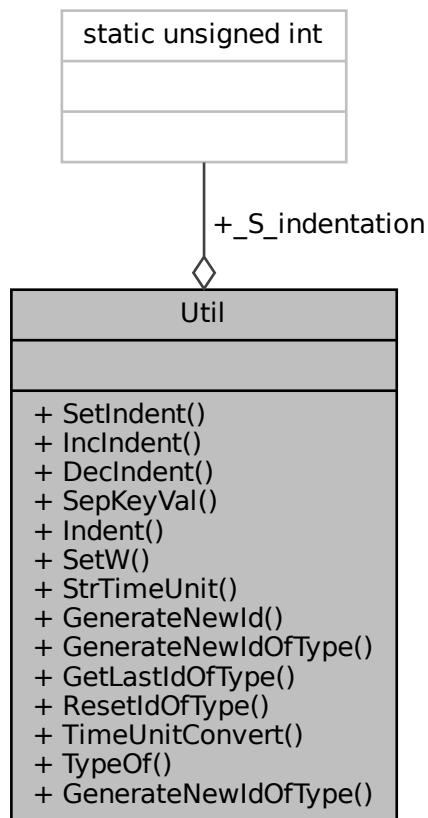
The documentation for this class was generated from the following files:

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

8.180 Util Class Reference

```
#include <Util.h>
```

Collaboration diagram for Util:



Public Types

- enum `TimeUnit` : int {

`TimeUnit::picosecond = 1, TimeUnit::nanosecond = 2, TimeUnit::microsecond = 3, TimeUnit::milisecond = 4,`
`TimeUnit::second = 5, TimeUnit::minute = 6, TimeUnit::hour = 7, TimeUnit::day = 8,`
`TimeUnit::week = 9 }`
- enum `TraceLevel` : int {

`TraceLevel::noTraces = 0, TraceLevel::errorFatal = 1, TraceLevel::errorRecover = 2, TraceLevel::warning = 3,`
`TraceLevel::report = 4, TraceLevel::simulatorResult = 10, TraceLevel::toolResult = 11, TraceLevel::modelResult = 12,`
`TraceLevel::componentResult = 13, TraceLevel::elementResult = 14, TraceLevel::modelSimulationEvent = 15,`
`TraceLevel::componentArrival = 20,`
`TraceLevel::simulatorInternal = 21, TraceLevel::toolInternal = 22, TraceLevel::modelSimulationInternal = 23,`
`TraceLevel::modelInternal = 24,`
`TraceLevel::componentInternal = 25, TraceLevel::elementInternal = 26, TraceLevel::simulatorDetailed = 30,`
`TraceLevel::toolDetailed = 31,`
`TraceLevel::modelSimulationDetailed = 32, TraceLevel::modelDetailed = 33, TraceLevel::componentDetailed = 34,`
`TraceLevel::elementDetailed = 35,`
`TraceLevel::everythingMostDetailed = 99 }`
- `typedef unsigned long identification`
- `typedef unsigned int rank`

Static Public Member Functions

- static void [SetIndent](#) (const unsigned short indent)
- static void [InIndent](#) ()
- static void [DeclIndent](#) ()
- static void [SepKeyVal](#) (std::string str, std::string *key, std::string *value)
- static std::string [Indent](#) ()
- static std::string [SetW](#) (std::string text, unsigned short width)
- static std::string [StrTimeUnit](#) ([Util::TimeUnit](#) timeUnit)
- static [Util::identification](#) [GenerateNewId](#) ()
- static [Util::identification](#) [GenerateNewIdOfType](#) (std::string objtype)
- static [Util::identification](#) [GetLastIdOfType](#) (std::string objtype)
- static void [ResetIdOfType](#) (std::string objtype)
- static double [TimeUnitConvert](#) ([Util::TimeUnit](#) timeUnit1, [Util::TimeUnit](#) timeUnit2)
- template<class T>
 static std::string [TypeOf](#) ()
- template<class T>
 static [Util::identification](#) [GenerateNewIdOfType](#) ()

Static Public Attributes

- static unsigned int [_S_indentation](#)

8.180.1 Detailed Description

Definition at line [80](#) of file [Util.h](#).

8.180.2 Member Typedef Documentation

8.180.2.1 [identification](#) `typedef unsigned long Util::identification`

Definition at line [82](#) of file [Util.h](#).

8.180.2.2 [rank](#) `typedef unsigned int Util::rank`

Definition at line [83](#) of file [Util.h](#).

8.180.3 Member Enumeration Documentation

8.180.3.1 [TimeUnit](#) `enum Util::TimeUnit : int [strong]`

Enumerator

picosecond	
nanosecond	
microsecond	
milisecond	
second	
minute	
hour	
day	
week	

Definition at line 85 of file [Util.h](#).

```

00085     : int {
00086     picosecond = 1,
00087     nanosecond = 2,
00088     microsecond = 3,
00089     milisecond = 4,
00090     second = 5,
00091     minute = 6,
00092     hour = 7,
00093     day = 8,
00094     week = 9
00095 };

```

8.180.3.2 TraceLevel enum [Util::TraceLevel](#) : int [strong]

Enumerator

noTraces	
errorFatal	
errorRecover	
warning	
report	
simulatorResult	
toolResult	
modelResult	
componentResult	
elementResult	
modelSimulationEvent	
componentArrival	
simulatorInternal	
toolInternal	
modelSimulationInternal	
modellInternal	
componentInternal	
elementInternal	
simulatorDetailed	
toolDetailed	
modelSimulationDetailed	
modelDetailed	
componentDetailed	
elementDetailed	
everythingMostDetailed	

Definition at line 98 of file [Util.h](#).

```
00098     : int {
00099     noTraces = 0,
00100     errorFatal = 1,
00101     errorRecover = 2,
00102     warning = 3,
00103     report = 4,
00104
00105     simulatorResult = 10,
00106     toolResult = 11,
00107     modelResult = 12,
00108     componentResult = 13,
00109     elementResult = 14,
00110
00111     modelSimulationEvent = 15,
00112
00113     componentArrival = 20,
00114     simulatorInternal = 21,
00115     toolInternal = 22,
00116     modelSimulationInternal = 23,
00117     modelInternal = 24,
00118     componentInternal = 25,
00119     elementInternal = 26,
00120
00121     simulatorDetailed = 30,
00122     toolDetailed = 31,
00123     modelSimulationDetailed = 32,
00124     modelDetailed = 33,
00125     componentDetailed = 34,
00126     elementDetailed = 35,
00127
00128     everythingMostDetailed = 99
00129 };
```

8.180.4 Member Function Documentation

8.180.4.1 DecIndent() void Util::DecIndent () [static]

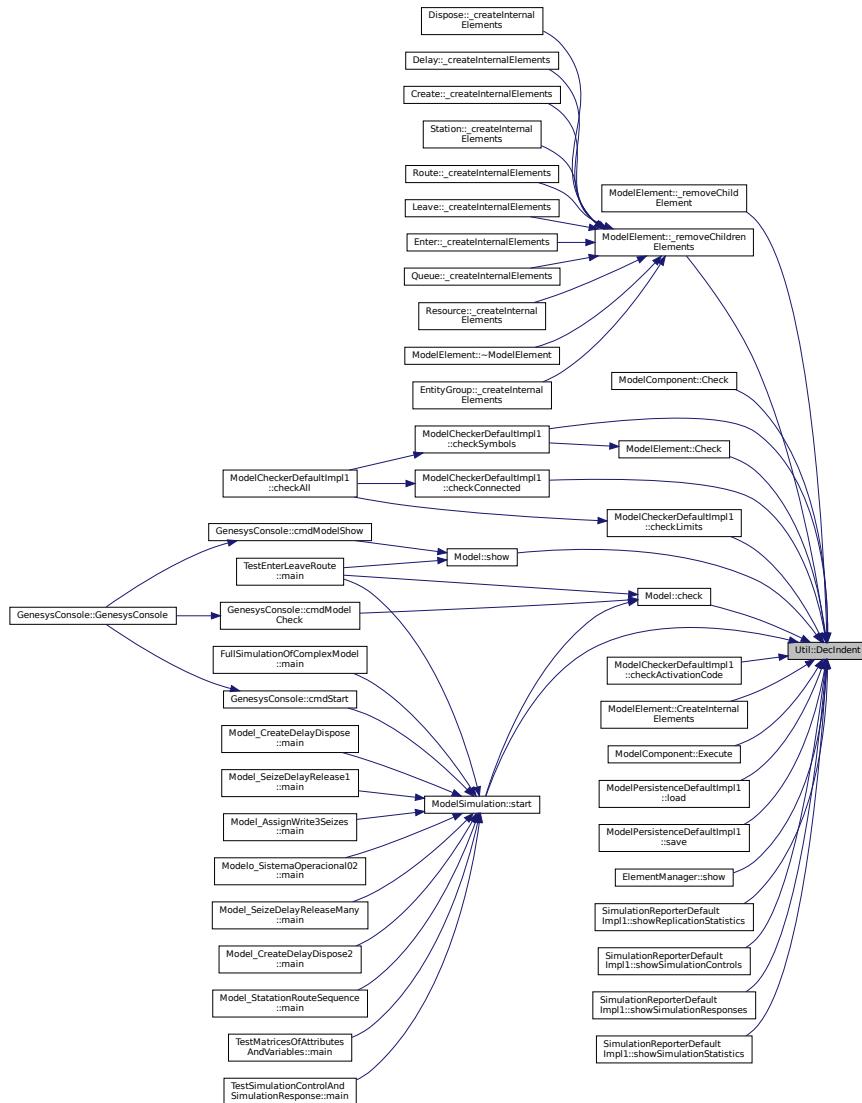
Definition at line 50 of file [Util.cpp](#).

```
00050     {
00051     Util::_S_indentation--;
00052 }
```

References [_S_indentation](#).

Referenced by [ModelElement::removeChildElement\(\)](#), [ModelElement::removeChildrenElements\(\)](#), [ModelComponent::Check\(\)](#), [ModelElement::Check\(\)](#), [Model::check\(\)](#), [ModelCheckerDefaultImpl1::checkActivationCode\(\)](#), [ModelCheckerDefaultImpl1::checkConn](#), [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [ModelElement::CreateInternalElements\(\)](#), [ModelComponent::Execute\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [Model::show\(\)](#), [ElementManager::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#), [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



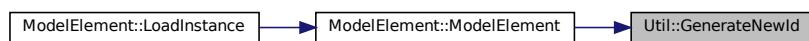
8.180.4.2 GenerateNewId() [Util::identification](#) Util::GenerateNewId () [static]

Definition at line 104 of file [Util.cpp](#).

```
00104
00105     Util::_S_lastId++;
00106     return Util::_S_lastId;
00107 }
```

Referenced by [ModelElement::ModelElement\(\)](#).

Here is the caller graph for this function:



8.180.4.3 GenerateNewIdOfType() [1/2] `template<class T >`
`static Util::identification Util::GenerateNewIdOfType () [inline], [static]`

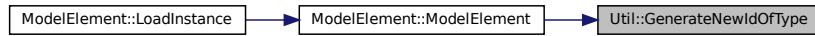
Every component or element has a unique ID for its class, but not unique for other classes. IDs are generated sequentially for each class.

Definition at line 176 of file [Util.h](#).

```
00176
00177     std::string objtype = Util::TypeOf<T>();
00178     std::map<std::string, Util::identification>::iterator it =
00179         Util::_S_lastIdOfType.find(objtype);
00180     if (it == Util::_S_lastIdOfType.end()) {
00181         // a new one. create the pair
00182         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00183         it = Util::_S_lastIdOfType.find(objtype);
00184     }
00185     Util::identification next = (*it).second;
00186     next++;
00187     (*it).second = next;
00188     return (*it).second;
00189 }
```

Referenced by [ModelElement::ModelElement\(\)](#).

Here is the caller graph for this function:



8.180.4.4 GenerateNewIdOfType() [2/2] `Util::identification Util::GenerateNewIdOfType (`
`std::string objtype) [static]`

Definition at line 109 of file [Util.cpp](#).

```
00109
00110     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00111     if (it == Util::_S_lastIdOfType.end()) {
00112         // a new one. create the pair
00113         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00114         it = Util::_S_lastIdOfType.find(objtype);
00115     }
00116     Util::identification next = (*it).second;
00117     next++;
00118     (*it).second = next;
00119     return (*it).second;
00120 }
```

8.180.4.5 GetLastIdOfType() `Util::identification Util::GetLastIdOfType (std::string objtype) [static]`

Definition at line 122 of file [Util.cpp](#).

```
00122                                         {
00123     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00124     if (it == Util::_S_lastIdOfType.end()) {
00125         // a new one. create the pair
00126         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00127         it = Util::_S_lastIdOfType.find(objtype);
00128     }
00129     //Util::identification next = (*it).second;
00130     //next++;
00131     //(*it).second = next;
00132     return (*it).second;
00133 }
00134 }
```

Referenced by [Entity::Entity\(\)](#).

Here is the caller graph for this function:



8.180.4.6 IncIndent() `void Util::IncIndent () [static]`

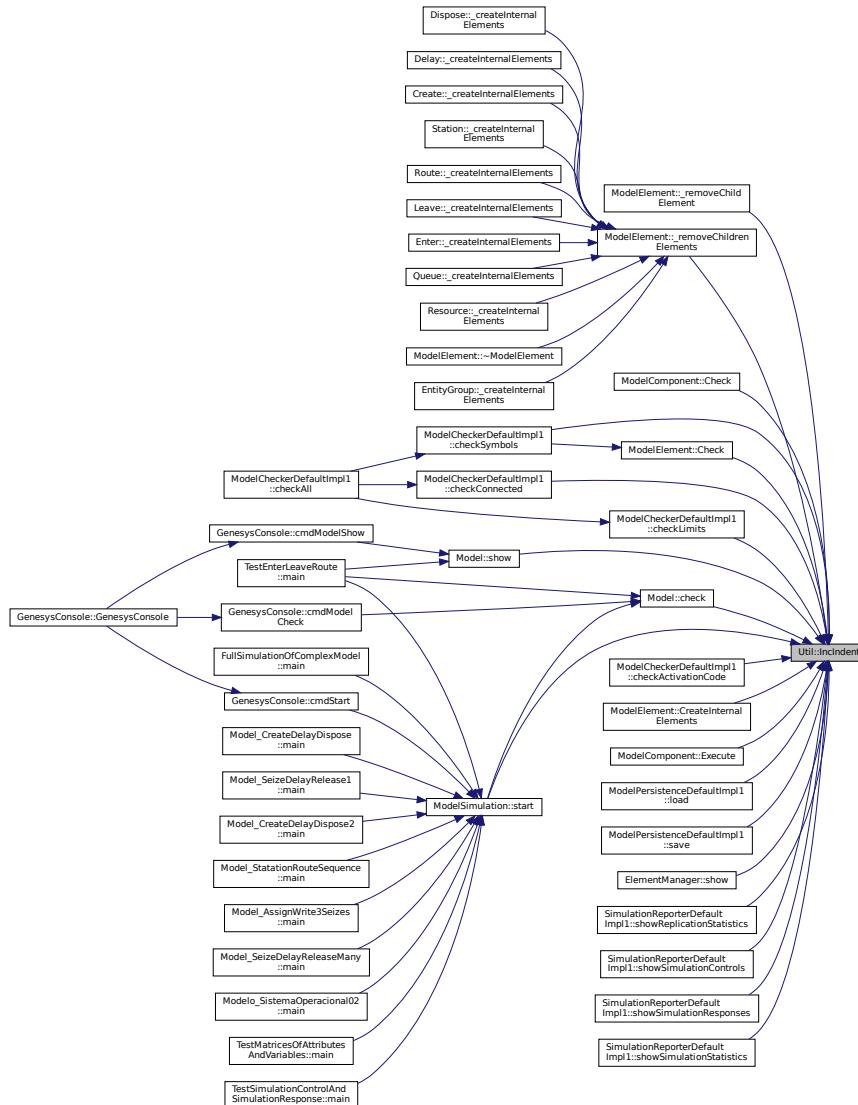
Definition at line 42 of file [Util.cpp](#).

```
00042             {
00043     Util::_S_indentation++;
00044 }
```

References [_S_indentation](#).

Referenced by [ModelElement::_removeChildElement\(\)](#), [ModelElement::_removeChildrenElements\(\)](#), [ModelComponent::Check\(\)](#), [ModelElement::Check\(\)](#), [Model::check\(\)](#), [ModelCheckerDefaultImpl1::checkActivationCode\(\)](#), [ModelCheckerDefaultImpl1::checkConn](#), [ModelCheckerDefaultImpl1::checkLimits\(\)](#), [ModelCheckerDefaultImpl1::checkSymbols\(\)](#), [ModelElement::CreateInternalElements\(\)](#), [ModelComponent::Execute\(\)](#), [ModelPersistenceDefaultImpl1::load\(\)](#), [ModelPersistenceDefaultImpl1::save\(\)](#), [Model::show\(\)](#), [ElementManager::show\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), [SimulationReporterDefaultImpl1::showStatistics\(\)](#), [SimulationReporterDefaultImpl1::showSimulationResponses\(\)](#), [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#), and [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



8.180.4.7 Indent() std::string Util::Indent() [static]

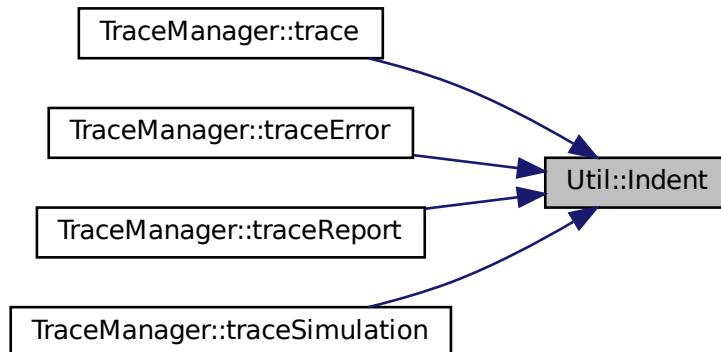
Definition at line 75 of file [Util.cpp](#).

```
00075     {
00076     std::string spaces = "";
00077     for (unsigned int i = 0; i < Util::_S_indentation; i++) {
00078         spaces += " | ";
00079     }
00080     return spaces;
00081 }
```

References [_S_indentation](#).

Referenced by [TraceManager::trace\(\)](#), [TraceManager::traceError\(\)](#), [TraceManager::traceReport\(\)](#), and [TraceManager::traceSimulation\(\)](#).

Here is the caller graph for this function:



8.180.4.8 ResetIdOfType() void Util::ResetIdOfType (std::string objtype) [static]

Definition at line 136 of file [Util.cpp](#).

```

00136     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00137     if (it == Util::_S_lastIdOfType.end()) {
00138         // a new one. create the pair
00139         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00140         it = Util::_S_lastIdOfType.find(objtype);
00141     }
00142     (*it).second = 0;
00143 }
00144 }
```

8.180.4.9 SepKeyVal() void Util::SepKeyVal (std::string str,
 std::string * key,
 std::string * value) [static]

Definition at line 54 of file [Util.cpp](#).

```

00054     // \todo: Check pointers when splitting string. There is an error
00055     //char *c;
00056     bool settingKey = true;
00057     //key = new std::string();
00058     //value = new std::string();
00059     key->clear();
00060     value->clear();
00061     for (std::string::iterator it = str.begin(); it != str.end(); it++) {
00062         if (settingKey) {
00063             if ((*it) != '=') {
00064                 key->append(new char((*it)));
00065             } else {
00066                 settingKey = false;
00067             }
00068         } else {
00069             value->append(new char((*it)));
00070         }
00071     }
00072 }
```

8.180.4.10 SetIndent() void Util::SetIndent (const unsigned short indent) [static]

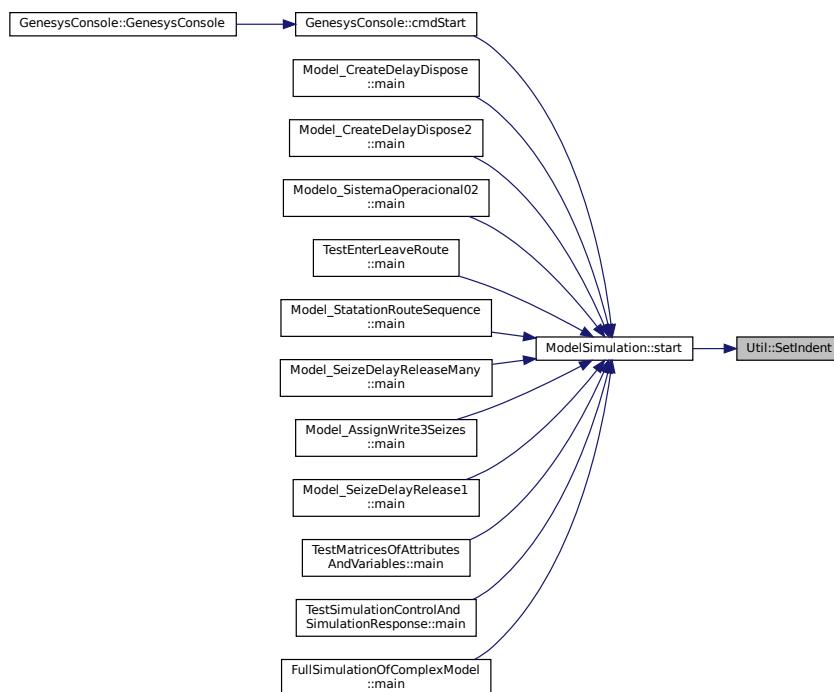
Definition at line 46 of file [Util.cpp](#).

```
00046
00047     Util::_S_indentation = indent;
00048 }
```

References [_S_indentation](#).

Referenced by [ModelSimulation::start\(\)](#).

Here is the caller graph for this function:



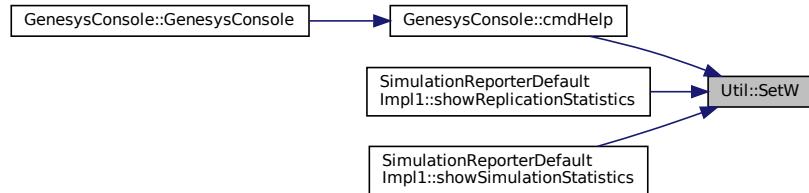
8.180.4.11 SetW() std::string Util::SetW (std::string text, unsigned short width) [static]

Definition at line 83 of file [Util.cpp](#).

```
00083
00084     std::string spaces(width, ' ');
00085     std::string result = text + spaces;
00086     return result.substr(0, width);
00087 }
```

Referenced by [GenesysConsole::cmdHelp\(\)](#), [SimulationReporterDefaultImpl1::showReplicationStatistics\(\)](#), and [SimulationReporterDefaultImpl1::showSimulationStatistics\(\)](#).

Here is the caller graph for this function:



8.180.4.12 StrTimeUnit()

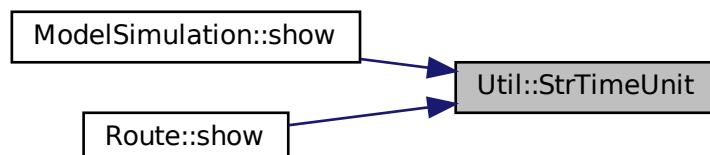
```
std::string Util::StrTimeUnit (
    Util::TimeUnit timeUnit ) [static]
```

Definition at line 89 of file [Util.cpp](#).

```
00089
00090     switch (static_cast<int> (timeUnit)) {
00091         case 1: return "picosecond";
00092         case 2: return "nanosecond";
00093         case 3: return "microsecond";
00094         case 4: return "millisecond";
00095         case 5: return "second";
00096         case 6: return "minute";
00097         case 7: return "hour";
00098         case 8: return "day";
00099         case 9: return "week";
00100     }
00101     return "";
00102 }
```

Referenced by [ModelSimulation::show\(\)](#), and [Route::show\(\)](#).

Here is the caller graph for this function:



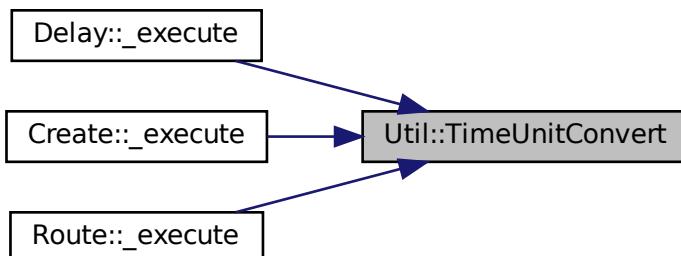
```
8.180.4.13 TimeUnitConvert() double Util::TimeUnitConvert (
    Util::TimeUnit timeUnit1,
    Util::TimeUnit timeUnit2 ) [static]
```

Definition at line 146 of file [Util.cpp](#).

```
00146
00147     double scaleValues[] = {1.0, 1000.0, 1000.0, 1000.0, 1000.0, 60.0, 60.0, 24.0, 7.0};
00148     // TU_picosecond = 1, TU_microsecond = 3, TU_milisecond = 4, TU_second = 5, TU_minute = 6, TU_hour
00149     = 7, TU_day = 8, TU_week = 9
00150     double res = 1.0;
00151     int intTimeUnit1, intTimeUnit2;
00152     intTimeUnit1 = static_cast<int> (timeUnit1);
00153     intTimeUnit2 = static_cast<int> (timeUnit2);
00154     if (intTimeUnit1 <= intTimeUnit2) {
00155         for (int i = intTimeUnit1 + 1; i <= intTimeUnit2; i++) {
00156             res /= scaleValues[i];
00157         }
00158     } else {
00159         for (int i = intTimeUnit2 + 1; i <= intTimeUnit1; i++) {
00160             res *= scaleValues[i];
00161         }
00162     }
00163 }
```

Referenced by [Delay::_execute\(\)](#), [Create::_execute\(\)](#), and [Route::_execute\(\)](#).

Here is the caller graph for this function:



```
8.180.4.14 TypeOf() template<class T >
static std::string Util::TypeOf ( ) [inline], [static]
```

Return the name of the class used as T.

Definition at line 158 of file [Util.h](#).

```
00158
00159     std::string name = typeid (T).name();
00160     std::map<std::string, std::string>::iterator it = _S_TypeOf.find(name);
00161     if (it != _S_TypeOf.end()) {
00162         return (*it).second;
00163     } else {
00164         std::string newname(name);
00165         while (std::isdigit(newname[0])) {
00166             newname.erase(0, 1);
00167         }
00168         _S_TypeOf.insert(std::pair<std::string, std::string>(name, newname));
00169         return newname;
00170     }
00171 }
```

8.180.5 Member Data Documentation

8.180.5.1 `_S_indentation` `unsigned int Util::_S_indentation [static]`

Definition at line 136 of file [Util.h](#).

Referenced by [DeclIndent\(\)](#), [InclIndent\(\)](#), [Indent\(\)](#), and [SetIndent\(\)](#).

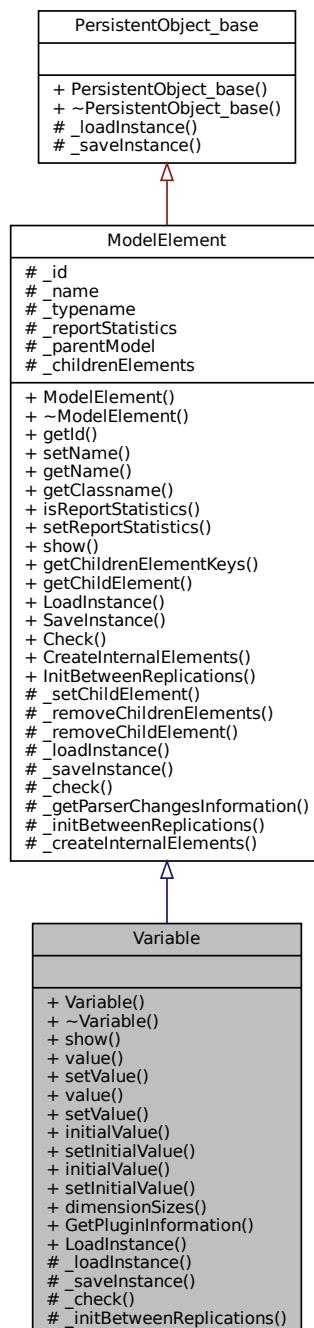
The documentation for this class was generated from the following files:

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

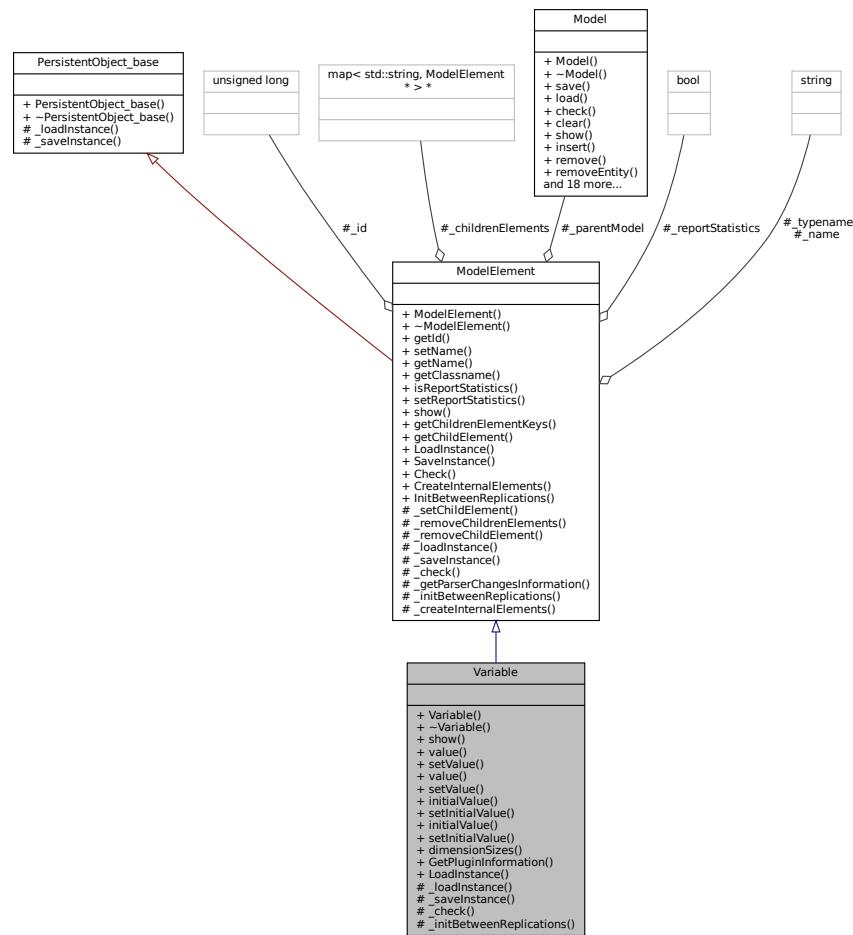
8.181 Variable Class Reference

```
#include <Variable.h>
```

Inheritance diagram for Variable:



Collaboration diagram for Variable:



Public Member Functions

- **Variable (Model *model, std::string name="")**
- virtual **~Variable ()=default**
- virtual std::string **show ()**
- double **value ()**
- void **setValue (double value)**
- double **value (std::string index)**
- void **setValue (std::string index, double value)**
- double **initialValue ()**
- void **setInitialValue (double value)**
- double **initialValue (std::string index)**
- void **setInitialValue (std::string index, double value)**
- **List< unsigned int > * dimensionSizes () const**

Static Public Member Functions

- static **PluginInformation * GetPluginInformation ()**
- static **ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**

Protected Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)
- virtual void `_initBetweenReplications` ()

Additional Inherited Members

8.181.1 Detailed Description

Variable module DESCRIPTION This data module is used to define a variable's dimension and values. You can reference variables in other modules (for example, the **Decide** module), reassign new values to variables with the **Assign** module, and use variables in any expression. You can use an external data file to specify variable values, and you can specify the variable's initial values in the **Variable** module. If you use both methods, the values are read at different times, depending on the options you specify, including the **File** Read Time, the Clear Option, and the replication parameters you specify in the Run Setup dialog box. For more information, see the online Help. There are three methods for manually editing the Initial Values of a **Variable** module: Using the standard spreadsheet interface. In the module spreadsheet, right-click on the Initial Values cell and select the Edit via spreadsheet menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the module dialog box. In the module spreadsheet, right-click on any cell and select the Edit via dialog menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the two-dimensional (2-D) spreadsheet interface. In the module spreadsheet, click on the Initial Values cell. TYPICAL USES Number of documents processed per hour Serial number to assign to parts for unique identification Space available in a facility PROMPTS Prompt Description Name The unique name of the variable being defined. Rows Number of rows in a one- or two-dimensional variable. Columns Number of columns in a two-dimensional variable. Report Statistics Check box for determining whether or not statistics will be collected. This field is visible when the rows and columns are not specified (that is, for single variables). Data Type The data type of the values stored in the variable. Valid types are Real and String. The default type is Real. Clear Option Defines the time (if at all) when the value(s) of the variable is reset to the initial value(s) specified. Specifying Statistics resets this variable to its initial value(s) whenever statistics are cleared. Specifying System resets this variable to its initial value(s) whenever the system is cleared. Specifying None indicates that this variable is never reset to its initial value(s), except prior to the first replication. File Name Name of the file from which to read the variable's value or values. You can use any file access type supported by Arena except sequential text files and Lotus spreadsheet (.wks) files. If the file name you specify has not been created yet, Arena will create it, but you must edit the file to specify the file access type, path, and recordset (if required). Recordset Name of the recordset in the specified file from which to read values. This field is available only if you specify a **File** Name for a file that has been set up with a file access type, path, and recordset. Arena uses the Rows and Columns properties to determine the amount of data to read from the recordset. A recordset is required for all file types except .xml. The recordset size must be equal to or greater than the number of rows and columns specified for the variable. File Read Time Specifies when to read the values from the file into the variable. If you select PreCheck, the values for the variable are read while the model is still in Edit mode (prior to the model being checked and compiled). If you select BeginSimulation, values are read when the model is compiled, prior to the first replication. If you select BeginReplication, values are read prior to each replication. Initial Values Lists the initial value or values of the variable. You can assign new values to the variable at different stages of the model by using the **Assign** module. Initial Value **Variable** value at the start of the simulation.

Definition at line 90 of file [Variable.h](#).

8.181.2 Constructor & Destructor Documentation

```
8.181.2.1 Variable() Variable::Variable (
    Model * model,
    std::string name = "" )
```

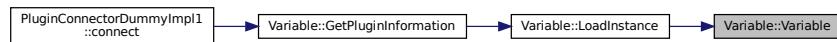
Definition at line 17 of file [Variable.cpp](#).

```
00017 : ModelElement(model, Util::TypeOf<Variable>(), name) {
00018     _name = name;
00019 }
```

References [ModelElement::_name](#).

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

Here is the caller graph for this function:



```
8.181.2.2 ~Variable() virtual Variable::~Variable ( ) [virtual], [default]
```

8.181.3 Member Function Documentation

```
8.181.3.1 _check() bool Variable::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 129 of file [Variable.cpp](#).

```
00129 {
00130     return true;
00131 }
```

```
8.181.3.2 _initBetweenReplications() void Variable::_initBetweenReplications ( ) [protected],
[virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 133 of file [Variable.cpp](#).

```
00133 {
00134     this->_values->clear();
00135     this->_values = this->_initialValues;
00136 }
```

```
8.181.3.3 _loadInstance() bool Variable::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

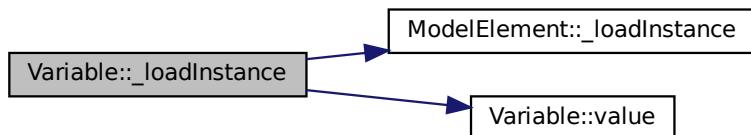
Definition at line 98 of file [Variable.cpp](#).

```
00098
00099     bool res = ModelElement::_loadInstance(fields);
00100     if (res) {
00103         unsigned int nv = std::stoi((*(fields->find("numValues"))).second);
00104         std::string pos;
00105         double value;
00106         for (unsigned int i = 0; i < nv; i++) {
00107             pos = ((*(fields->find("pos" + std::to_string(i)))).second);
00108             value = std::stod((*(fields->find("value" + std::to_string(i)))).second);
00109             this->_initialValues->emplace(pos, value);
00110         }
00111     }
00112     return res;
00113 }
```

References [ModelElement::_loadInstance\(\)](#), and [value\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.181.3.4 _saveInstance() std::map< std::string, std::string > * Variable::_saveInstance ( )
[protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 115 of file [Variable.cpp](#).

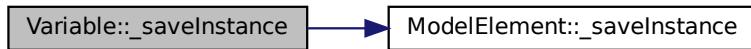
```
00115
00116     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00117     //Util::TypeOf<Variable>());
00119     fields->emplace("numValues", std::to_string(this->_initialValues->size()));
00120     unsigned int i = 0;
00121     for (std::map<std::string, double>::iterator it = this->_initialValues->begin(); it != _initialValues->end(); it++) {
00122         fields->emplace("pos" + std::to_string(i), (*it).first);
```

```

00123     fields->emplace("value" + std::to_string(i), std::to_string((*it).second));
00124     i++;
00125 }
00126 return fields;
00127 }
```

References [ModelElement::_saveInstance\(\)](#).

Here is the call graph for this function:



8.181.3.5 dimensionSizes() `List< unsigned int > * Variable::dimensionSizes () const`

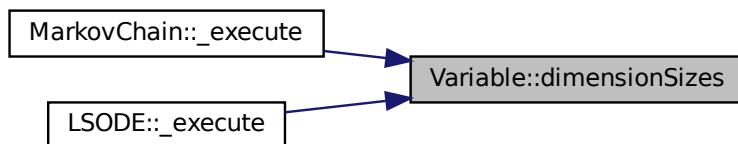
Definition at line 84 of file [Variable.cpp](#).

```

00084 {
00085     return _dimensionSizes;
00086 }
```

Referenced by [MarkovChain::_execute\(\)](#), and [LSODE::_execute\(\)](#).

Here is the caller graph for this function:



8.181.3.6 GetPluginInformation() `PluginInformation * Variable::GetPluginInformation () [static]`

Definition at line 25 of file [Variable.cpp](#).

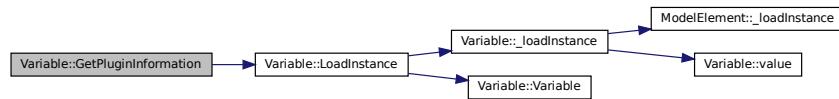
```

00025 {
00026     PluginInformation* info = new PluginInformation(Util::TypeOf<Variable>(),
00027         &Variable::LoadInstance);
00028     return info;
00029 }
```

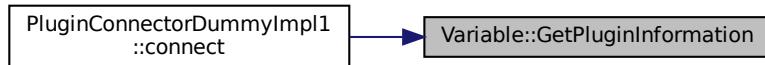
References [LoadInstance\(\)](#).

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.181.3.7 initialValue() [1/2] double Variable::initialValue ()

Definition at line 57 of file [Variable.cpp](#).

```
00057
00058     return initialValue("");
00059 }
```

8.181.3.8 initialValue() [2/2] double Variable::initialValue (std::string index)

Definition at line 65 of file [Variable.cpp](#).

```
00065
00066     std::map<std::string, double>::iterator it = _initialValues->find(index);
00067     if (it == _initialValues->end()) {
00068         return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00069     } else {
00070         return it->second;
00071     }
00072 }
```

8.181.3.9 LoadInstance() ModelElement * Variable::LoadInstance (Model * model, std::map< std::string, std::string > * fields) [static]

Definition at line 88 of file [Variable.cpp](#).

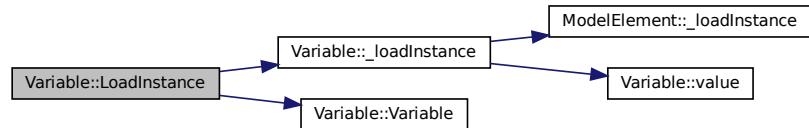
```
00088
00089     Variable* newElement = new Variable(model);
00090     try {
00091         newElement->_loadInstance(fields);
00092     } catch (const std::exception& e) {
00093
00094 }
```

```
00095     return newElement;
00096 }
```

References [_loadInstance\(\)](#), and [Variable\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.181.3.10 setInitialValue() [1/2] void Variable::setInitialValue (double value)

Definition at line 61 of file [Variable.cpp](#).

```
00061
00062     setInitialValue("", value);
00063 }
```

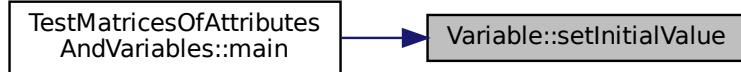
References [value\(\)](#).

Referenced by [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.181.3.11 setInitialValue() [2/2] void Variable::setInitialValue (
 std::string index,
 double value)

Definition at line 74 of file [Variable.cpp](#).

```
00074     std::map<std::string, double>::iterator it = _initialValues->find(index);  
00075     {  
00076         if (it == _initialValues->end()) {  
00077             // index does not exist. Create it.  
00078             _initialValues->insert(std::pair<std::string, double>(index, value));  
00079         } else {  
00080             it->second = value;  
00081         }  
00082     }
```

References [value\(\)](#).

Here is the call graph for this function:



8.181.3.12 setValue() [1/2] void Variable::setValue (
 double value)

Definition at line 43 of file [Variable.cpp](#).

```
00043     {  
00044         setValue("", value);  
00045     }
```

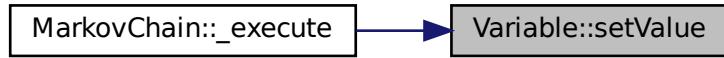
References [value\(\)](#).

Referenced by [MarkovChain::_execute\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.181.3.13 setValue() [2/2] void Variable::setValue (std::string index, double value)

Definition at line 47 of file [Variable.cpp](#).

```
00047     std::map<std::string, double>::iterator it = _values->find(index);  
00048     if (it == _values->end()) {  
00049         // index does not exist. Create it.  
00050         _values->insert({index, value}); // (std::pair<std::string, double>(index, value));  
00051     } else {  
00052         it->second = value;  
00053     }  
00054 }
```

References [value\(\)](#).

Here is the call graph for this function:



8.181.3.14 show() std::string Variable::show () [virtual]

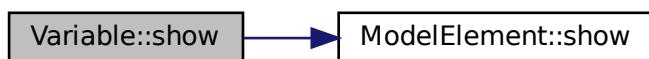
Reimplemented from [ModelElement](#).

Definition at line 21 of file [Variable.cpp](#).

```
00021     {
00022         return ModelElement::show(); // \todo: include values
00023     }
```

References [ModelElement::show\(\)](#).

Here is the call graph for this function:

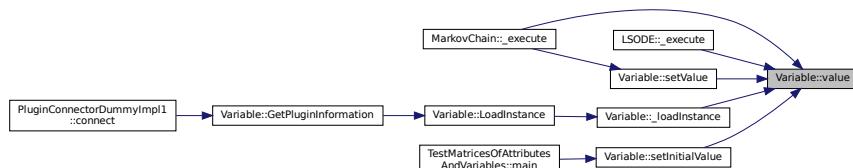
**8.181.3.15 value() [1/2]** double Variable::value ()

Definition at line 30 of file [Variable.cpp](#).

```
00030     {
00031         return value("");
00032     }
```

Referenced by [MarkovChain::_execute\(\)](#), [LSODE::_execute\(\)](#), [_loadInstance\(\)](#), [setInitialValue\(\)](#), and [setValue\(\)](#).

Here is the caller graph for this function:

**8.181.3.16 value() [2/2]** double Variable::value (std::string index)

Definition at line 34 of file [Variable.cpp](#).

```
00034     {
00035         std::map<std::string, double>::iterator it = _values->find(index);
00036         if (it == _values->end()) {
00037             return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00038         } else {
00039             return it->second;
00040         }
00041     }
```

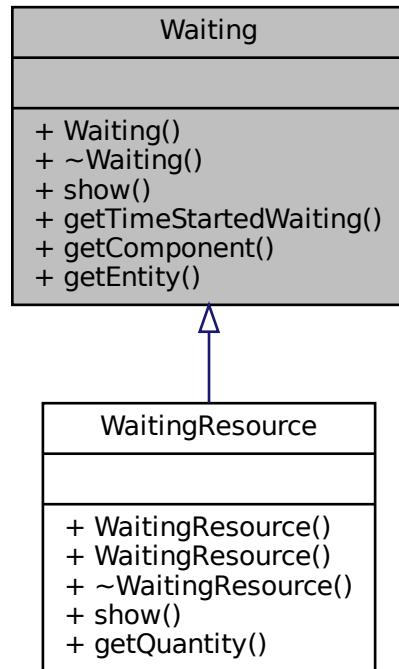
The documentation for this class was generated from the following files:

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

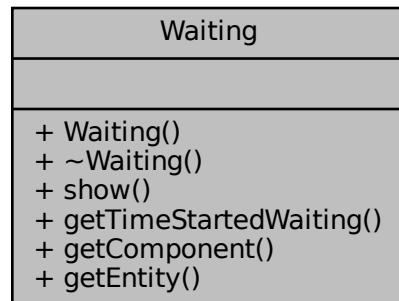
8.182 Waiting Class Reference

```
#include <Queue.h>
```

Inheritance diagram for Waiting:



Collaboration diagram for Waiting:



Public Member Functions

- `Waiting (Entity *entity, ModelComponent *component, double timeStartedWaiting)`
- `virtual ~Waiting ()=default`
- `virtual std::string show ()`
- `double getTimeStartedWaiting () const`
- `ModelComponent * getComponent () const`
- `Entity * getEntity () const`

8.182.1 Detailed Description

Definition at line 25 of file [Queue.h](#).

8.182.2 Constructor & Destructor Documentation

8.182.2.1 Waiting() `Waiting::Waiting (`
 `Entity * entity,`
 `ModelComponent * component,`
 `double timeStartedWaiting) [inline]`

Definition at line 28 of file [Queue.h](#).

```
00028
00029     _entity = entity;
00030     _component = component;
00031     _timeStartedWaiting = timeStartedWaiting;
00032 }
```

8.182.2.2 ~Waiting() `virtual Waiting::~Waiting () [virtual], [default]`

8.182.3 Member Function Documentation

8.182.3.1 getComponent() `ModelComponent* Waiting::getComponent () const [inline]`

Definition at line 49 of file [Queue.h](#).

```
00049
00050     return _component;
00051 }
```

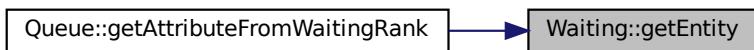
8.182.3.2 getEntity() `Entity* Waiting::getEntity () const [inline]`

Definition at line 53 of file [Queue.h](#).

```
00053             {
00054         return _entity;
00055     }
```

Referenced by [Queue::getAttributeFromWaitingRank\(\)](#).

Here is the caller graph for this function:

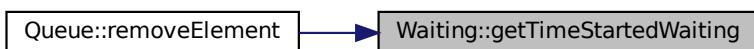
**8.182.3.3 getTimeStartedWaiting()** `double Waiting::getTimeStartedWaiting () const [inline]`

Definition at line 45 of file [Queue.h](#).

```
00045             {
00046         return _timeStartedWaiting;
00047     }
```

Referenced by [Queue::removeElement\(\)](#).

Here is the caller graph for this function:

**8.182.3.4 show()** `virtual std::string Waiting::show () [inline], [virtual]`

Reimplemented in [WaitingResource](#).

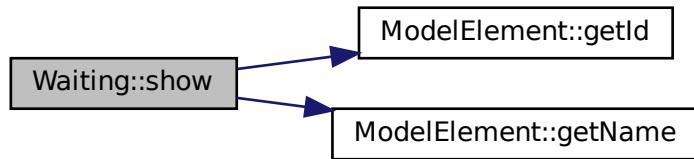
Definition at line 37 of file [Queue.h](#).

```
00037             {
00038         return //ModelElement::show() +
00039             ",entity=" + std::to_string(_entity->getId()) +
00040             ",component=\"" + _component->getName() + "\" +
00041             ",timeStatedWaiting=" + std::to_string(_timeStartedWaiting);
00042     }
```

References [ModelElement::getId\(\)](#), and [ModelElement::getName\(\)](#).

Referenced by [WaitingResource::show\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



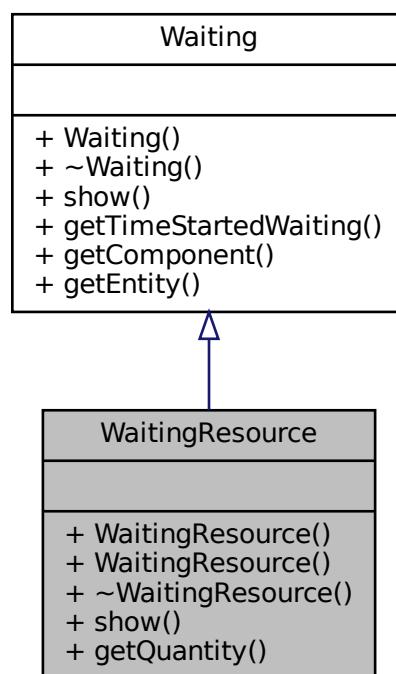
The documentation for this class was generated from the following file:

- [Queue.h](#)

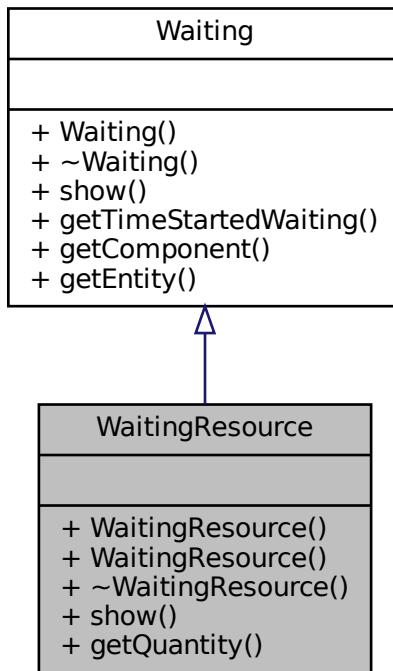
8.183 WaitingResource Class Reference

```
#include <Seize.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 ()=default`
- `virtual std::string show ()`
- `unsigned int getQuantity () const`

8.183.1 Detailed Description

Definition at line 25 of file [Seize.h](#).

8.183.2 Constructor & Destructor Documentation

8.183.2.1 WaitingResource() [1/2] WaitingResource::WaitingResource (

```
Entity * entity,
ModelComponent * component,
double timeStartedWaiting,
unsigned int quantity) [inline]
```

Definition at line 28 of file [Seize.h](#).

```
00028     : Waiting(entity, component, timeStartedWaiting) {
00029         _quantity = quantity;
00030 }
```

8.183.2.2 WaitingResource() [2/2] WaitingResource::WaitingResource (

```
const WaitingResource & orig) [inline]
```

Definition at line 32 of file [Seize.h](#).

```
00032     : Waiting(orig) {
00033 }
```

8.183.2.3 ~WaitingResource() virtual WaitingResource::~WaitingResource () [virtual], [default]

8.183.3 Member Function Documentation

8.183.3.1 getQuantity() unsigned int WaitingResource::getQuantity () const [inline]

Definition at line 44 of file [Seize.h](#).

```
00044 {
00045     return _quantity;
00046 }
```

8.183.3.2 show() virtual std::string WaitingResource::show () [inline], [virtual]

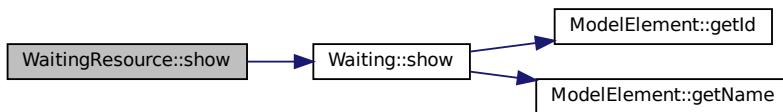
Reimplemented from [Waiting](#).

Definition at line 38 of file [Seize.h](#).

```
00038     {
00039         return Waiting::show() +
00040             ",quantity=" + std::to_string(this->_quantity);
00041 }
```

References [Waiting::show\(\)](#).

Here is the call graph for this function:



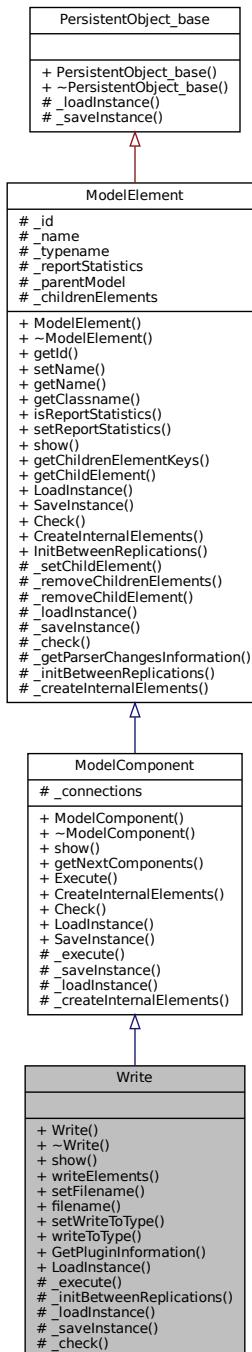
The documentation for this class was generated from the following file:

- [Seize.h](#)

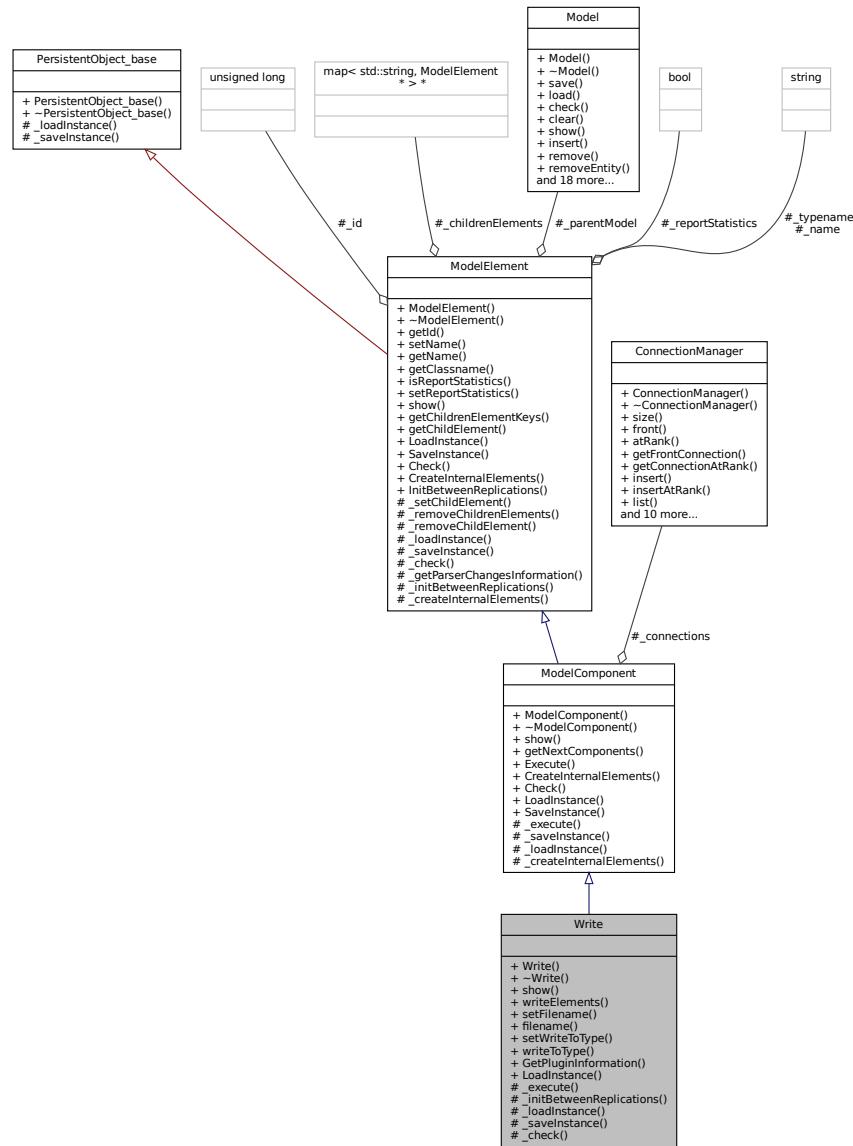
8.184 Write Class Reference

```
#include <Write.h>
```

Inheritance diagram for Write:



Collaboration diagram for Write:



Public Types

- enum `WriteToType` : int { `WriteToType::SCREEN` = 1, `WriteToType::FILE` = 2 }

Public Member Functions

- `Write (Model *model, std::string name="")`
- `virtual ~Write ()=default`
- `virtual std::string show ()`
- `List< WriteElement * > * writeElements () const`
- `void setFilename (std::string _filename)`
- `std::string filename () const`
- `void setWriteToType (WriteToType _writeToType)`
- `Write::WriteToType writeToType () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

8.184.1 Detailed Description

This component ...

Definition at line 35 of file [Write.h](#).

8.184.2 Member Enumeration Documentation

8.184.2.1 WriteToType enum `Write::WriteToType : int [strong]`

Enumerator

SCREEN	
FILE	

Definition at line 38 of file [Write.h](#).

```
00038 : int {  
00039     SCREEN = 1, FILE = 2  
00040 };
```

8.184.3 Constructor & Destructor Documentation

8.184.3.1 Write() `Write::Write (` `Model * model,` `std::string name = "")`

Definition at line 19 of file [Write.cpp](#).

```
00019 : ModelComponent(model, Util::TypeOf<Write>(), name) {  
00020 }
```

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

Here is the caller graph for this function:



8.184.3.2 ~Write() virtual Write::~Write () [virtual], [default]

8.184.4 Member Function Documentation

8.184.4.1 _check() bool Write::_check (std::string * errorMessage) [protected], [virtual]

Reimplemented from [ModelElement](#).

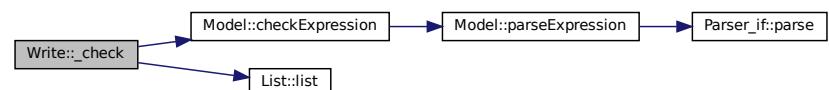
Definition at line 128 of file [Write.cpp](#).

```

00128         {
00129     bool resultAll = true;
00130     WriteElement* msgElem;
00131     unsigned short i = 0;
00132     std::list<WriteElement*>* msgs = this->_writeElements->list();
00133     for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
00134         msgElem = (*it);
00135         i++;
00136         if (msgElem->isExpression) {
00137             resultAll &= _parentModel->checkExpression(msgElem->text, "writeExpression" +
00138                 std::to_string(i), errorMessage);
00139         }
00140         // when cheking the model (before simulating it), remove the file if exists
00141         std::remove(_filename.c_str());
00142     }
00143     return resultAll;
00144 }
```

References [ModelElement::_parentModel](#), [Model::checkExpression\(\)](#), [WriteElement::isExpression](#), [List< T >::list\(\)](#), and [WriteElement::text](#).

Here is the call graph for this function:



8.184.4.2 `_execute()` void Write::_execute (Entity * entity) [protected], [virtual]

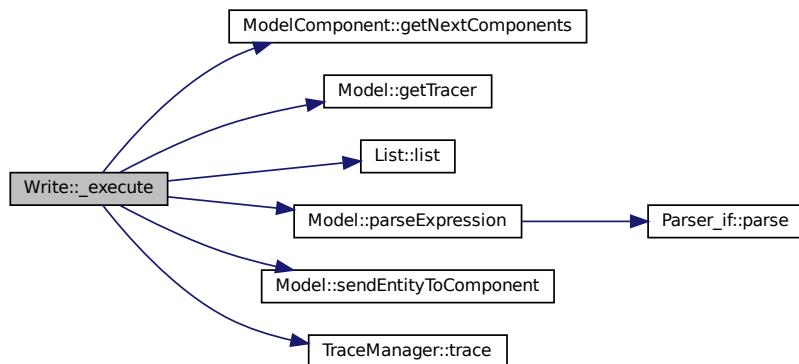
Implements [ModelComponent](#).

Definition at line 56 of file [Write.cpp](#).

```
00056
00057     WriteElement* msgElem;
00058     std::list<WriteElement*>* msgs = this->_writeElements->list();
00059     std::string message = "";
00060     for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
00061         msgElem = (*it);
00062         if (msgElem->isExpression) {
00063             message += std::to_string(_parentModel->parseExpression(msgElem->text));
00064         } else {
00065             message += msgElem->text;
00066         }
00067         if (msgElem->newline) {
00068             if (this->_writeToType == Write::WriteToType::SCREEN) { //\todo: Write To FILE not
00069                 implemented
00070                 _parentModel->getTracer()->trace(Util::TraceLevel::report, message);
00071             } else if (this->_writeToType == Write::WriteToType::FILE) {
00072                 // open file
00073                 std::ofstream savefile;
00074                 savefile.open(_filename, std::ofstream::app);
00075                 savefile << message << std::endl;
00076                 savefile.close();
00077             }
00078             message = "";
00079         }
00080     }
00081     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
0.0);
00082 }
```

References [ModelElement::_parentModel](#), [FILE](#), [ModelComponent::getNextComponents\(\)](#), [Model::getTracer\(\)](#), [WriteElement::isExpression](#), [List< T >::list\(\)](#), [WriteElement::newline](#), [Model::parseExpression\(\)](#), [Util::report](#), [SCREEN](#), [Model::sendEntityToComponent\(\)](#), [WriteElement::text](#), and [TraceManager::trace\(\)](#).

Here is the call graph for this function:



8.184.4.3 `_initBetweenReplications()` void Write::_initBetweenReplications () [protected], [virtual]

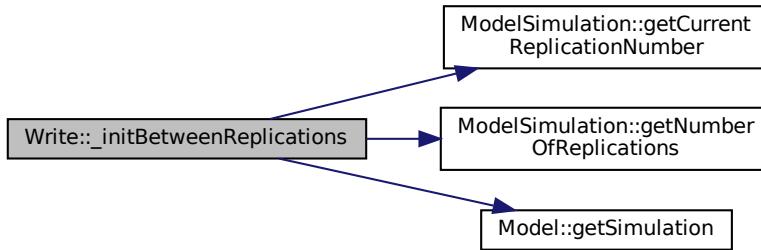
Reimplemented from [ModelElement](#).

Definition at line 84 of file [Write.cpp](#).

```
00084     {
00085         try {
00086             std::ofstream savefile;
00087             savefile.open(_filename, std::ofstream::app);
00088             savefile << "# Replication number " <<
00089             _parentModel->getSimulation()->getCurrentReplicationNumber() << "/" <<
00090             _parentModel->getSimulation()->getNumberOfReplications() << std::endl;
00091             savefile.close();
00092         } catch (...) {
00093     }
```

References [ModelElement::_parentModel](#), [ModelSimulation::getCurrentReplicationNumber\(\)](#), [ModelSimulation::getNumberOfReplications\(\)](#) and [Model::getSimulation\(\)](#).

Here is the call graph for this function:



8.184.4.4 `_loadInstance()` `bool Write::_loadInstance (`

```
std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

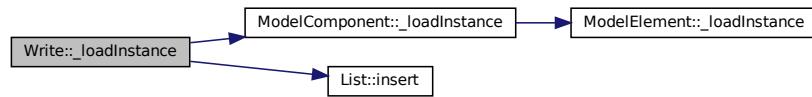
Definition at line 95 of file [Write.cpp](#).

```
00095     {
00096         bool res = ModelComponent::_loadInstance(fields);
00097         if (res) {
00098             this->_writeToType = static_cast<WriteToType>
00099             (std::stoi((*fields->find("writeToType")).second));
00100             unsigned short writesSize = std::stoi((*fields->find("writesSize")).second);
00101             for (unsigned short i = 0; i < writesSize; i++) {
00102                 std::string text = (*fields->find(text)).second;
00103                 bool isExpression = static_cast<bool> (std::stoi((*fields->find("isExpression")).second));
00104                 bool newline = static_cast<bool> (std::stoi((*fields->find("newline")).second));
00105                 this->_writeElements->insert(new WriteElement(text, isExpression, newline));
00106             }
00107         return res;
00108     }
```

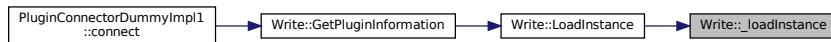
References [ModelComponent::_loadInstance\(\)](#), and [List< T >::insert\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.184.4.5 `_saveInstance()` `std::map< std::string, std::string > * Write::_saveInstance ()`
 [protected], [virtual]

Reimplemented from [ModelComponent](#).

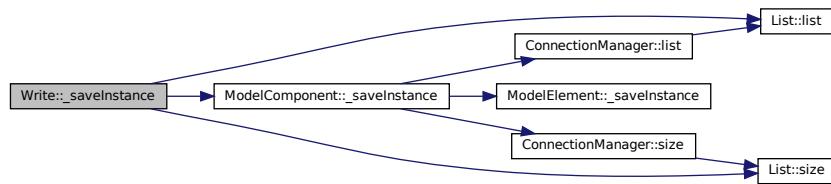
Definition at line 110 of file [Write.cpp](#).

```

00110
00111     std::map<std::string, std::string>* fields = ModelComponent::\_saveInstance\(\);
00112     fields->emplace("writeToType", std::to_string(static\_cast<int> \(\_writeToType\)));
00113     fields->emplace("writesSize", std::to_string(_writeElements->size\(\)));
00114     unsigned short i = 0;
00115     WriteElement* writeElem;
00116     for (std::list<WriteElement*>::iterator it = _writeElements->list\(\)->begin(); it != _writeElements->list\(\)->end(); it++) {
00117         writeElem = (*it);
00118         fields->emplace("isExpression" + std::to_string(i), std::to_string(writeElem->isExpression));
00119         fields->emplace("newline" + std::to_string(i), std::to_string(writeElem->newline));
00120         fields->emplace("text" + std::to_string(i), writeElem->text);
00121         i++;
00122     }
00123     //this->_writeElements
00124     return fields;
00125 }
00126 
```

References [ModelComponent::_saveInstance\(\)](#), [WriteElement::isExpression](#), [List< T >::list\(\)](#), [WriteElement::newline](#), [List< T >::size\(\)](#), and [WriteElement::text](#).

Here is the call graph for this function:



8.184.4.6 `filename()` `std::string Write::filename() const`

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

```
00044     {
00045         return _filename;
00046     }
```

8.184.4.7 `GetPluginInformation()` `PluginInformation * Write::GetPluginInformation() [static]`

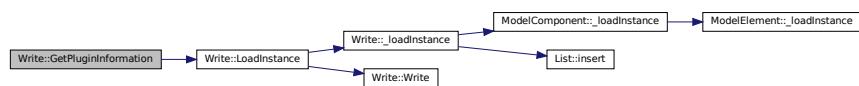
Definition at line 146 of file `Write.cpp`.

```
00146     {
00147         PluginInformation* info = new PluginInformation(Util::TypeOf<Write>(), &Write::LoadInstance);
00148         // ...
00149         return info;
00150     }
```

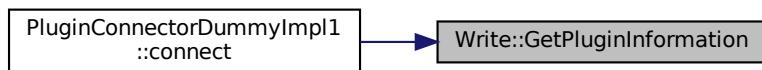
References `LoadInstance()`.

Referenced by `PluginConnectorDummyImpl1::connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.184.4.8 `LoadInstance()` `ModelComponent * Write::LoadInstance(`

```
Model * model,
std::map< std::string, std::string > * fields ) [static]
```

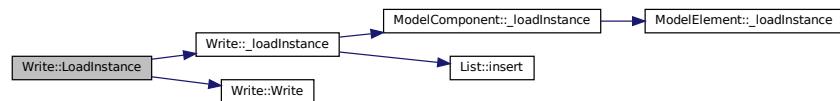
Definition at line 26 of file `Write.cpp`.

```
00026     {
00027         Write* newComponent = new Write(model);
00028         try {
00029             newComponent->_loadInstance(fields);
00030         } catch (const std::exception& e) {
00031         }
00032         return newComponent;
00033     }
```

References `_loadInstance()`, and `Write()`.

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

Here is the call graph for this function:



Here is the caller graph for this function:



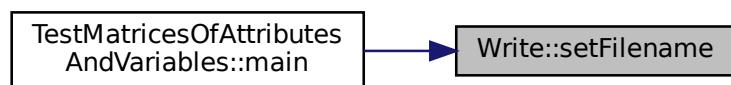
8.184.4.9 setFilename() `void Write::setFilename (std::string _filename)`

Definition at line 40 of file [Write.cpp](#).

```
00040
00041     this->_filename = _filename;
00042 }
```

Referenced by [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the caller graph for this function:



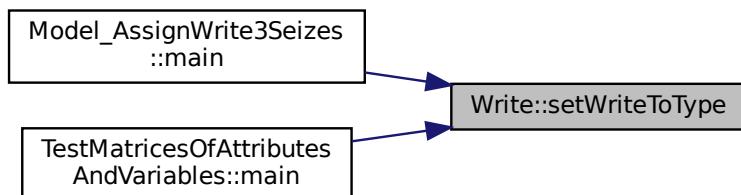
8.184.4.10 setWriteToType() void Write::setWriteToType (WriteToType _writeToType)

Definition at line 48 of file [Write.cpp](#).

```
00048
00049     this->_writeToType = _writeToType;
00050 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), and [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the caller graph for this function:



8.184.4.11 show() std::string Write::show () [virtual]

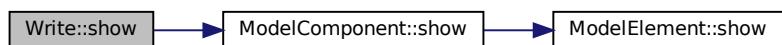
Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Write.cpp](#).

```
00022 {
00023     return ModelComponent::show() + "";
00024 }
```

References [ModelComponent::show\(\)](#).

Here is the call graph for this function:



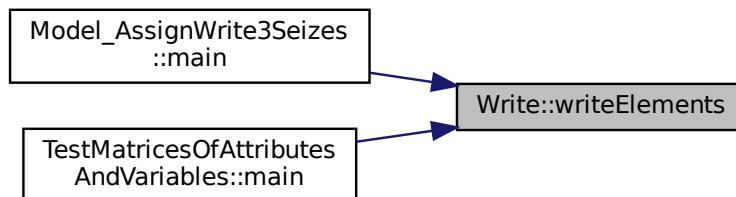
8.184.4.12 writeElements() `List< WriteElement * > * Write::writeElements () const`

Definition at line 36 of file [Write.cpp](#).

```
00036
00037     return _writeElements;
00038 }
```

Referenced by [Model_AssignWrite3Seizes::main\(\)](#), and [TestMatricesOfAttributesAndVariables::main\(\)](#).

Here is the caller graph for this function:

**8.184.4.13 writeToType()** `Write::WriteToType Write::writeToType () const`

Definition at line 52 of file [Write.cpp](#).

```
00052
00053     return _writeToType;
00054 }
```

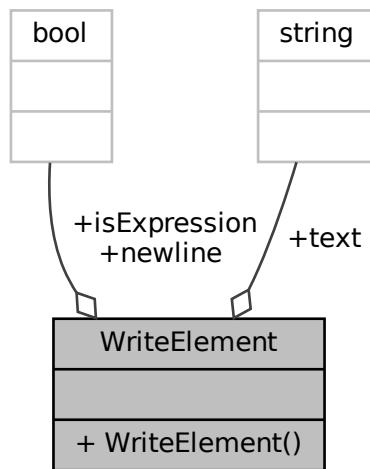
The documentation for this class was generated from the following files:

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

8.185 WriteElement Class Reference

```
#include <Write.h>
```

Collaboration diagram for WriteElement:



Public Member Functions

- `WriteElement (std::string text, bool isExpression=false, bool newline=false)`

Public Attributes

- `std::string text`
- `bool isExpression`
- `bool newline`

8.185.1 Detailed Description

Definition at line 19 of file [Write.h](#).

8.185.2 Constructor & Destructor Documentation

8.185.2.1 WriteElement() `WriteElement::WriteElement (`
 `std::string text,`
 `bool isExpression = false,`
 `bool newline = false) [inline]`

Definition at line 22 of file [Write.h](#).

```

00022
00023     this->text = text;
00024     this->isExpression = isExpression;
00025     this->newline = newline;
00026 }
```

References `isExpression`, `newline`, and `text`.

8.185.3 Member Data Documentation

8.185.3.1 **isExpression** bool WriteElement::isExpression

Definition at line 28 of file [Write.h](#).

Referenced by [Write::_check\(\)](#), [Write::_execute\(\)](#), [Write::_saveInstance\(\)](#), and [WriteElement\(\)](#).

8.185.3.2 **newline** bool WriteElement::newline

Definition at line 29 of file [Write.h](#).

Referenced by [Write::_execute\(\)](#), [Write::_saveInstance\(\)](#), and [WriteElement\(\)](#).

8.185.3.3 **text** std::string WriteElement::text

Definition at line 27 of file [Write.h](#).

Referenced by [Write::_check\(\)](#), [Write::_execute\(\)](#), [Write::_saveInstance\(\)](#), and [WriteElement\(\)](#).

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

- [Write.h](#)

9 File Documentation

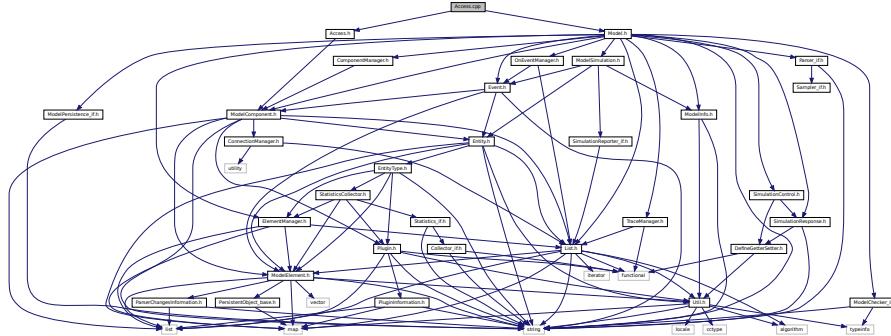
9.1 .dep.inc File Reference

9.2 .dep.inc

```
00001 # This code depends on make tool being used
00002 DEPFILES=$(wildcard ${addsuffix .d, ${OBJECTFILES} ${TESTOBJECTFILES}})
00003 ifneq (${DEPFILES},)
00004 include ${DEPFILES}
00005 endif
```

9.3 Access.cpp File Reference

```
#include "Access.h"
#include "Model.h"
Include dependency graph for Access.cpp:
```



9.4 Access.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Access.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:14
00012 */
00013
00014 #include "Access.h"
00015
00016 #include "Model.h"
00017
00018 Access::Access(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Access>(), name) {
00019 }
00020
00021 std::string Access::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Access::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Access* newComponent = new Access(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031     return newComponent;
00032 }
00033
00034
00035 void Access::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Access::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Access::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Access::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
```

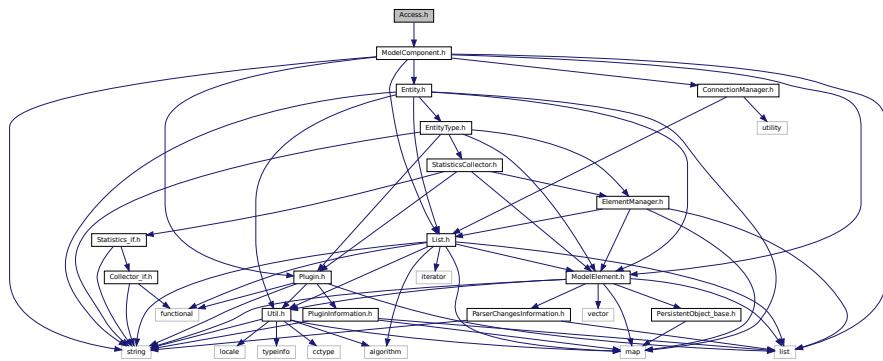
```

00054     return fields;
00055 }
00056
00057 bool Access::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Access::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Access>(), &Access::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

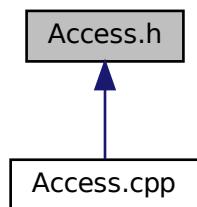
```

9.5 Access.h File Reference

#include "ModelComponent.h"
Include dependency graph for Access.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Access](#)

9.6 Access.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Access.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:14
00012 */
00013
00014 #ifndef ACCESS_H
00015 #define ACCESS_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Access : public ModelComponent {
00020 public: // constructors
00021     Access(Model* model, std::string name = "");
00022     virtual ~Access() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00028 protected: // virtual
00029     virtual void _execute(Entity* entity);
00030     virtual void _initBetweenReplications();
00031     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032     virtual std::map<std::string, std::string>* _saveInstance();
00033     virtual bool _check(std::string* errorMessage);
00034 private: // methods
00035 private: // attributes 1:1
00036 private: // attributes 1:n
00037 };
00038
00039
00040
00041 #endif /* ACCESS_H */
```

9.7 AnalysisOfVariance_if.h File Reference

Classes

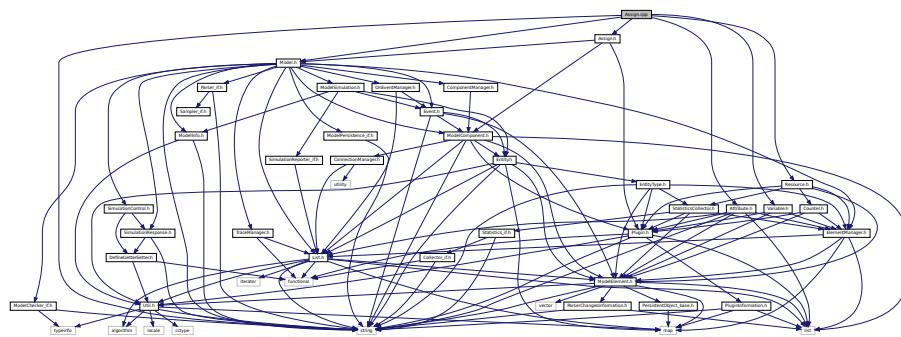
- class [AnalysisOfVariance_if](#)

9.8 AnalysisOfVariance_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: AnalysisOfVariance_if.h
00009  * Author: rlcancian
00010 *
00011 * Created on 21 de Junho de 2019, 16:08
00012 */
00013
00014 #ifndef ANALYSISOFVARIANCE_IF_H
00015 #define ANALYSISOFVARIANCE_IF_H
00016
00017 class AnalysisOfVariance_if {
00018 public:
00019     virtual void setCollector() = 0;
00020 };
00021
00022 #endif /* ANALYSISOFVARIANCE_IF_H */
```

9.9 Assign.cpp File Reference

```
#include "Assign.h"
#include <string>
#include "Model.h"
#include "Variable.h"
#include "Attribute.h"
#include "Resource.h"
Include dependency graph for Assign.cpp:
```



9.10 Assign.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Assign.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Agosto de 2018, 10:10
00012 */
00013
00014 #include "Assign.h"
00015 #include <string>
00016 #include "Model.h"
00017 #include "Variable.h"
00018 #include "Attribute.h"
00019 #include "Resource.h"
00020
00021 Assign::Assign(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Assign>(), name) {
00022 }
00023
00024 std::string Assign::show() {
00025     return ModelComponent::show() +
00026         "";
00027 }
00028
00029 List<Assign::Assignment*>* Assign::getAssignments() const {
00030     return _assignments;
00031 }
00032
00033 PluginInformation* Assign::GetPluginInformation() {
00034     PluginInformation* info = new PluginInformation(Util::TypeOf<Assign>(), &Assign::LoadInstance);
00035     info->insertDynamicLibFileDependence("attribute.so");
00036     info->insertDynamicLibFileDependence("variable.so");
00037     return info;
00038 }
00039
00040 ModelComponent* Assign::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00041     Assign* newComponent = new Assign(model);
00042     try {
00043         newComponent->_loadInstance(fields);
00044     } catch (const std::exception& e) {
00045
00046     }
00047     return newComponent;
00048 }
```

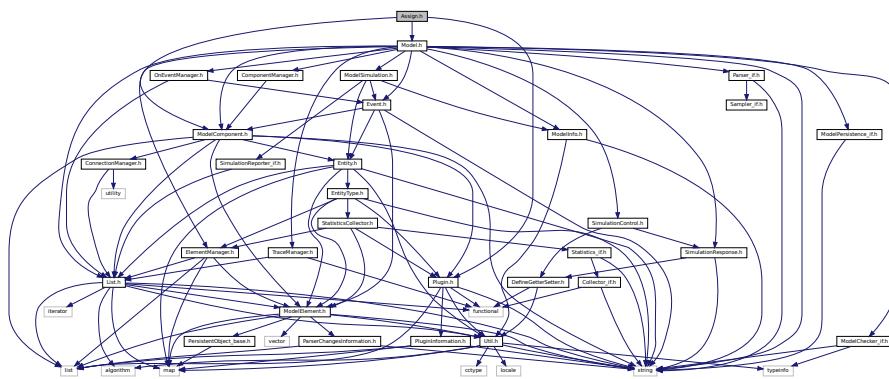
```

00048 }
00049
00050 void Assign::_execute(Entity* entity) {
00051     Assignment* let;
00052     std::list<Assignment*>* lets = this->_assignments->list();
00053     for (std::list<Assignment*>::iterator it = lets->begin(); it != lets->end(); it++) {
00054         let = (*it);
00055         double value = _parentModel->parseExpression(let->getExpression());
00056         _parentModel->parseExpression(let->getDestination() + "=" + std::to_string(value));
00057         _parentModel->getTracer()->trace("Let \"\" + let->getDestination() + "\" = " +
00058             std::to_string(value) + " / " + let->getExpression());
00059     }
00060     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00061     0.0);
00062
00063 void Assign::_initBetweenReplications() {
00064 }
00065
00066 bool Assign::_loadInstance(std::map<std::string, std::string>* fields) {
00067     bool res = ModelComponent::_loadInstance(fields);
00068     if (res) {
00069         unsigned int nv = std::stoi((*(fields->find("assignments"))).second);
00070         for (unsigned int i = 0; i < nv; i++) {
00071             //DestinationType dt = static_cast<DestinationType>
00072             (std::stoi((*(fields->find("destinationType") + std::to_string(i))).second));
00073             std::string dest = ((*(fields->find("destination" + std::to_string(i)))).second);
00074             std::string exp = ((*(fields->find("expression" + std::to_string(i)))).second);
00075             Assignment* assmt = new Assignment(dest, exp);
00076             this->_assignments->insert(assmt);
00077         }
00078     }
00079     return res;
00080 }
00081 std::map<std::string, std::string>* Assign::_saveInstance() {
00082     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00083     //Util::TypeOf<Assign>());
00084     Assignment* let;
00085     fields->emplace("assignments", std::to_string(_assignments->size()));
00086     unsigned short i = 0;
00087     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->end(); it++) {
00088         let = (*it);
00089         //fields->emplace("destinationType" + std::to_string(i), std::to_string(static_cast<int>
00090         (let->getDestinationType())));
00091         fields->emplace("destination" + std::to_string(i), let->getDestination());
00092         fields->emplace("expression" + std::to_string(i), "\"" + let->getExpression() + "\"");
00093     }
00094     return fields;
00095 }
00096 bool Assign::_check(std::string* errorMessage) {
00097     Assignment* let;
00098     bool resultAll = true;
00099     // \todo: Reimplement it. Since 201910, attributes may have index, just like "atrib1[2]" or
00100     // "att[10,1]". Because of that, the string may contain not only the name of the attribute, but also its
00101     // index and therefore, fails on the test below.
00102     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->end(); it++) {
00103         let = (*it);
00104         resultAll &= _parentModel->checkExpression(let->getExpression(), "assignment", errorMessage);
00105     }
00106     return resultAll;
00107 }
```

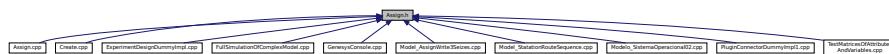
9.11 Assign.h File Reference

```
#include "ModelComponent.h"
#include "Model.h"
#include "Plugin.h"
```

Include dependency graph for Assign.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Assign](#)
- class [Assign::Assignment](#)

9.12 Assign.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File: Assign.h
00008  * Author: rafael.luiz.cancian
00009  *
00010  */
00011 /* Created on 31 de Agosto de 2018, 10:10
00012 */
00013
00014 #ifndef ASSIGN_H
00015 #define ASSIGN_H
00016
00017 #include "ModelComponent.h"
00018 #include "Model.h"
00019 #include "Plugin.h"
00020
00058 class Assign : public ModelComponent {
00059 public:
00060
00064     class Assignment {
00065         public:
00066
00067             Assignment(std::string destination, std::string expression) {
00068                 this->_destination = destination;
00069                 this->_expression = expression;
00070                 // an assignment is always in the form:
00071                 // (destinationType) destination = expression
00072             };
00073
00074             void setDestination(std::string _destination) {
00075                 this->_destination = _destination;
00076             }
00077

```

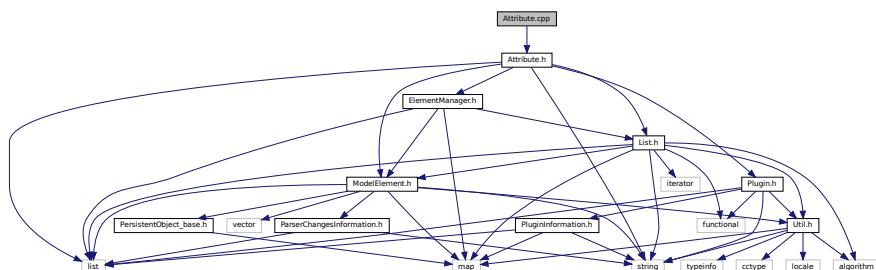
```

00078     std::string getDestination() const {
00079         return _destination;
00080     }
00081
00082     void setExpression(std::string _expression) {
00083         this->_expression = _expression;
00084     }
00085
00086     std::string getExpression() const {
00087         return _expression;
00088     }
00089 private:
00090     std::string _destination = "";
00091     std::string _expression = "";
00092
00093 };
00094 public:
00095     Assign(Model* model, std::string name = "");
00096     virtual ~Assign() = default;
00097 public:
00098     virtual std::string show();
00099 public:
00100     static PluginInformation* GetPluginInformation();
00101     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00102 public:
00103     List<Assignment*>* getAssignments() const;
00104 protected:
00105     virtual void _execute(Entity* entity);
00106     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00107     virtual void _initBetweenReplications();
00108     virtual std::map<std::string, std::string>* _saveInstance();
00109     virtual bool _check(std::string* errorMessage);
00110 private:
00111 private:
00112     List<Assignment*>* _assignments = new List<Assignment*>();
00113 };
00114
00115 #endif /* ASSIGN_H */
00116

```

9.13 Attribute.cpp File Reference

```
#include "Attribute.h"
Include dependency graph for Attribute.cpp:
```



9.14 Attribute.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Attribute.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 25 de Setembro de 2018, 16:37
00012 */
00013
00014 #include "Attribute.h"

```

```

00015 //using namespace GenesysKernel;
00016
00017
00018 Attribute::Attribute(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Attribute>(),
00019     name) {
00020
00021     std::string Attribute::show() {
00022         return ModelElement::show();
00023     }
00024
00025     bool Attribute::_loadInstance(std::map<std::string, std::string>* fields) {
00026         return ModelElement::_loadInstance(fields);
00027     }
00028
00029     PluginInformation* Attribute::GetPluginInformation() {
00030         PluginInformation* info = new PluginInformation(Util::TypeOf<Attribute>(),
00031             &Attribute::LoadInstance);
00032         return info;
00033     }
00034
00035     ModelElement* Attribute::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00036         Attribute* newElement = new Attribute(model);
00037         try {
00038             newElement->_loadInstance(fields);
00039         } catch (const std::exception& e) {
00040
00041         }
00042         return newElement;
00043     }
00044
00045     std::map<std::string, std::string>* Attribute::_saveInstance() {
00046         std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00047         //Util::TypeOf<Attribute>());
00048         return fields;
00049     }
00050     bool Attribute::_check(std::string* errorMessage) {
00051         return true;
00052     }

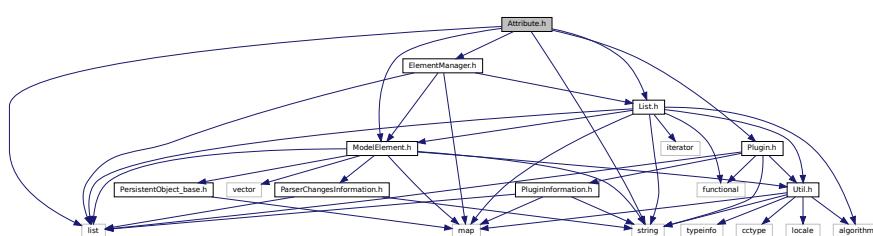
```

9.15 Attribute.h File Reference

```

#include <string>
#include <list>
#include "List.h"
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
Include dependency graph for Attribute.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class Attribute

9.16 Attribute.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Attribute.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 25 de Setembro de 2018, 16:37
00012 */
00013
00014 #ifndef ATTRIBUTE_H
00015 #define ATTRIBUTE_H
00016
00017 #include <string>
00018 #include <list>
00019 #include "List.h"
00020 #include "ModelElement.h"
00021 #include "ElementManager.h"
00022 #include "Plugin.h"
00023
00024 //namespace GenesysKernel {
00025
00063     class Attribute : public ModelElement {
00064     public:
00065         Attribute(Model* model, std::string name = "");
00066         virtual ~Attribute() = default;
00067     public:
00068         virtual std::string show();
00069     public:
00070         static PluginInformation* GetPluginInformation();
00071         static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00072     protected:
00073         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00074         virtual std::map<std::string, std::string>* _saveInstance();
00075         virtual bool _check(std::string* errorMessage);
00076     private:
00077         //List<unsigned int>* _dimensionSizes = new List<unsigned int>();
00078     };
00079 //namespace\\
00080 #endif /* ATTRIBUTE_H */
```

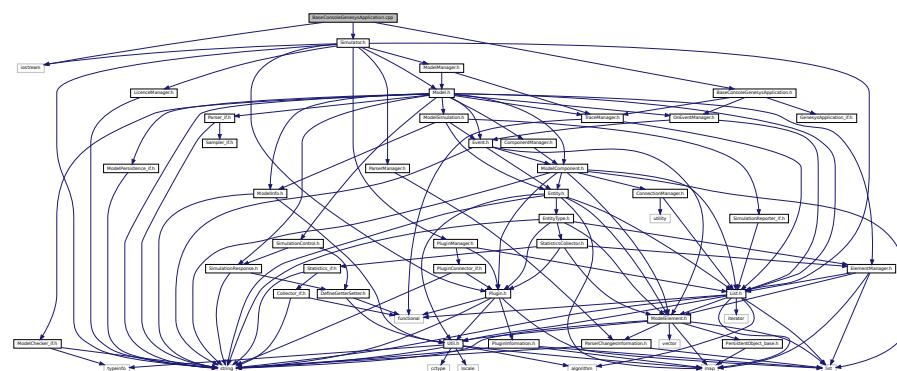
9.17 BaseConsoleGenesysApplication.cpp File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

```
#include <iostream>
```

```
#include "Simulator.h"
```

Include dependency graph for BaseConsoleGenesysApplication.cpp:



9.18 BaseConsoleGenesysApplication.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: BaseConsoleGenesysApplication.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 16:25
00012 */
00013
00014 #include "BaseConsoleGenesysApplication.h"
00015 #include <iostream>
00016
00017 #include "Simulator.h"
00018
00019 BaseConsoleGenesysApplication::BaseConsoleGenesysApplication() {
00020 }
00021
00022
00023 // default Trace Handlers
00024
00025 void BaseConsoleGenesysApplication::traceHandler(TraceEvent e) {
00026     std::cout << e.getText() << std::endl;
00027 }
00028
00029 void BaseConsoleGenesysApplication::traceErrorHandler(TraceErrorEvent e) {
00030     std::cout << e.getText() << std::endl;
00031 }
00032
00033 void BaseConsoleGenesysApplication::traceReportHandler(TraceEvent e) {
00034     std::cout << "" << e.getText() << "" << std::endl;
00035 }
00036
00037 void BaseConsoleGenesysApplication::traceSimulationHandler(TraceSimulationEvent e) {
00038     std::cout << e.getText() << std::endl;
00039 }
00040
00041
00042 // default Event Handlers
00043
00044 void BaseConsoleGenesysApplication::onSimulationStartHandler(SimulationEvent* re) {
00045     std::cout << "(Event Handler) " << "Simulation is starting" << std::endl;
00046 }
00047
00048 void BaseConsoleGenesysApplication::onReplicationStartHandler(SimulationEvent* re) {
00049     std::cout << "(Event Handler) " << "Replication " << re->getReplicationNumber() << " starting." <<
00050     std::endl;
00051
00052 void BaseConsoleGenesysApplication::onProcessEventHandler(SimulationEvent* re) {
00053     std::cout << "(Event Handler) " << "Processing event " << re->getEventProcessed()->show() <<
00054     std::endl;
00055
00056 void BaseConsoleGenesysApplication::onReplicationEndHandler(SimulationEvent* re) {
00057     std::cout << "(Event Handler) " << "Replication " << re->getReplicationNumber() << " ending." <<
00058     std::endl;
00059
00060 void BaseConsoleGenesysApplication::onSimulationEndHandler(SimulationEvent* re) {
00061     std::cout << "Event (Handler) " << "Replication " << re->getReplicationNumber() << " ending." <<
00062     std::endl;
00063
00064 void BaseConsoleGenesysApplication::onEntityRemoveHandler(SimulationEvent* re) {
00065     std::cout << "(Event Handler) " << "Entity " << re->getEventProcessed()->getEntity() << " was
00066     removed." << std::endl;
00067
00068 void BaseConsoleGenesysApplication::setDefaultEventHandlers(OnEventManager* oem) {
00069     //oem->addOnSimulationStartHandler(this,
00070     //&BaseConsoleGenesysApplication::onSimulationStartHandler);
00071     //oem->addOnReplicationStartHandler(this,
00072     //BaseConsoleGenesysApplication::onReplicationStartHandler);
00073     oem->addOnProcessEventHandler(this, &BaseConsoleGenesysApplication::onProcessEventHandler);
00074     //oem->addOnReplicationEndHandler(this, &BaseConsoleGenesysApplication::onReplicationEndHandler);
00075     //oem->addOnSimulationEndHandler(this, &BaseConsoleGenesysApplication::onSimulationEndHandler);
00076     //oem->addOnEntityRemoveHandler(this, &BaseConsoleGenesysApplication::onEntityRemoveHandler);
00077 }
00078
00079 void BaseConsoleGenesysApplication::setDefaultTraceHandlers(TraceManager* tm) {
00080     tm->addTraceHandler<BaseConsoleGenesysApplication>(this,

```

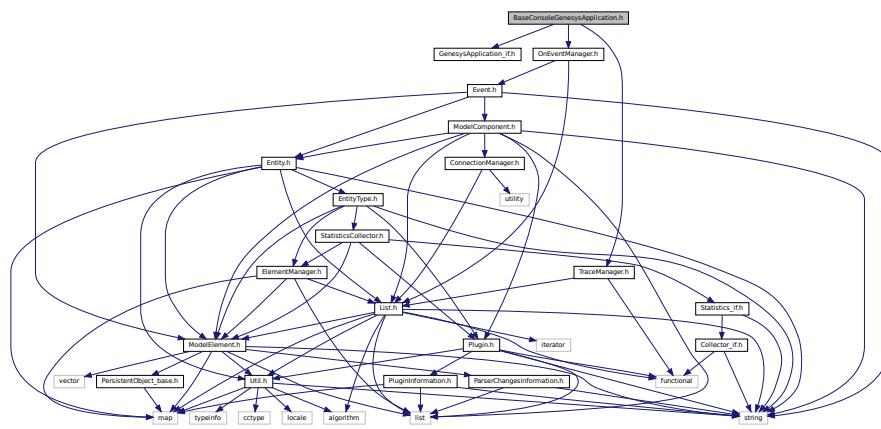
```

    &BaseConsoleGenesysApplication::traceHandler);
00079     tm->addTraceErrorHandler<BaseConsoleGenesysApplication>(this,
00080     &BaseConsoleGenesysApplication::traceErrorHandler);
00081     tm->addTraceReportHandler<BaseConsoleGenesysApplication>(this,
00082     &BaseConsoleGenesysApplication::traceReportHandler);
00083     tm->addTraceSimulationHandler<BaseConsoleGenesysApplication>(this,
00084     &BaseConsoleGenesysApplication::traceSimulationHandler);
00085 }
00086
00084 void BaseConsoleGenesysApplication::insertFakePluginsByHand(Simulator* simulator) {
00085     // model components
00086     // arena basic process
00087     simulator->getPlugins()->insert("create.so");
00088     simulator->getPlugins()->insert("dispose.so");
00089     simulator->getPlugins()->insert("decide.so");
00090     simulator->getPlugins()->insert("batch.so");
00091     simulator->getPlugins()->insert("separate.so");
00092     simulator->getPlugins()->insert("assign.so");
00093     simulator->getPlugins()->insert("record.so");
00094     simulator->getPlugins()->insert("submodel.so");
00095     simulator->getPlugins()->insert("entitytype.so");
00096     simulator->getPlugins()->insert("entitygroup.so");
00097     simulator->getPlugins()->insert("attribute.so");
00098     simulator->getPlugins()->insert("counter.so");
00099     simulator->getPlugins()->insert("queue.so");
00100     simulator->getPlugins()->insert("set.so");
00101     simulator->getPlugins()->insert("resource.so");
00102     simulator->getPlugins()->insert("variable.so");
00103     simulator->getPlugins()->insert("schedule.so");
00104     simulator->getPlugins()->insert("entitygroup.so");
00105     // arena advanced process
00106     simulator->getPlugins()->insert("delay.so");
00107     simulator->getPlugins()->insert("dropoff.so");
00108     simulator->getPlugins()->insert("hold.so");
00109     simulator->getPlugins()->insert("match.so");
00110     simulator->getPlugins()->insert("pickup.so");
00111     simulator->getPlugins()->insert("read.so");
00112     simulator->getPlugins()->insert("write.so");
00113     simulator->getPlugins()->insert("release.so");
00114     simulator->getPlugins()->insert("remove.so");
00115     simulator->getPlugins()->insert("seize.so");
00116     simulator->getPlugins()->insert("search.so");
00117     simulator->getPlugins()->insert("signal.so");
00118     simulator->getPlugins()->insert("store.so");
00119     simulator->getPlugins()->insert("unstore.so");
00120     simulator->getPlugins()->insert("expression.so");
00121     simulator->getPlugins()->insert("failure.so");
00122     simulator->getPlugins()->insert("file.so");
00123     simulator->getPlugins()->insert("statisticscollector.so");
00124     simulator->getPlugins()->insert("storage.so");
00125     // arena transfer station
00126     simulator->getPlugins()->insert("enter.so");
00127     simulator->getPlugins()->insert("leave.so");
00128     simulator->getPlugins()->insert("pickstation.so");
00129     simulator->getPlugins()->insert("route.so");
00130     simulator->getPlugins()->insert("sequence.so");
00131     simulator->getPlugins()->insert("station.so");
00132     // arena transfer conveyor
00133     simulator->getPlugins()->insert("access.so");
00134     simulator->getPlugins()->insert("exit.so");
00135     simulator->getPlugins()->insert("start.so");
00136     simulator->getPlugins()->insert("stop.so");
00137     simulator->getPlugins()->insert("conveyour.so");
00138     simulator->getPlugins()->insert("segment.so");
00139     // arena transfer transport
00140     simulator->getPlugins()->insert("alocate.so");
00141     simulator->getPlugins()->insert("free.so");
00142     simulator->getPlugins()->insert("halt.so");
00143     simulator->getPlugins()->insert("move.so");
00144     simulator->getPlugins()->insert("request.so");
00145     simulator->getPlugins()->insert("transporter.so");
00146     simulator->getPlugins()->insert("distance.so");
00147     simulator->getPlugins()->insert("network.so");
00148     simulator->getPlugins()->insert("networklink.so");
00149     // others
00150     simulator->getPlugins()->insert("dummy.so");
00151     simulator->getPlugins()->insert("lsode.so");
00152     simulator->getPlugins()->insert("biochemical.so");
00153     simulator->getPlugins()->insert("markovchain.so");
00154     simulator->getPlugins()->insert("cellularautomata.so");
00155     simulator->getPlugins()->insert("cppforgenesys.so");
00156 }

```

9.19 BaseConsoleGenesysApplication.h File Reference

```
#include "GenesysApplication_if.h"
#include "TraceManager.h"
#include "OnEventManager.h"
Include dependency graph for BaseConsoleGenesysApplication.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [BaseConsoleGenesysApplication](#)

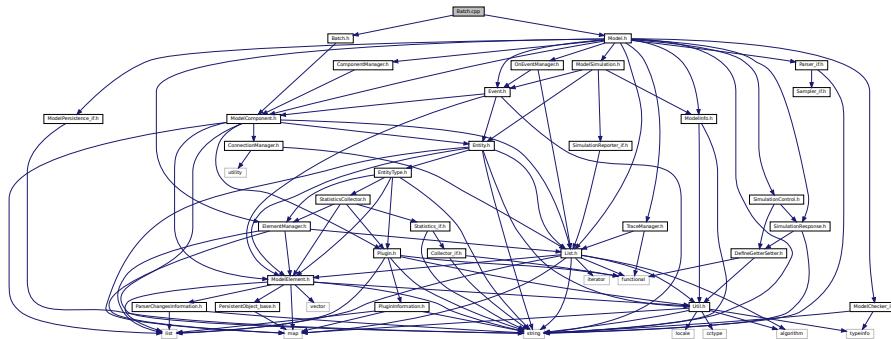
9.20 BaseConsoleGenesysApplication.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: BaseConsoleGenesysApplication.h
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 16:25
00012 */
00013
00014 #ifndef BASECONSOLEGENESYSAPPLICATION_H
00015 #define BASECONSOLEGENESYSAPPLICATION_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "TraceManager.h"
00019 #include "OnEventManager.h"
00020
00021 class BaseConsoleGenesysApplication : public GenesysApplication_if {
00022 public:
00023     BaseConsoleGenesysApplication();
00024     virtual ~BaseConsoleGenesysApplication() = default;
00025 public:
00026     virtual int main(int argc, char** argv) = 0;
00027 public:
```

```
00028     void setDefaultTraceHandlers(TraceManager* tm);
00029     void setDefaultEventHandlers(OnEventManager* oem);
00030     void insertFakePluginsByHand(Simulator* simulator);
00031 protected:
00032     // default Trace Handlers
00033     virtual void traceHandler(TraceEvent e);
00034     virtual void traceErrorHandler(TraceErrorEvent e);
00035     virtual void traceReportHandler(TraceEvent e);
00036     virtual void traceSimulationHandler(TraceSimulationEvent e);
00037     // default Event Handlers
00038     virtual void onSimulationStartHandler(SimulationEvent* re);
00039     virtual void onReplicationStartHandler(SimulationEvent* re);
00040     virtual void onProcessEventHandler(SimulationEvent* re);
00041     virtual void onReplicationEndHandler(SimulationEvent* re);
00042     virtual void onSimulationEndHandler(SimulationEvent* re);
00043     virtual void onEntityRemoveHandler(SimulationEvent* re);
00044 private:
00045
00046 };
00047
00048 #endif /* BASECONSOLEGENESYSAPPLICATION_H */
00049
```

9.21 Batch.cpp File Reference

```
#include "Batch.h"  
#include "Model.h"  
Include dependency graph for Batch.cpp:
```



9.22 Batch.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:    Batch.cpp
00009 * Author:  rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Batch.h"
00015
00016 #include "Model.h"
00017
00018 Batch::Batch(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Batch>(), name) {
00019 }
00020
00021 std::string Batch::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Batch::LoadInstance(Model* model, std::map<std::string, std::string*> fields) {
00026     Batch* newComponent = new Batch(model);
00027     try {
```

```

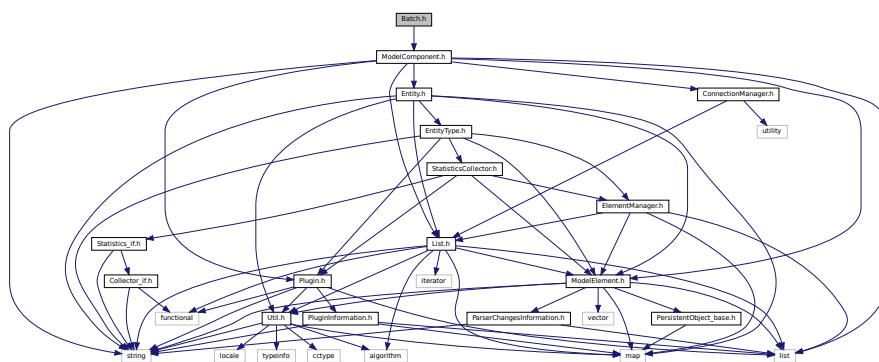
00028     newComponent->_loadInstance(fields);
00029 } catch (const std::exception& e) {
00030 }
00031 }
00032 return newComponent;
00033 }
00034
00035 void Batch::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Batch::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Batch::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Batch::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Batch::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Batch::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Batch>(), &Batch::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

```

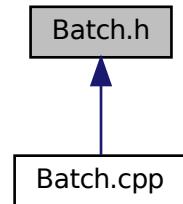
9.23 Batch.h File Reference

#include "ModelComponent.h"

Include dependency graph for Batch.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Batch](#)

9.24 Batch.h

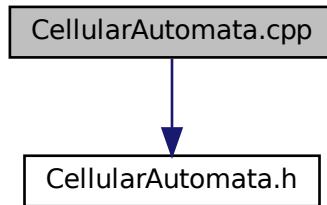
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  *
00008  * File:      Batch.h
00009  * Author:    rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef BATCH_H
00015 #define BATCH_H
00016
00017 #include "ModelComponent.h"
00018
00053 class Batch : public ModelComponent {
00054 public: // constructors
00055     Batch(Model* model, std::string name = "");
00056     virtual ~Batch() = default;
00057 public: // virtual
00058     virtual std::string show();
00059 public: // static
00060     static PluginInformation* GetPluginInformation();
00061     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00062 protected: // virtual
00063     virtual void _execute(Entity* entity);
00064     virtual void _initBetweenReplications();
00065     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00066     virtual std::map<std::string, std::string>* _saveInstance();
00067     virtual bool _check(std::string* errorMessage);
00068 private: // methods
00069 private: // attributes 1:1
00070 private: // attributes 1:n
00071 };
00072
00073
00074 #endif /* BATCH_H */
00075

```

9.25 CellularAutomata.cpp File Reference

```
#include "CellularAutomata.h"  
Include dependency graph for CellularAutomata.cpp:
```

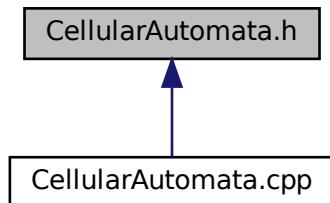


9.26 CellularAutomata.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006 /*  
00007 *  
00008 * File: CellularAutomata.cpp  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 03 de Junho de 2019, 15:14  
00012 */  
00013  
00014 #include "CellularAutomata.h"  
00015  
00016 CellularAutomata::CellularAutomata() {  
00017 }  
00018  
00019
```

9.27 CellularAutomata.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [CelularAutomata](#)

9.28 CellularAutomata.h

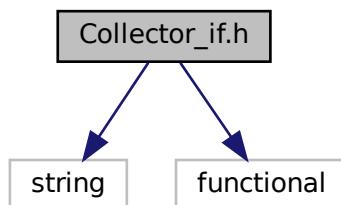
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  CelularAutomata.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef CELULARAUTOMATA_H
00015 #define CELULARAUTOMATA_H
00016
00017 class CelularAutomata {
00018 public:
00019     CelularAutomata();
00020     virtual ~CelularAutomata() = default;
00021 private:
00022 };
00024
00025 #endif /* CELULARAUTOMATA_H */
00026

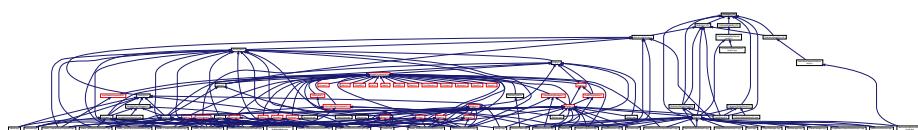
```

9.29 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)`

9.29.1 Typedef Documentation

9.29.1.1 CollectorAddValueHandler `typedef std::function<void(double) > CollectorAddValueHandler`

Definition at line [22](#) of file [Collector_if.h](#).

9.29.1.2 CollectorClearHandler `typedef std::function<void() > CollectorClearHandler`

Definition at line [29](#) of file [Collector_if.h](#).

9.29.2 Function Documentation

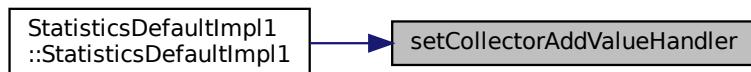
9.29.2.1 setCollectorAddValueHandler() `template<typename Class >` `CollectorAddValueHandler setCollectorAddValueHandler (` `void(Class::*)(double) function,` `Class * object)`

Definition at line [25](#) of file [Collector_if.h](#).

```
00025
00026     return std::bind(function, object, std::placeholders::_1);
00027 }
```

Referenced by [StatisticsDefaultImpl1::StatisticsDefaultImpl1\(\)](#).

Here is the caller graph for this function:



```
9.29.2.2 setCollectorClearHandler() template<typename Class >
CollectorClearHandler setCollectorClearHandler (
    void(Class::*)( ) function,
    Class * object )
```

Definition at line 32 of file [Collector_if.h](#).

```
00032
00033     return std::bind(function, object);
00034 }
```

Referenced by [StatisticsDefaultImpl1::StatisticsDefaultImpl1\(\)](#).

Here is the caller graph for this function:

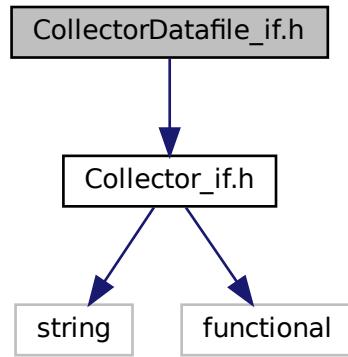


9.30 Collector_if.h

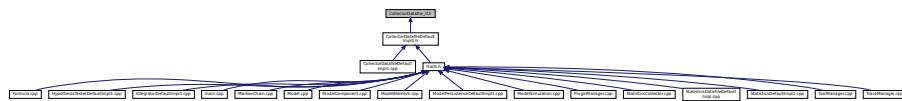
```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Collector_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 14:16
00012 */
00013
00014 #ifndef COLLECTOR_IF_H
00015 #define COLLECTOR_IF_H
00016
00017 #include <string>
00018 #include <functional>
00019
00020 typedef std::function<void(double) > CollectorAddValueHandler;
00021
00022 template<typename Class>
00023 CollectorAddValueHandler setCollectorAddValueHandler(void (Class::*function)(double), Class * object)
00024 {
00025     return std::bind(function, object, std::placeholders::_1);
00026 }
00027
00028
00029 typedef std::function<void() > CollectorClearHandler;
00030
00031 template<typename Class>
00032 CollectorClearHandler setCollectorClearHandler(void (Class::*function)( ), Class * object) {
00033     return std::bind(function, object);
00034 }
00035
00036 class Collector_if {
00037     public:
00038         virtual void clear() = 0;
00039         virtual void addValue(double value) = 0;
00040         virtual double getLastValue() = 0;
00041         virtual unsigned long numElements() = 0;
00042
00043     public:
00044         virtual void setAddValueHandler(CollectorAddValueHandler addValueHandler) = 0;
00045         virtual void setClearHandler(CollectorClearHandler clearHandler) = 0;
00046
00047 };
00048
00049 #endif /* COLLECTOR_IF_H */
00050
```

9.31 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](#)

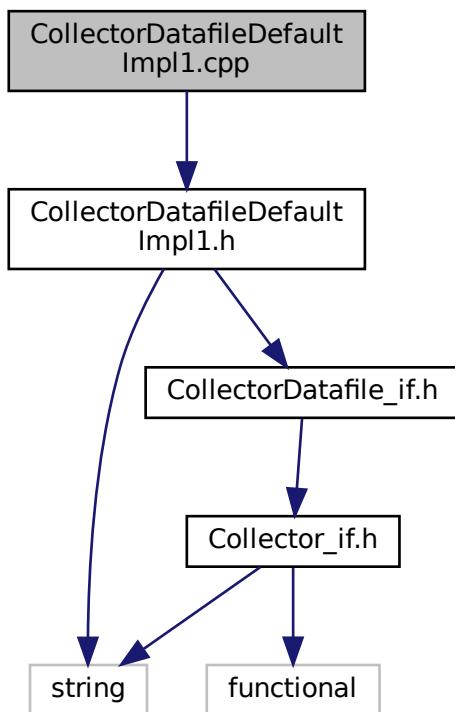
9.32 CollectorDatafile_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File:   CollectorDatafile_if.h
00008  * Author: rafael.luiz.cancian
00009  *
00010  *
00011  * Created on 30 de Agosto de 2018, 16:45
00012 */
00013 #ifndef COLLECTORDATAFILE_IF_H
00014 #define COLLECTORDATAFILE_IF_H
00015
00016
00017 #include "Collector_if.h"
00018
00019 class CollectorDatafile_if : public Collector_if {
00020 public:
00021     virtual double getValue(unsigned int rank) = 0;
00022     virtual void seekFirstValue() = 0;
00023     virtual double getNextValue() = 0;
00024     virtual std::string getDataFilename() = 0;
00025     virtual void setDataFilename(std::string filename) = 0;
00026 };
00027
00028 #endif /* COLLECTORDATAFILE_IF_H */
00029
  
```

9.33 CollectorDatafileDefaultImpl1.cpp File Reference

```
#include "CollectorDatafileDefaultImpl1.h"
Include dependency graph for CollectorDatafileDefaultImpl1.cpp:
```



9.34 CollectorDatafileDefaultImpl1.cpp

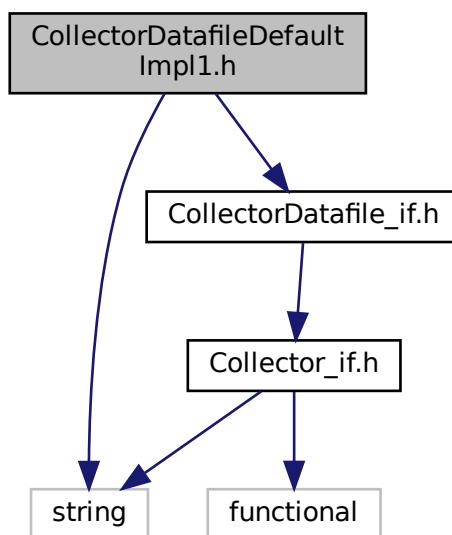
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File:   CollectorDatafileDefaultImpl1.cpp
00008  * Author: rafael.luiz.cancian
00009  *
00010  *
00011  * Created on 1 de Agosto de 2018, 20:58
00012 */
00013
00014 #include "CollectorDatafileDefaultImpl1.h"
00015
00016 CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1() {
00017 }
00018
00019 void CollectorDatafileDefaultImpl1::clear() {
00020 }
00021
00022 void CollectorDatafileDefaultImpl1::addValue(double value) {
00023 }
00024
00025 double CollectorDatafileDefaultImpl1::getLastValue() {
00026     return 0.0; // \todo:
00027 }
00028
00029 unsigned long CollectorDatafileDefaultImpl1::numElements() {
  
```

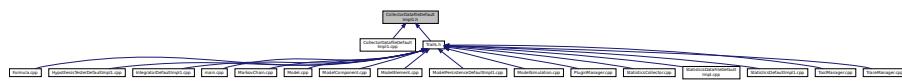
```
00030     return 0.0; // \todo:
00031 }
00032
00033 double CollectorDatafileDefaultImpl1::getValue(unsigned int num) {
00034     return 0.0; // \todo:
00035 }
00036
00037 double CollectorDatafileDefaultImpl1::getNextValue() {
00038     return 0.0; // \todo:
00039 }
00040
00041 void CollectorDatafileDefaultImpl1::seekFirstValue() {
00042 }
00043
00044 std::string CollectorDatafileDefaultImpl1::getDataFilename() {
00045     return _filename;
00046 }
00047
00048 void CollectorDatafileDefaultImpl1::setDataFilename(std::string filename) {
00049     _filename = filename;
00050 }
00051
00052 void CollectorDatafileDefaultImpl1::setAddValueHandler(CollectorAddValueHandler.addValueHandler) {
00053 }
00055
00056 void CollectorDatafileDefaultImpl1::setClearHandler(CollectorClearHandler.clearHandler) {
00057 }
00058 }
```

9.35 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`

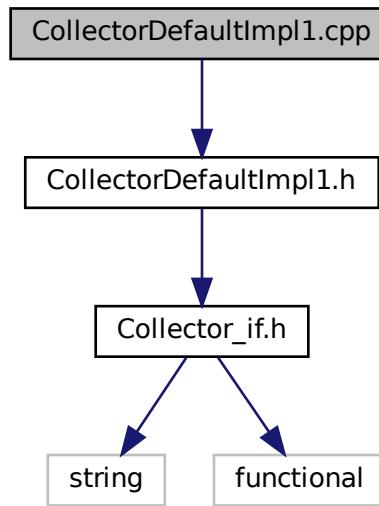
9.36 CollectorDatafileDefaultImpl1.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: CollectorDatafileDefaultImpl1.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 1 de Agosto de 2018, 20:58  
00012 */  
00013  
00014 #ifndef COLLECTORDATAFILEDEFAULTIMPL1_H  
00015 #define COLLECTORDATAFILEDEFAULTIMPL1_H  
00016  
00017 #include <string>  
00018  
00019 #include "CollectorDatafile_if.h"  
00020  
00021 class CollectorDatafileDefaultImpl1 : public CollectorDatafile_if {  
00022 public:  
00023     CollectorDatafileDefaultImpl1();  
00024     virtual ~CollectorDatafileDefaultImpl1() = default;  
00025 public: // inherited from Collector_if  
00026     void clear();  
00027     void addValue(double value);  
00028     double getLastValue();  
00029     unsigned long numElements();  
00030 public:  
00031     double getValue(unsigned int num);  
00032     double getNextValue();  
00033     void seekFirstValue();  
00034     std::string getDataFilename();  
00035     void setDataFilename(std::string filename);  
00036 public:  
00037     void setAddValueHandler(CollectorAddValueHandler.addValueHandler);  
00038     void setClearHandler(CollectorClearHandler.clearHandler);  
00039 private:  
00040     std::string _filename;  
00041 };  
00042  
00043 #endif /* COLLECTORDATAFILEDEFAULTIMPL1_H */  
00044
```

9.37 CollectorDefaultImpl1.cpp File Reference

```
#include "CollectorDefaultImpl.h"
```

Include dependency graph for CollectorDefaultImpl1.cpp:



9.38 CollectorDefaultImpl1.cpp

```

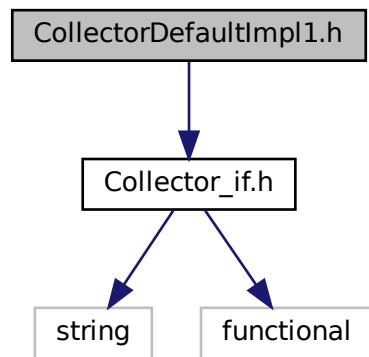
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: CollectorDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 1 de Agosto de 2018, 20:43
00012 */
00013
00014 #include "CollectorDefaultImpl1.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 CollectorDefaultImpl1::CollectorDefaultImpl1() {
00019 }
00020
00021 void CollectorDefaultImpl1::clear() {
00022     _numElements = 0;
00023     if (_clearHandler != nullptr) {
00024         _clearHandler();
00025     }
00026 }
00027
00028 void CollectorDefaultImpl1::addValue(double value) {
00029     _lastValue = value;
00030     _numElements++;
00031     if (_addValueHandler != nullptr) {
00032         _addValueHandler(value);
00033     }
00034 }
00035
00036 double CollectorDefaultImpl1::getLastValue() {
00037     return this->_lastValue;
00038 }
00039
00040 unsigned long CollectorDefaultImpl1::numElements() {
00041     return this->_numElements;
00042 }
00043
00044 void CollectorDefaultImpl1::setAddValueHandler(CollectorAddValueHandler addValueHandler) {
  
```

```

00045     _addValueHandler = addValueHandler;
00046 }
00047
00048 void CollectorDefaultImpl1::setClearHandler(CollectorClearHandler clearHandler) {
00049     _clearHandler = clearHandler;
00050 }
```

9.39 CollectorDefaultImpl1.h File Reference

#include "Collector_if.h"
Include dependency graph for CollectorDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CollectorDefaultImpl1](#)

9.40 CollectorDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   CollectorDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 1 de Agosto de 2018, 20:43
00012 */
00013
00014 #ifndef COLLECTORDEFAULTIMPL1_H
00015 #define COLLECTORDEFAULTIMPL1_H
00016 
```

```

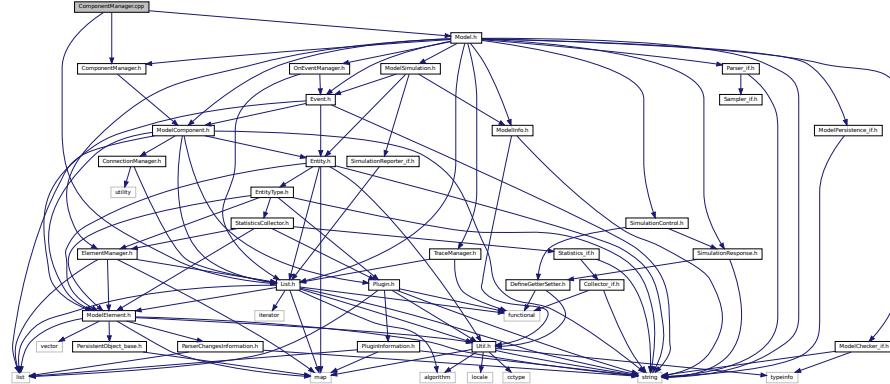
00017 #include "Collector_if.h"
00018 //namespace GenesysKernel {
00019
00020     class CollectorDefaultImpl1 : public Collector_if {
00021     public:
00022         CollectorDefaultImpl1();
00023         virtual ~CollectorDefaultImpl1() = default;
00024     public:
00025         void clear();
00026         void addValue(double value);
00027         double getLastValue();
00028         unsigned long numElements();
00029     public:
00030         void setAddValueHandler(CollectorAddValueHandler addValueHandler);
00031         void setClearHandler(CollectorClearHandler clearHandler);
00032     private:
00033         double _lastValue;
00034         unsigned long _numElements = 0;
00035         CollectorAddValueHandler _addValueHandler = nullptr;
00036         CollectorClearHandler _clearHandler = nullptr;
00037     };
00038 //namespace\\}
00039 /* COLLECTORDEFAULTIMPL1_H */
00040

```

9.41 ComponentManager.cpp File Reference

```
#include "ComponentManager.h"
#include "List.h"
#include "Model.h"
```

Include dependency graph for ComponentManager.cpp:



9.42 ComponentManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ComponentManager.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 28 de Maio de 2019, 10:41
00012 */
00013
00014 #include "ComponentManager.h"
00015 #include "List.h"
00016 #include "Model.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 ComponentManager::ComponentManager(Model* model) {
00021     _parentModel = model;

```

```

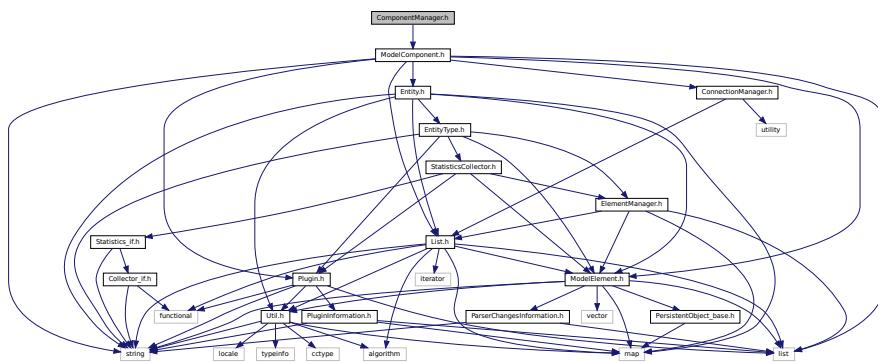
00022     _components = new List<ModelComponent*>();
00023     _components->setSortFunc([](const ModelComponent* a, const ModelComponent * b) {
00024         return a->getId() < b->getId();
00025     });
00026 }
00027
00028 bool ComponentManager::insert(ModelComponent* comp) {
00029     if (_components->find(comp) == _components->list()->end()) {
00030         _components->insert(comp);
00031         _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \\" + 
00032             comp->getName() + "\\" successfully inserted");
00033         _hasChanged = true;
00034     }
00035     _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \\" + 
00036         comp->getName() + "\\" could not be inserted");
00037     return false;
00038 }
00039 void ComponentManager::remove(ModelComponent* comp) {
00040     _components->remove(comp);
00041     _parentModel->getTracer()->trace(Util::TraceLevel::componentResult, "Component \\" + 
00042         comp->getName() + "\\" successfully removed");
00043     _hasChanged = true;
00044 }
00045 void ComponentManager::clear() {
00046     this->_components->clear();
00047     _hasChanged = true;
00048 }
00049
00050 //ModelComponent* ComponentManager::getComponent(Util::identification id) {
00051 //}
00052
00053 //ModelComponent* ComponentManager::getComponent(std::string name) {
00054 //}
00055
00056 unsigned int ComponentManager::getNumberOfComponents() {
00057     return _components->size();
00058 }
00059
00060 std::list<ModelComponent*>::iterator ComponentManager::begin() {
00061     return _components->list()->begin();
00062 }
00063
00064 std::list<ModelComponent*>::iterator ComponentManager::end() {
00065     return _components->list()->end();
00066 }
00067
00068 ModelComponent* ComponentManager::front() {
00069     return _components->front();
00070 }
00071
00072 ModelComponent* ComponentManager::next() {
00073     return _components->next();
00074 }
00075
00076 bool ComponentManager::hasChanged() const {
00077     return _hasChanged;
00078 }
00079
00080 void ComponentManager::setHasChanged(bool _hasChanged) {
00081     this->_hasChanged = _hasChanged;
00082 }

```

9.43 ComponentManager.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for ComponentManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ComponentManager

9.44 ComponentManager.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: ComponentManager.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 28 de Maio de 2019, 10:41  
00012 */  
00013  
00014 #ifndef COMPONENTMANAGER_H  
00015 #define COMPONENTMANAGER_H  
00016  
00017 #include "ModelComponent.h"  
00018  
00019 //namespace GenesysKernel {  
00020     //class Model;  
00021  
00022     class ComponentManager {  
00023     public:  
00024         ComponentManager(Model* model);  
00025         virtual ~ComponentManager() = default;  
00026     public:  
00027         bool insert(ModelComponent* comp);  
00028         void remove(ModelComponent* comp);  
00029         void clear();  
00030         //bool check(ModelComponent* comp, std::string expressionName, std::string* errorMessage);  
00031         //bool check(std::string compName, std::string expressionName, bool mandatory, std::string*  
errorMessage);  
00032     public:  
00033         //ModelComponent* getComponent(Util::identification id);  
00034         //ModelComponent* getComponent(std::string name);  
00035         unsigned int getNumberOfComponents();  
00036         std::list<ModelComponent*>::iterator begin();  
00037         std::list<ModelComponent*>::iterator end();  
00038         ModelComponent* front();
```

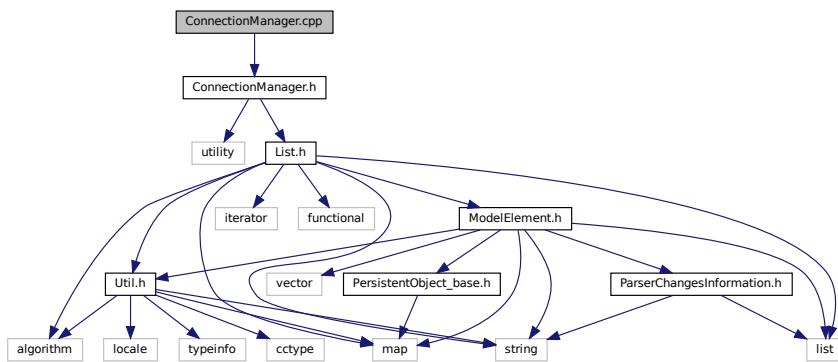
```

00039     ModelComponent* next();
00040     bool hasChanged() const;
00041     void setHasChanged(bool _hasChanged);
00042     //int getRankOf(std::string name);
00043 public:
00044 private:
00045     List<ModelComponent*>* _components;
00046     Model* _parentModel;
00047     bool _hasChanged = false;
00048     std::list<ModelComponent*>::iterator _componentIterator;
00049 };
00050 //namespace \\
00051 #endif /* COMPONENTMANAGER_H */
00052

```

9.45 ConnectionManager.cpp File Reference

#include "ConnectionManager.h"
Include dependency graph for ConnectionManager.cpp:



9.46 ConnectionManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ConnectionManager.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 1 de Julho de 2019, 18:39
00012 */
00013
00014 #include "ConnectionManager.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 ConnectionManager::ConnectionManager() {
00019 }
00020
00021 unsigned int ConnectionManager::size() {
00022     return _nextConnections->size();
00023 }
00024
00025 ModelComponent* ConnectionManager::front() {
00026     return _nextConnections->front()->first;
00027 }
00028
00029 ModelComponent* ConnectionManager::atRank(unsigned int rank) {
00030     return _nextConnections->getAtRank(rank)->first;
00031 }
00032
00033 Connection* ConnectionManager::getFrontConnection() {

```

```

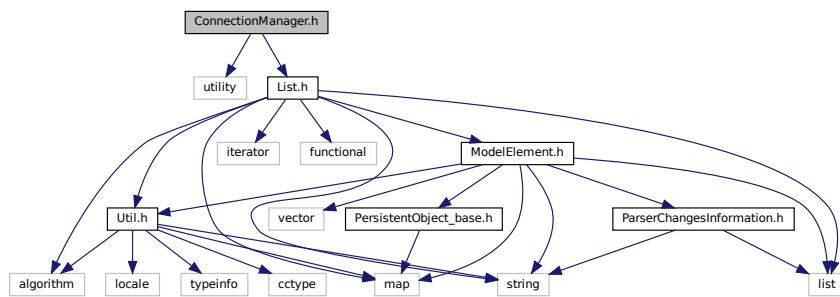
00034     return _nextConnections->front();
00035 }
00036
00037 Connection* ConnectionManager::getConnectionAtRank(unsigned int rank) {
00038     return _nextConnections->getAtRank(rank);
00039 }
00040
00041 void ConnectionManager::insert(ModelComponent* component, unsigned int inputNumber) {
00042     _nextConnections->insert(new Connection(component, inputNumber));
00043     _currentOutputConnections++;
00044 }
00045
00046 void ConnectionManager::insertAtRank(unsigned int rank, Connection* connection) {
00047     _nextConnections->setAtRank(rank, connection); // \TODO: it does not work if there is less than
00048     // rank connections. Model designer responsibility?
00049     _currentOutputConnections++;
00050 }
00051 std::list<Connection*>* ConnectionManager::list() const {
00052     return _nextConnections->list();
00053 }
00054
00055 //void ConnectionManager::setCurrentOutputConnections(unsigned int _currentOutputConnections) {
00056 //    this->_currentOutputConnections = _currentOutputConnections;
00057 //}
00058
00059 unsigned int ConnectionManager::getCurrentOutputConnections() const {
00060     return _currentOutputConnections;
00061 }
00062
00063 void ConnectionManager::setMaxOutputConnections(unsigned int _maxOutputConnections) {
00064     this->_maxOutputConnections = _maxOutputConnections;
00065 }
00066
00067 unsigned int ConnectionManager::getMaxOutputConnections() const {
00068     return _maxOutputConnections;
00069 }
00070
00071 void ConnectionManager::setMinOutputConnections(unsigned int _minOutputConnections) {
00072     this->_minOutputConnections = _minOutputConnections;
00073 }
00074
00075 unsigned int ConnectionManager::getMinOutputConnections() const {
00076     return _minOutputConnections;
00077 }
00078
00079 //void ConnectionManager::setCurrentInputConnections(unsigned int _currentInputConnections) {
00080 //    this->_currentInputConnections = _currentInputConnections;
00081 //}
00082
00083 unsigned int ConnectionManager::getCurrentInputConnections() const {
00084     return _currentInputConnections;
00085 }
00086
00087 void ConnectionManager::setMaxInputConnections(unsigned int _maxInputConnections) {
00088     this->_maxInputConnections = _maxInputConnections;
00089 }
00090
00091 unsigned int ConnectionManager::getMaxInputConnections() const {
00092     return _maxInputConnections;
00093 }
00094
00095 void ConnectionManager::setMinInputConnections(unsigned int _minInputConnections) {
00096     this->_minInputConnections = _minInputConnections;
00097 }
00098
00099 unsigned int ConnectionManager::getMinInputConnections() const {
00100     return _minInputConnections;
00101 }

```

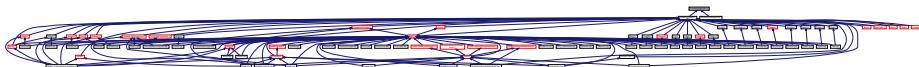
9.47 ConnectionManager.h File Reference

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

Include dependency graph for ConnectionManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ConnectionManager](#)

TypeDefs

- `typedef std::pair< ModelComponent *, unsigned int > Connection`

9.47.1 Typedef Documentation

9.47.1.1 Connection `typedef std::pair<ModelComponent*, unsigned int> Connection`

Definition at line 22 of file [ConnectionManager.h](#).

9.48 ConnectionManager.h

```

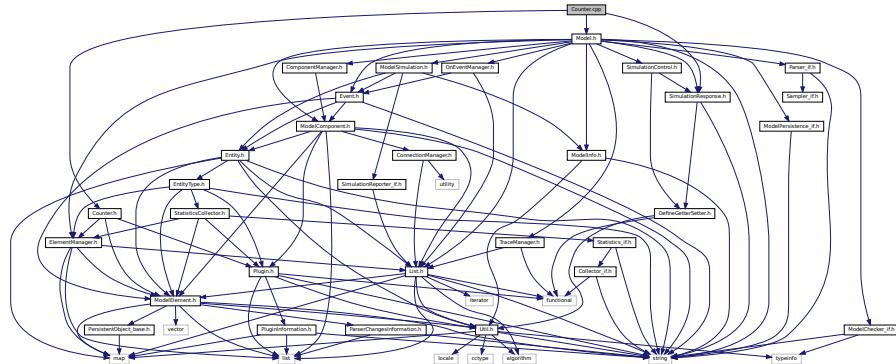
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ConnectionManager.h
00009 * Author: rlcancian
00010 *
00011 * Created on 1 de Julho de 2019, 18:39
00012 */
00013
00014 #ifndef CONNECTIONMANAGER_H
00015 #define CONNECTIONMANAGER_H
00016
  
```

```
00017 #include <utility>
00018 #include "List.h"
00019
00020 //namespace GenesysKernel {
00021     class ModelComponent;
00023
00024     typedef std::pair<ModelComponent*, unsigned int> Connection;
00025
00026     class ConnectionManager {
00027     public:
00028         ConnectionManager();
00029         virtual ~ConnectionManager() = default;
00030     public:
00031         unsigned int size();
00032         ModelComponent* front();
00033         ModelComponent* atRank(unsigned int rank);
00034         Connection* getFrontConnection();
00035         Connection* getConnectionAtRank(unsigned int rank);
00036         void insert(ModelComponent* component, unsigned int inputNumber = 0);
00037         void insertAtRank(unsigned int rank, Connection* connection);
00038         //void insert(ModelComponent* component, unsigned int inputNumber = 0);
00039         std::list<Connection*>* list() const;
00040         //void setCurrentOutputConnections(unsigned int _currentOutputConnections);
00041         unsigned int getCurrentOutputConnections() const;
00042         void setMaxOutputConnections(unsigned int _maxOutputConnections);
00043         unsigned int getMaxOutputConnections() const;
00044         void setMinOutputConnections(unsigned int _minOutputConnections);
00045         unsigned int getMinOutputConnections() const;
00046         //void setCurrentInputConnections(unsigned int _currentInputConnections);
00047         unsigned int getCurrentInputConnections() const;
00048         void setMaxInputConnections(unsigned int _maxInputConnections);
00049         unsigned int getMaxInputConnections() const;
00050         void setMinInputConnections(unsigned int _minInputConnections);
00051         unsigned int getMinInputConnections() const;
00052     private:
00053         List<Connection*> _nextConnections = new List<Connection*>();
00054         unsigned int _minInputConnections = 1;
00055         unsigned int _maxInputConnections = 1;
00056         unsigned int _currentInputConnections = 1;
00057         unsigned int _minOutputConnections = 1;
00058         unsigned int _maxOutputConnections = 1;
00059         unsigned int _currentOutputConnections = 1;
00060     };
00061 //namespace\\}
00062 #endif /* CONNECTIONMANAGER_H */
```

9.49 Counter.cpp File Reference

```
#include "Counter.h"
#include "SimulationResponse.h"
#include "Model.h"
Include dependency graph for Counter.cpp:
```

Include dependency graph for Counter.cpp:



9.50 Counter.cpp

00001 /*

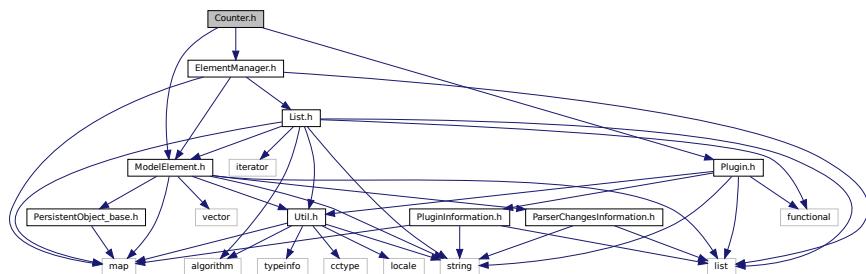
```

00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: CounterDefaultImpl.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 29 de Maio de 2019, 11:24
00012 */
00013
00014 #include "Counter.h"
00015 #include "SimulationResponse.h"
00016 #include "Model.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 Counter::Counter(Model* model, std::string name, ModelElement* parent) : ModelElement(model,
00021     Util::TypeOf<Counter>(), name) {
00022     _parent = parent;
00023     GetterMember getterMember = DefineGetterMember<Counter>(this, &Counter::getCountValue);
00024     //std::string parentName = "";
00025     //if (_parent != nullptr)
00026     //    parentName = _parent->name();
00027     SimulationResponse* resp = new SimulationResponse(Util::TypeOf<Counter>(), /*parentName + ":" +*/
00028         _name, getterMember);
00029     _parentModel->getResponses()->insert(resp);
00030 }
00031
00032 std::string Counter::show() {
00033     return ModelElement::show() +
00034         ", count=" + std::to_string(this->_count);
00035 }
00036
00037 void
00038 Counter::clear() {
00039     _count = 0;
00040 }
00041
00042 void Counter::incCountValue(int value) {
00043     _count += value;
00044 }
00045
00046 unsigned long Counter::getCountValue() const {
00047     return _count;
00048 }
00049
00050 ModelElement* Counter::getParent() const {
00051     return _parent;
00052 }
00053
00054 PluginInformation* Counter::GetPluginInformation() {
00055     PluginInformation* info = new PluginInformation(Util::TypeOf<Counter>(), &Counter::LoadInstance);
00056     info->setGenerateReport(true);
00057     return info;
00058 }
00059
00060 ModelElement* Counter::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00061     Counter* newElement = new Counter(model);
00062     try {
00063         newElement->_loadInstance(fields);
00064     } catch (const std::exception& e) {
00065     }
00066     return newElement;
00067 }
00068
00069
00070 bool Counter::_loadInstance(std::map<std::string, std::string>* fields) {
00071     return ModelElement::_loadInstance(fields);
00072 }
00073
00074 std::map<std::string, std::string>* Counter::_saveInstance() {
00075     return ModelElement::_saveInstance();
00076 }
00077 //std::list<std::map<std::string, std::string>*>* _saveInstance(std::string type){}
00078
00079 bool Counter::_check(std::string* errorMessage) {
00080     return true;
00081 }
00082

```

9.51 Counter.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
Include dependency graph for Counter.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Counter](#)

9.52 Counter.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: CounterDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 29 de Maio de 2019, 11:24
00012 */
00013
00014 #ifndef COUNTERDEFAULTIMPL1_H
00015 #define COUNTERDEFAULTIMPL1_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020 //namespace GenesysKernel {
00021
00022     class Counter : public ModelElement {
00023     public:
00024         Counter(Model* model, std::string name = "", ModelElement* parent = nullptr);
00025         virtual ~Counter() = default;
00026     public:
00027         virtual std::string show();
00028     public:
00029         static PluginInformation* GetPluginInformation();
00030         static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00031     public:
00032         void clear();
00033         void incCountValue(int value = 1);
00034         unsigned long getCountValue() const;
```

```

00038     ModelElement* getParent() const;
00039 protected: // from ModelElement
00040     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00041     virtual std::map<std::string, std::string>* _saveInstance();
00042     virtual bool _check(std::string* errorMessage);
00043 private:
00044     ModelElement* _parent;
00045     unsigned long _count = 0;
00046 };
00047 //namespace\\}
00048 /* COUNTERDEFAULTIMPL1_H */
00049

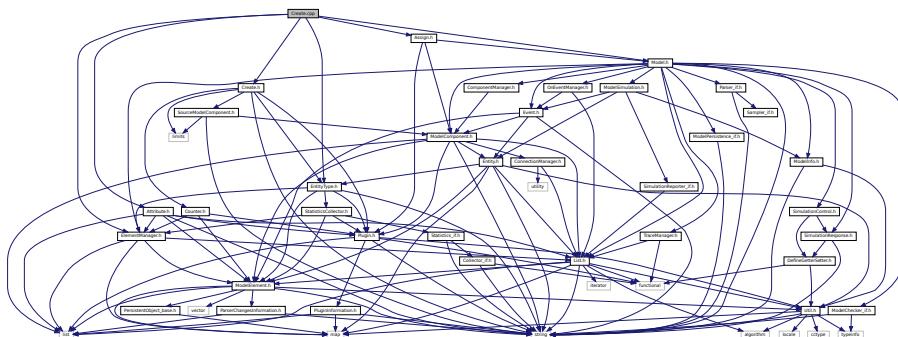
```

9.53 Create.cpp File Reference

```

#include "Create.h"
#include "Model.h"
#include "EntityType.h"
#include "ElementManager.h"
#include "Attribute.h"
#include "Assign.h"
Include dependency graph for Create.cpp:

```



9.54 Create.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Create.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 20:12
00012 */
00013
00014 #include "Create.h"
00015 #include "Model.h"
00016 #include "EntityType.h"
00017 #include "ElementManager.h"
00018 #include "Attribute.h"
00019 #include "Assign.h"
00020
00021 Create::Create(Model* model, std::string name) : SourceModelComponent(model, Util::TypeOf<Create>(),
00022     name) {
00023     //_numberOut = new Counter(_parentModel, _name + "." + "Count_number_in", this);
00024     // \todo Check if element has already been inserted and this is not needed:
00025     _parentModel->elements()->insert(_numberOut);
00026     _connections->setMinInputConnections(0);
00027     _connections->setMaxInputConnections(0);
00028     GetterMember getter = DefineGetterMember<SourceModelComponent>(this,
00029         &Create::getEntitiesPerCreation);
00030     SetterMember setter = DefineSetterMember<SourceModelComponent>(this,
00031         &Create::setEntitiesPerCreation);

```

```

00028     model->getControls()->insert(new SimulationControl(Util::TypeOf<Create>(), _name +
00029     ".EntitiesPerCreation", getter, setter));
00030     /* \todo:
00031     model->getControls()->insert(new SimulationControl(Util::TypeOf<Create>(), "Time Between
00032     Creations",
00033         DefineGetterMember<SourceModelComponent>(this, &Create::getTimeBetweenCreationsExpression),
00034         DefineSetterMember<SourceModelComponent>(this, &Create::setTimeBetweenCreationsExpression))
00035     );
00036 */
00037 std::string Create::show() {
00038     return SourceModelComponent::show();
00039 }
00040
00041 void Create::_execute(Entity* entity) {
00042     double tnow = _parentModel->getSimulation()->getSimulatedTime();
00043     entity->setAttributeValue("Entity.ArrivalTime", tnow); // 
00044     ->find("Entity.ArrivalTime")->second->setValue(tnow);
00045     //entity->setAttributeValue("Entity.Picture", 1); // 
00046     ->find("Entity.ArrivalTime")->second->setValue(tnow);
00047     double timeBetweenCreations, timeScale, newArrivalTime;
00048     unsigned int _maxCreations = _parentModel->parseExpression(this->_maxCreationsExpression);
00049     for (unsigned int i = 0; i<this->_entitiesPerCreation; i++) {
00050         if (_entitiesCreatedSoFar < _maxCreations) {
00051             _entitiesCreatedSoFar++;
00052             Entity* newEntity = new Entity(_parentModel);
00053             newEntity->setEntityType(entity->getEntityType());
00054             // _parentModel->elements()->insert(newEntity); // ->getEntities()->insert(newEntity);
00055             timeBetweenCreations =
00056             _parentModel->parseExpression(this->_timeBetweenCreationsExpression);
00057             timeScale = Util::TimeUnitConvert(this->_timeBetweenCreationsTimeUnit,
00058             _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00059             newArrivalTime = tnow + timeBetweenCreations*timeScale;
00060             Event* newEvent = new Event(newArrivalTime, newEntity, this);
00061             _parentModel->getFutureEvents()->insert(newEvent);
00062             _parentModel->getTracer()->trace("Arrival of entity " +
00063             std::to_string(newEntity->entityNumber()) + " schedule for time " + std::to_string(newArrivalTime));
00064             // _model->getTrace()->trace("Arrival of entity "+std::to_string(entity->getId()) + "
00065             schedule for time " +std::to_string(newArrivalTime));
00066         }
00067     if (_reportStatistics)
00068         _numberOut->incCountValue();
00069     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00070 }
00071
00072 PluginInformation* Create::GetPluginInformation() {
00073     PluginInformation* info = new PluginInformation(Util::TypeOf<Create>(), &Create::LoadInstance);
00074     info->setSource(true);
00075     info->insertDynamicLibFileDependence("attribute.so");
00076     info->insertDynamicLibFileDependence("entitytype.so");
00077     info->insertDynamicLibFileDependence("statisticscollector.so");
00078     return info;
00079 }
00080
00081 ModelComponent* Create::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00082     Create* newComponent = new Create(model);
00083     try {
00084         newComponent->_loadInstance(fields);
00085     } catch (const std::exception& e) {
00086     }
00087     return newComponent;
00088 }
00089
00090 void Create::_initBetweenReplications() {
00091     SourceModelComponent::_initBetweenReplications();
00092 }
00093
00094 std::map<std::string, std::string>* Create::_saveInstance() {
00095     std::map<std::string, std::string>* fields = SourceModelComponent::_saveInstance();
00096     return fields;
00097 }
00098
00099 bool Create::_check(std::string* errorMessage) {
00100     bool resultAll = SourceModelComponent::_check(errorMessage);
00101     return resultAll;
00102 }
00103
00104 void Create::_createInternalElements() {
00105     if (_reportStatistics && _numberOut == nullptr) {
00106         _numberOut = new Counter(_parentModel, _name + "." + "CountNumberIn", this);

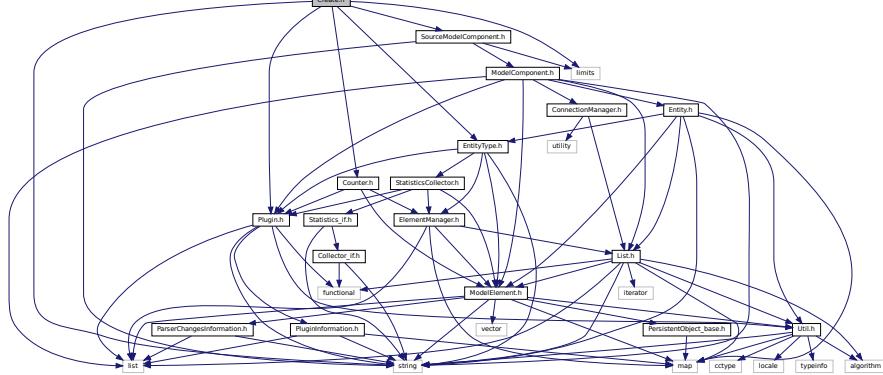
```

```
00107     _childrenElements->insert({"CountNumberIn", _numberOut});
00108     // \todo _childrenElements->insert("Count_number_in", _numberOut);
00109 } else if (!reportStatistics && _numberOut != nullptr) {
00110     this->_removeChildrenElements();
00111     // \todo _childrenElements->remove("Count_number_in");
00112     //_numberOut->~Counter();
00113     _numberOut = nullptr;
00114 }
00115 }
```

9.55 Create.h File Reference

```
#include <string>
#include <limits>
#include "SourceModelComponent.h"
#include "EntityType.h"
#include "Counter.h"
#include "Plugin.h"
Include dependency graph for Create.h:
```

Include dependency graph creation



This graph shows which files directly or indirectly include this file:



Classes

- class Create

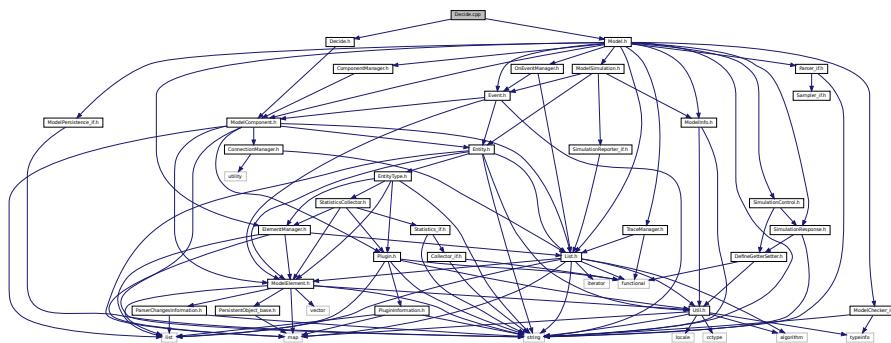
9.56 Create.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Create.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 20:12
00012 */
```

```
00013
00014 #ifndef CREATE_H
00015 #define CREATE_H
00016
00017 #include <string>
00018 #include <limits>
00019 #include "SourceModelComponent.h"
00020 #include "EntityType.h"
00021 #include "Counter.h"
00022 #include "Plugin.h"
00023
00067 class Create : public SourceModelComponent {
00068 public:
00069     Create(Model* model, std::string name = "");
00070     virtual ~Create() = default;
00071 public:
00072     virtual std::string show();
00073 public:
00074     static PluginInformation* GetPluginInformation();
00075     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00076 protected:
00077     virtual void _execute(Entity* entity);
00078     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00079     virtual void _initBetweenReplications();
00080     virtual std::map<std::string, std::string>* _saveInstance();
00081     virtual bool _check(std::string* errorMessage);
00082     virtual void _createInternalElements();
00083 private: // children elements
00084     Counter* _numberOut = nullptr;
00085 };
00086
00087 #endif /* CREATE_H */
```

9.57 Decide.cpp File Reference

```
#include "Decide.h"
#include "Model.h"
Include dependency graph for Decide.cpp:
```



9.58 Decide.cpp

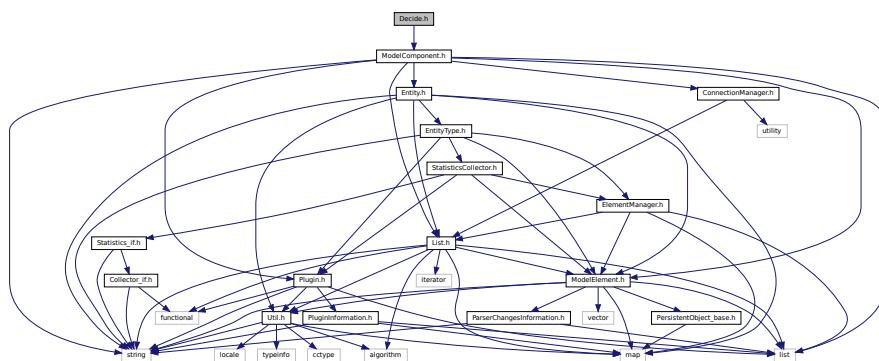
```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Decide.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 20:39
00012 */
00013
00014 #include "Decide.h"
00015 #include "Model.h"
00016
00017 Decide::Decide(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Decide>(), name) {
```

```

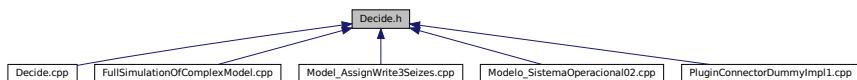
00018 }
00019
00020 List<std::string>* Decide::getConditions() const {
00021     return _conditions;
00022 }
00023
00024 std::string Decide::show() {
00025     return ModelComponent::show() + "";
00026 }
00027
00028 void Decide::_execute(Entity* entity) {
00029     double value;
00030     unsigned short i = 0;
00031     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->list()->end(); it++) {
00032         value = _parentModel->parseExpression((*it));
00033         _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00034             entity, this, std::to_string(i + 1) + "th condition evaluated to " + std::to_string(value) + " // "
00035             + (*it));
00036         if (value) {
00037             _parentModel->sendEntityToComponent(entity,
00038                 this->getNextComponents()->getConnectionAtRank(i), 0.0);
00039         }
00040         _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00041             entity, this, "No condition has been evaluated true");
00042         _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getConnectionAtRank(i),
00043             0.0);
00044 }
00045
00046 void Decide::_initBetweenReplications() {
00047 }
00048
00049 bool Decide::_loadInstance(std::map<std::string, std::string>* fields) {
00050     bool res = ModelComponent::_loadInstance(fields);
00051     if (res) {
00052         unsigned int nv = std::stoi((*(fields->find("conditions"))).second);
00053         for (unsigned int i = 0; i < nv; i++) {
00054             this->_conditions->insert((*(fields->find("condition" + std::to_string(i)))).second);
00055         }
00056     }
00057     return res;
00058 }
00059
00060 std::map<std::string, std::string>* Decide::_saveInstance() {
00061     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00062     //Util::TypeOf<Decide>());
00063     unsigned short i = 0;
00064     fields->emplace("conditions", std::to_string(_conditions->size()));
00065     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->list()->end(); it++) {
00066         fields->emplace("condition" + std::to_string(i++), "\\" + (*it) + "\\");
00067     }
00068     return fields;
00069 }
00070
00071 bool Decide::_check(std::string* errorMessage) {
00072     bool allResult = true;
00073     std::string condition;
00074     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->list()->end(); it++) {
00075         condition = (*it);
00076         allResult &= _parentModel->checkExpression(condition, "condition", errorMessage);
00077     }
00078     return allResult;
00079 }
00080
00081 PluginInformation* Decide::GetPluginInformation() {
00082     PluginInformation* info = new PluginInformation(Util::TypeOf<Decide>(), &Decide::LoadInstance);
00083     return info;
00084 }
00085
00086 ModelComponent* Decide::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00087     Decide* newComponent = new Decide(model);
00088     try {
00089         newComponent->_loadInstance(fields);
00090     } catch (const std::exception& e) {
00091     }
00092     return newComponent;
00093 }
```

9.59 Decide.h File Reference

```
#include "ModelComponent.h"
Include dependency graph for Decide.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Decide

9.60 Decide.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Decide.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 20:39
00012 */
00013
00014 #ifndef DECIDE_H
00015 #define DECIDE_H
00016
00017 #include "ModelComponent.h"
00018
00073 class Decide : public ModelComponent {
00074 public:
00075     Decide(Model* model, std::string name = "");
00076     virtual ~Decide() = default;
00077 public:
00078     List<std::string>* getConditions() const;
00079
00080 public:
00081     virtual std::string show();
00082 public:
00083     static PluginInformation* GetPluginInformation();
00084     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
```

```

00085 protected:
00086     virtual void _execute(Entity* entity);
00087     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00088     virtual void _initBetweenReplications();
00089     virtual std::map<std::string, std::string>* _saveInstance();
00090     virtual bool _check(std::string* errorMessage);
00091 private:
00092     List<std::string>* _conditions = new List<std::string>();
00093 private:
00094
00095 };
00096
00097 #endif /* DECIDE_H */
00098

```

9.61 DefaultTraceAndEventHandlers.h File Reference

9.62 DefaultTraceAndEventHandlers.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: DefaultTraceAndEventHandlers.h
00009  * Author: rlcancian
00010 *
00011  * Created on 3 de Setembro de 2019, 16:12
00012 */
00013
00014 #ifndef DEFAULTTRACEANDEVENTHANDLERS_H
00015 #define DEFAULTTRACEANDEVENTHANDLERS_H
00016
00017 /*
00018 #include "TraceManager.h"
00019 #include "OnEventManager.h"
00020 #include <iostream>
00021
00022 // default Trace Handlers
00023 void traceHandlerFunction(TraceEvent e) {
00024     std::cout << e.getText() << std::endl;
00025 }
00026
00027 void traceSimulationHandlerFunction(TraceSimulationEvent e) {
00028     std::cout << e.getText() << std::endl;
00029 }
00030
00031 void onSimulationStartHandlerFunction(SimulationEvent* re) {
00032     //std::cout << "(Handler) Simulation is starting" << std::endl;
00033 }
00034
00035
00036 // default Event Handlers
00037 void onReplicationStartHandlerFunction(SimulationEvent* re) {
00038     // std::cout << "(Handler) Replication " << re->getReplicationNumber() << " starting." << std::endl;
00039 }
00040
00041 void onProcessEventHandlerFunction(SimulationEvent* re) {
00042     //std::cout << "(Handler) Processing event " << re->getEventProcessed()->show() << std::endl;
00043 }
00044
00045 void onReplicationEndHandlerFunction(SimulationEvent* re) {
00046     //std::cout << "(Handler) Replication " << re->getReplicationNumber() << " ending." << std::endl;
00047 }
00048
00049 void onEntityRemoveHandlerFunction(SimulationEvent* re) {
00050     //std::cout << "(Handler) Entity " << re->getEventProcessed()->getEntity() << " was removed." <<
00051     std::endl;
00052 }
00053 #endif /* DEFAULTTRACEANDEVENTHANDLERS_H */
00054

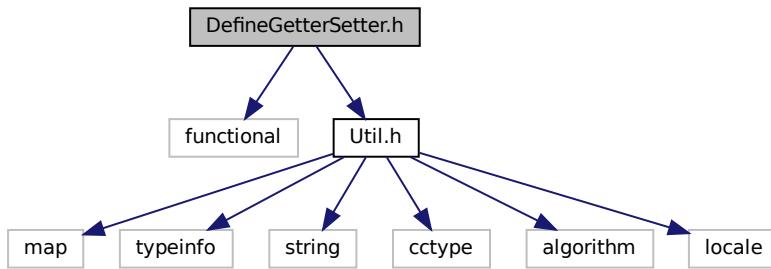
```

9.63 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`
- `typedef std::function< std::string() > GetterMemberString`
- `typedef std::function< void(std::string) > SetterMemberString`

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, unsigned long(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(unsigned long))`
- `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) const)`

- template<typename Class >
`GetterMemberString DefineGetterMember` (Class *object, std::string(Class::*function)() const)
- template<typename Class >
`SetterMemberString DefineSetterMember` (Class *object, void(Class::*function)(std::string) const)
- template<typename Class >
`GetterMember DefineGetterMember` (Class *object, Util::TimeUnit(Class::*function)() const)
- template<typename Class >
`SetterMember DefineSetterMember` (Class *object, void(Class::*function)(Util::TimeUnit))

9.63.1 Typedef Documentation

9.63.1.1 **GetterMember** `typedef std::function<double() > GetterMember`

Definition at line 20 of file [DefineGetterSetter.h](#).

9.63.1.2 **GetterMemberString** `typedef std::function<std::string() > GetterMemberString`

Definition at line 23 of file [DefineGetterSetter.h](#).

9.63.1.3 **SetterMember** `typedef std::function<void(double) > SetterMember`

Definition at line 21 of file [DefineGetterSetter.h](#).

9.63.1.4 **SetterMemberString** `typedef std::function<void(std::string) > SetterMemberString`

Definition at line 24 of file [DefineGetterSetter.h](#).

9.63.2 Function Documentation

9.63.2.1 **DefineGetterMember()** [1/7] `template<typename Class >` `GetterMember DefineGetterMember (` `Class * object,` `bool(Class::*)() const function)`

Definition at line 71 of file [DefineGetterSetter.h](#).

```
00071
00072     return std::bind(function, object);
00073 }
```

9.63.2.2 DefineGetterMember() [2/7] template<typename Class >
GetterMember DefineGetterMember (
 Class * object,
 double(Class::*)() function)

Definition at line 29 of file DefineGetterSetter.h.

```
00029  
00030     return std::bind(function, object);  
00031 }
```

9.63.2.3 DefineGetterMember() [3/7] template<typename Class >
GetterMember DefineGetterMember (
 Class * object,
 std::string(Class::*)() const function)

Definition at line 83 of file DefineGetterSetter.h.

```
00083  
00084     return std::bind(function, object);  
00085 }
```

9.63.2.4 DefineGetterMember() [4/7] template<typename Class >
GetterMemberString DefineGetterMember (
 Class * object,
 std::string(Class::*)() const function)

Definition at line 93 of file DefineGetterSetter.h.

```
00093  
00094     return std::bind(function, object);  
00095 }
```

9.63.2.5 DefineGetterMember() [5/7] template<typename Class >
GetterMember DefineGetterMember (
 Class * object,
 unsigned int(Class::*)() const function)

Definition at line 46 of file DefineGetterSetter.h.

```
00046  
00047     return std::bind(function, object);  
00048 }
```

9.63.2.6 DefineGetterMember() [6/7] template<typename Class >
GetterMember DefineGetterMember (
 Class * object,
 unsigned long(Class::*)() const function)

Definition at line 59 of file DefineGetterSetter.h.

```
00059  
00060     return std::bind(function, object);  
00061 }
```

9.63.2.7 DefineGetterMember() [7/7] template<typename Class >

```
GetterMember DefineGetterMember (
    Class * object,
    Util::TimeUnit(Class::*)() const function )
```

Definition at line 106 of file [DefineGetterSetter.h](#).

```
00106
00107     return std::bind(function, object);
00108 }
```

9.63.2.8 DefineSetterMember() [1/7] template<typename Class >

```
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*)(bool) function )
```

Definition at line 76 of file [DefineGetterSetter.h](#).

```
00076
00077     return std::bind(function, object, std::placeholders::_1);
00078 }
```

9.63.2.9 DefineSetterMember() [2/7] template<typename Class >

```
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*)(double) function )
```

Definition at line 39 of file [DefineGetterSetter.h](#).

```
00039
00040     return std::bind(function, object, std::placeholders::_1);
00041 }
```

9.63.2.10 DefineSetterMember() [3/7] template<typename Class >

```
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*)(std::string) const function )
```

Definition at line 88 of file [DefineGetterSetter.h](#).

```
00088
00089     return std::bind(function, object, std::placeholders::_1);
00090 }
```

9.63.2.11 DefineSetterMember() [4/7] template<typename Class >

```
SetterMemberString DefineSetterMember (
    Class * object,
    void(Class::*)(std::string) const function )
```

Definition at line 98 of file [DefineGetterSetter.h](#).

```
00098
00099     return std::bind(function, object, std::placeholders::_1);
00100 }
```

9.63.2.12 DefineSetterMember() [5/7] template<typename Class >
SetterMember DefineSetterMember (Class * object,
void(Class::*)(unsigned int) function)

Definition at line 51 of file DefineGetterSetter.h.

```
00051
00052     return std::bind(function, object, std::placeholders::_1);
00053 }
```

9.63.2.13 DefineSetterMember() [6/7] template<typename Class >

SetterMember DefineSetterMember (Class * object,
void(Class::*)(unsigned long) function)

Definition at line 64 of file DefineGetterSetter.h.

```
00064
00065     return std::bind(function, object, std::placeholders::_1);
00066 }
```

9.63.2.14 DefineSetterMember() [7/7] template<typename Class >

SetterMember DefineSetterMember (Class * object,
void(Class::*)(Util::TimeUnit) function)

Definition at line 111 of file DefineGetterSetter.h.

```
00111
00112     return std::bind(function, object, std::placeholders::_1);
00113 }
```

9.64 DefineGetterSetter.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: DefineGetterSetter.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 5 de Agosto de 2018, 13:52
00012 */
00013
00014 #ifndef DEFINEGETTERSETTER_H
00015 #define DEFINEGETTERSETTER_H
00016
00017 #include <functional>
00018 #include "Util.h"
00019
00020 typedef std::function<double()> GetterMember;
00021 typedef std::function<void(double)> SetterMember;
00022
00023 typedef std::function<std::string()> GetterMemberString;
00024 typedef std::function<void(std::string)> SetterMemberString;
00025
00026 // double
00027
00028 template<typename Class>
00029 GetterMember DefineGetterMember(Class * object, double (Class::*function)()) {
00030     return std::bind(function, object);
00031 }
00032
00033 template<typename Class>
```

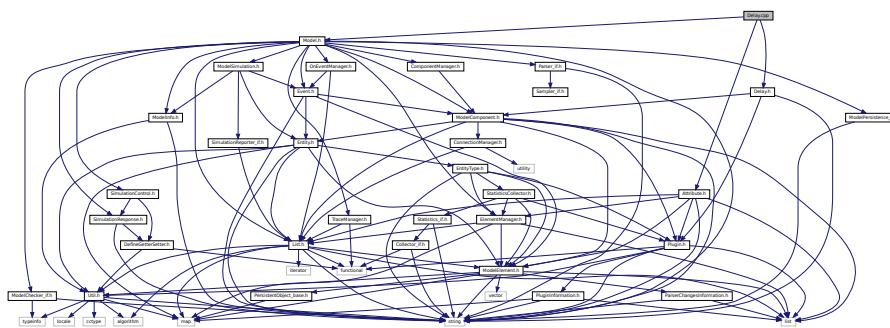
```

00034 GetterMember DefineGetterMember(Class * object, double (Class::*function)() const) {
00035     return std::bind(function, object);
00036 }
00037
00038 template<typename Class>
00039 SetterMember DefineSetterMember(Class * object, void (Class::*function)(double)) {
00040     return std::bind(function, object, std::placeholders::_1);
00041 }
00042
00043 // unsigned int
00044
00045 template<typename Class>
00046 GetterMember DefineGetterMember(Class * object, unsigned int (Class::*function)() const) {
00047     return std::bind(function, object);
00048 }
00049
00050 template<typename Class>
00051 SetterMember DefineSetterMember(Class * object, void (Class::*function)(unsigned int)) {
00052     return std::bind(function, object, std::placeholders::_1);
00053 }
00054
00055
00056 // unsigned long
00057
00058 template<typename Class>
00059 GetterMember DefineGetterMember(Class * object, unsigned long (Class::*function)() const) {
00060     return std::bind(function, object);
00061 }
00062
00063 template<typename Class>
00064 SetterMember DefineSetterMember(Class * object, void (Class::*function)(unsigned long)) {
00065     return std::bind(function, object, std::placeholders::_1);
00066 }
00067
00068 // bool
00069
00070 template<typename Class>
00071 GetterMember DefineGetterMember(Class * object, bool (Class::*function)() const) {
00072     return std::bind(function, object);
00073 }
00074
00075 template<typename Class>
00076 SetterMember DefineSetterMember(Class * object, void (Class::*function)(bool)) {
00077     return std::bind(function, object, std::placeholders::_1);
00078 }
00079
00080 //std::string
00081
00082 template<typename Class>
00083 GetterMember DefineGetterMember(Class * object, std::string(Class::*function)() const) {
00084     return std::bind(function, object);
00085 }
00086
00087 template<typename Class>
00088 SetterMember DefineSetterMember(Class * object, void (Class::*function)(std::string) const) {
00089     return std::bind(function, object, std::placeholders::_1);
00090 }
00091
00092 template<typename Class>
00093 GetterMemberString DefineGetterMember(Class * object, std::string(Class::*function)() const) {
00094     return std::bind(function, object);
00095 }
00096
00097 template<typename Class>
00098 SetterMemberString DefineSetterMember(Class * object, void (Class::*function)(std::string) const) {
00099     return std::bind(function, object, std::placeholders::_1);
00100 }
00101
00102
00103 // Util::TimeUnit
00104
00105 template<typename Class>
00106 GetterMember DefineGetterMember(Class * object, /*GenesysKernel::*/Util::TimeUnit(Class::*function)()
00107     const) {
00108     return std::bind(function, object);
00109 }
00110
00111 template<typename Class>
00112 SetterMember DefineSetterMember(Class * object, void
00113     (Class::*function) /*(/GenesysKernel::*/Util::TimeUnit)) {
00114     return std::bind(function, object, std::placeholders::_1);
00115 }
00116
00117 #endif /* DEFINEGETTERSETTER_H */

```

9.65 Delay.cpp File Reference

```
#include "Delay.h"
#include "Model.h"
#include "Attribute.h"
Include dependency graph for Delay.cpp:
```



9.66 Delay.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Delay.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:49
00012 */
00013
00014 #include "Delay.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 Delay::Delay(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Delay>(), name) {
00019
00020     GetterMember getter = DefineGetterMember<Delay>(this, &Delay::delay);
00021     SetterMember setter = DefineSetterMember<Delay>(this, &Delay::setDelay);
00022     model->getControls()->insert(new SimulationControl(Util::TypeOf<Delay>(), _name + ".Delay",
00023                                     getter, setter));
00024
00025     //GetterMember getter2 = DefineGetterMember<Delay>(this, &Delay::delayTimeUnit);
00026     //SetterMember setter2 = DefineSetterMember<Delay>(this, &Delay::setDelayTimeUnit);
00027     //model->controls()->insert(new SimulationControl(Util::TypeOf<Delay>(), _name + ".DelayTimeUnit",
00028                                     getter2, setter2));
00029 }
00030
00031 void Delay::setDelay(double delay) {
00032     _delayExpression = std::to_string(delay);
00033 }
00034
00035 double Delay::delay() const {
00036     return _parentModel->parseExpression(_delayExpression);
00037 }
00038
00039
00040 std::string Delay::show() {
00041     return ModelComponent::show() +
00042             ",delayExpression=" + this->_delayExpression +
00043             ",timeUnit=" + std::to_string(static_cast<int> (this->_delayTimeUnit));
00044 }
00045
00046 ModelComponent* Delay::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00047     Delay* newComponent = new Delay(model);
00048     try {
00049         newComponent->_loadInstance(fields);
00050     } catch (const std::exception& e) {
```

```

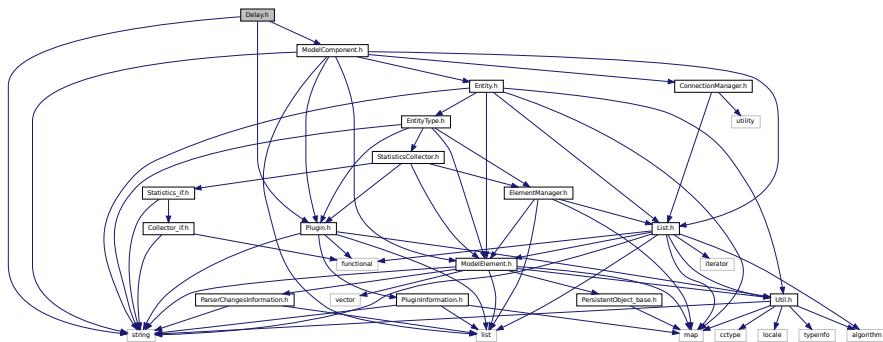
00051     }
00052     return newComponent;
00053 }
00055
00056 void Delay::setDelayExpression(std::string _delayExpression) {
00057     this->_delayExpression = _delayExpression;
00058 }
00059
00060 std::string Delay::delayExpression() const {
00061     return _delayExpression;
00062 }
00063
00064 void Delay::setDelayTimeUnit(Util::TimeUnit _delayTimeUnit) {
00065     this->_delayTimeUnit = _delayTimeUnit;
00066 }
00067
00068 Util::TimeUnit Delay::delayTimeUnit() const {
00069     return _delayTimeUnit;
00070 }
00071
00072 void Delay::_execute(Entity* entity) {
00073     double waitTime = _parentModel->parseExpression(_delayExpression) *
00074         Util::TimeUnitConvert(_delayTimeUnit, _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00075     if (_reportStatistics) {
00076         _cstatWaitTime->getStatistics()->getCollector()->addValue(waitTime);
00077         if (entity->getEntityType() > isReportStatistics())
00078             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityType() +
00079             ".WaitTime")->getStatistics()->getCollector()->addValue(waitTime);
00080         entity->setAttributeValue("Entity.TotalWaitTime",
00081             entity->getAttributeValue("Entity.TotalWaitTime") + waitTime);
00082         double delayEndTime = _parentModel->getSimulation()->getSimulatedTime() + waitTime;
00083         Event* newEvent = new Event(delayEndTime, entity,
00084             this->getNextComponents() -> getFrontConnection());
00085         _parentModel->getFutureEvents() -> insert(newEvent);
00086         _parentModel->getTracer()->trace("End of delay of entity " +
00087             std::to_string(entity->entityNumber()) + " scheduled to time " + std::to_string(delayEndTime));
00088     }
00089
00090     bool Delay::_loadInstance(std::map<std::string, std::string>* fields) {
00091         bool res = ModelComponent::_loadInstance(fields);
00092         if (res) {
00093             this->_delayExpression = (*fields->find("delayExpression")).second;
00094             this->_delayTimeUnit = static_cast<Util::TimeUnit>(std::stoi(loadField(fields,
00095                 "delayExpressionTimeUnit", std::to_string(static_cast<int> (Util::TimeUnit::second)))));
00096         }
00097     }
00098
00099     std::map<std::string, std::string>* Delay::_saveInstance() {
00100         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00101         //Util::TypeOf<Delay>());
00102         fields->emplace("delayExpression", "\"" + this->_delayExpression + "\"");
00103         if (_delayTimeUnit != Util::TimeUnit::second) fields->emplace("delayExpressionTimeUnit",
00104             std::to_string(static_cast<int> (this->_delayTimeUnit)));
00105         return fields;
00106     }
00107
00108     bool Delay::_check(std::string* errorMessage) {
00109         //include attributes needed
00110         ElementManager* elements = _parentModel->getElements();
00111         std::vector<std::string> neededNames = {"Entity.TotalWaitTime"};
00112         std::string neededName;
00113         for (unsigned int i = 0; i < neededNames.size(); i++) {
00114             neededName = neededNames[i];
00115             if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00116                 Attribute* attr1 = new Attribute(_parentModel, neededName);
00117                 elements->insert(attr1);
00118             }
00119         }
00120         return _parentModel->checkExpression(_delayExpression, "Delay expression", errorMessage);
00121     }
00122
00123     void Delay::_createInternalElements() {
00124         if (_reportStatistics && _cstatWaitTime == nullptr) {
00125             _cstatWaitTime = new StatisticsCollector(_parentModel, _name + "." + "WaitTime", this);
00126             _childrenElements->insert({"WaitTime", _cstatWaitTime});
00127             // include StatisticsCollector needed in EntityType
00128             ElementManager* elements = _parentModel->getElements();
00129             std::list<ModelElement*>* entypes =
00130                 elements->getElementTypeList(Util::TypeOf<EntityType>())->list();
00131             for (std::list<ModelElement*>::iterator it = entypes->begin(); it != entypes->end(); it++) {
00132                 EntityType* entype = static_cast<EntityType*> ((*it));
00133             }
00134         }
00135     }

```

```
00129         if ((*it)->isReportStatistics())
00130             enttype->addGetStatisticsCollector(enttype->getName() + ".WaitTime"); // force create
this CStat before simulation starts
00131     }
00132 } else {
00133     _removeChildrenElements();
00134     // \todo remove StatisticsCollector needed in EntityType
00135 }
00136 }
00137
00138 PluginInformation* Delay::GetPluginInformation() {
00139     PluginInformation* info = new PluginInformation(Util::TypeOf<Delay>(), &Delay::LoadInstance);
00140     return info;
00141 }
```

9.67 Delay.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Plugin.h"
Include dependency graph for Delay.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Delay

9.68 Delay.h

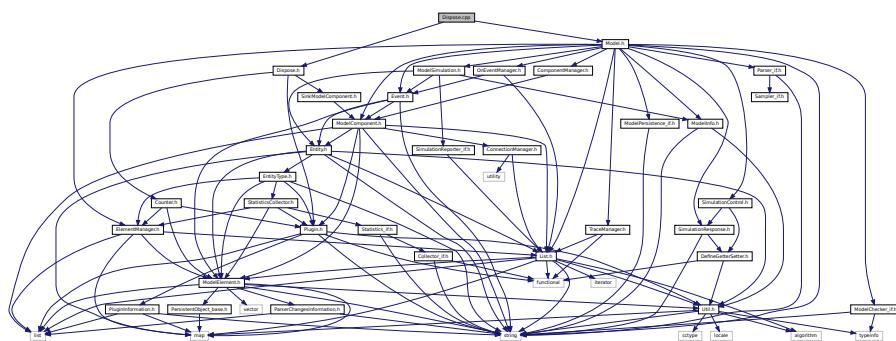
```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:     Delay.h
00009 * Author:   rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:49
00012 */
00013
00014 #ifndef DELAY_H
```

```
00015 #define DELAY_H
00016
00017 #include <string>
00018 #include "ModelComponent.h"
00019 #include "Plugin.h"
00020
00041 class Delay : public ModelComponent {
00042 public:
00043     Delay(Model* model, std::string name = "");
00044     virtual ~Delay() = default;
00045 public:
00046     void setDelayExpression(std::string _delayExpression);
00047     std::string delayExpression() const;
00048     void setDelay(double delay);
00049     double delay() const;
00050     void setDelayTimeUnit(Util::TimeUnit _delayTimeUnit);
00051     Util::TimeUnit delayTimeUnit() const;
00052 public:
00053     virtual std::string show();
00054 public:
00055     static PluginInformation* GetPluginInformation();
00056     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00057 protected:
00058     virtual void _execute(Entity* entity);
00059     virtual void _initBetweenReplications();
00060     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00061     virtual std::map<std::string, std::string>* _saveInstance();
00062     virtual bool _check(std::string* errorMessage);
00063     virtual void _createInternalElements();
00064 private:
00065     std::string _delayExpression = "1.0";
00066     Util::TimeUnit _delayTimeUnit = Util::TimeUnit::second;
00067 private: // inner children elements
00068     StatisticsCollector* _cstatWaitTime = nullptr;
00069 };
00070
00071 #endif /* DELAY_H */
```

9.69 Dispose.cpp File Reference

```
#include "Dispose.h"  
#include "Model.h"
```

Include dependency graph for Dispose.cpp:



9.70 Dispose.cpp

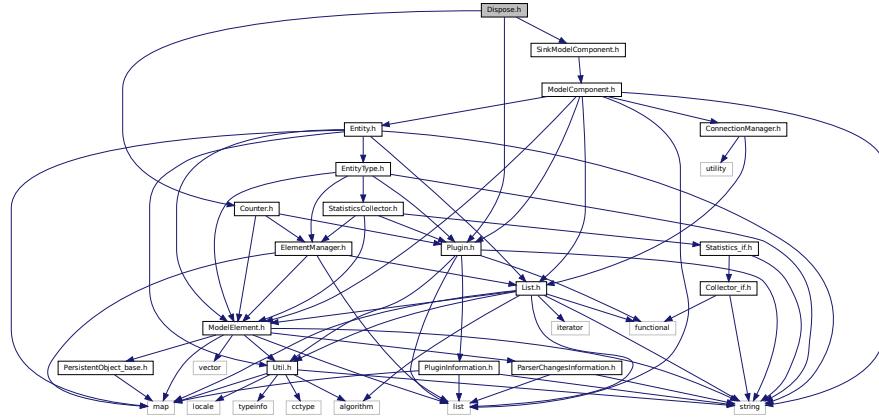
```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:    Dispose.cpp
00009 * Author:  rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 20:13
```

```
00012  */
00013
00014 #include "Dispose.h"
00015 #include "Model.h"
00016
00017 Dispose::Dispose(Model* model, std::string name) : SinkModelComponent(model, Util::TypeOf<Dispose>(),
00018     name) {
00019     // _numberOut = new Counter(_parentModel, _name + "." + "Count_number_out", this);
00020     _connections->setMinOutputConnections(0);
00021     _connections->setMaxOutputConnections(0);
00022 }
00023 std::string Dispose::show() {
00024     return SinkModelComponent::show();
00025 }
00026
00027 void Dispose::_execute(Entity* entity) {
00028     if (_reportStatistics) {
00029         _numberOut->incCountValue();
00030         if (entity->getEntityType()->isReportStatistics()) {
00031             double timeInSystem = _parentModel->getSimulation()->getSimulatedTime() -
00032                 entity->getAttributeValue("Entity.ArrivalTime");
00033             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityType() + "." +
00034                 "TotalTimeInSystem")->getStatistics()->getCollector()->addValue(timeInSystem);
00035         }
00036     }
00037     _parentModel->removeEntity(entity, _reportStatistics);
00038 }
00039
00040 bool Dispose::_loadInstance(std::map<std::string, std::string>* fields) {
00041     return ModelComponent::_loadInstance(fields);
00042 }
00043
00044 void Dispose::_initBetweenReplications() {
00045     SinkModelComponent::_initBetweenReplications();
00046 }
00047
00048 std::map<std::string, std::string>* Dispose::_saveInstance() {
00049     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00050     return fields;
00051 }
00052
00053 bool Dispose::_check(std::string* errorMessage) {
00054     /*
00055     return true;
00056 */
00057 void Dispose::_createInternalElements() {
00058     if (_reportStatistics && _numberOut == nullptr) {
00059         // creates the counter (and then the CStats)
00060         _numberOut = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00061         _childrenElements->insert({"CountNumberIn", _numberOut});
00062         // include StatisticsCollector needed for each EntityType
00063         std::list<ModelElement*>* enttypes =
00064             _parentModel->getElements()->getElementList(Util::TypeOf<EntityType>())->list();
00065         for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00066             if ((*it)->isReportStatistics())
00067                 static_cast<EntityType*>((*it))->addGetStatisticsCollector((*it)->getName() + "." +
00068                     "TotalTimeInSystem"); // force create this CStat before model checking
00069         }
00070     } else if (! _reportStatistics && _numberOut != nullptr) {
00071         // _numberOut->~Counter();
00072         //_numberOut = nullptr;
00073         // \todo: delete the CSTATS?
00074         _removeChildrenElements();
00075     }
00076 }
00077 PluginInformation* Dispose::GetPluginInformation() {
00078     PluginInformation* info = new PluginInformation(Util::TypeOf<Dispose>(), &Dispose::LoadInstance);
00079     info->setSink(true);
00080     return info;
00081 }
00082 ModelComponent* Dispose::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00083     Dispose* newComponent = new Dispose(model);
00084     try {
00085         newComponent->_loadInstance(fields);
00086     } catch (const std::exception& e) {
00087     }
00088     return newComponent;
00089 }
00090 }
```

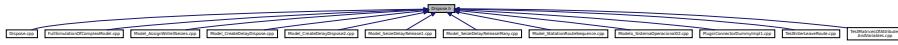
9.71 Dispose.h File Reference

```
#include "SinkModelComponent.h"
#include "Counter.h"
#include "Plugin.h"
```

Include dependency graph for Dispose.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Dispose](#)

9.72 Dispose.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Dispose.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 20:13
00012 */
00013
00014 #ifndef DISPOSE_H
00015 #define DISPOSE_H
00016
00017 #include "SinkModelComponent.h"
00018 #include "Counter.h"
00019 #include "Plugin.h"
00020
00038 class Dispose : public SinkModelComponent {
00039 public:
00040     Dispose(Model* model, std::string name = "");
00041     virtual ~Dispose() = default;
00042 public:
00043     virtual std::string show();
00044 public:
00045     static PluginInformation* GetPluginInformation();
```

```

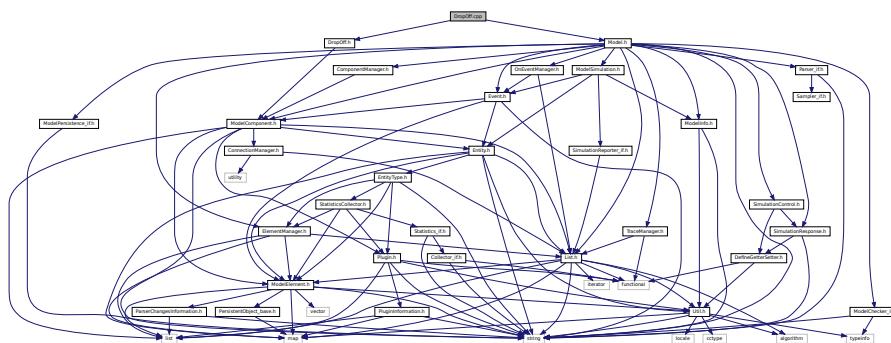
00046     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00047 protected:
00048     virtual void _execute(Entity* entity);
00049     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050     virtual void _initBetweenReplications();
00051     virtual std::map<std::string, std::string>* _saveInstance();
00052     virtual bool _check(std::string* errorMessage);
00053     virtual void _createInternalElements();
00054 private: // children elements
00055     Counter* _numberOut = nullptr;
00056 };
00057
00058 #endif /* DISPOSE_H */
00059

```

9.73 DropOff.cpp File Reference

```
#include "DropOff.h"
#include "Model.h"
```

Include dependency graph for DropOff.cpp:



9.74 DropOff.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: DropOff.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #include "DropOff.h"
00015 #include "Model.h"
00016
00017 DropOff::DropOff(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<DropOff>(),
00018   name) {
00019
00020   std::string DropOff::show() {
00021     return ModelComponent::show() + "";
00022   }
00023
00024 ModelComponent* DropOff::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025   DropOff* newComponent = new DropOff(model);
00026   try {
00027     newComponent->_loadInstance(fields);
00028   } catch (const std::exception& e) {
00029   }
00030   return newComponent;
00031 }
00032
00033
00034 void DropOff::_execute(Entity* entity) {

```

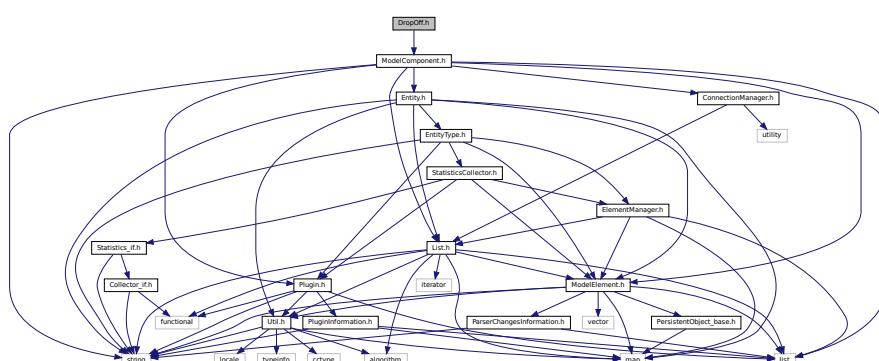
```

00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
00038
00039 bool DropOff::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
00046
00047 void DropOff::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* DropOff::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool DropOff::_check(std::string* errorMessage) {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
00061
00062 PluginInformation* DropOff::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<DropOff>(), &DropOff::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067

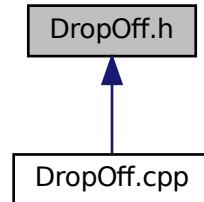
```

9.75 DropOff.h File Reference

#include "ModelComponent.h"
Include dependency graph for DropOff.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [DropOff](#)

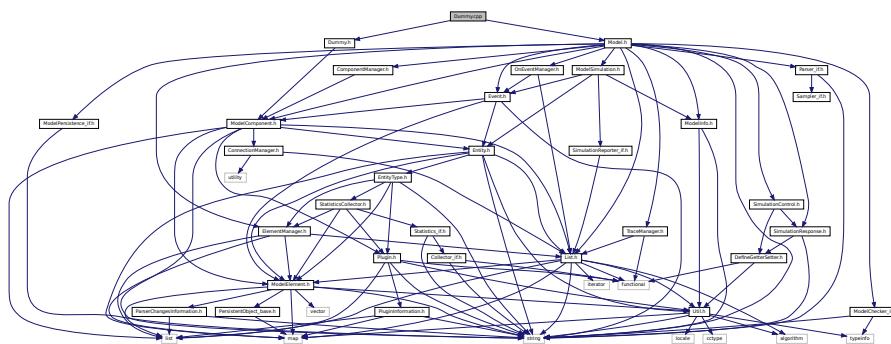
9.76 DropOff.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: DropOff.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #ifndef DROPOFF_H
00015 #define DROPOFF_H
00016
00017 #include "ModelComponent.h"
00018
00041 class DropOff : public ModelComponent {
00042 public: // constructors
00043     DropOff(Model* model, std::string name = "");
00044     virtual ~DropOff() = default;
00045 public: // virtual
00046     virtual std::string show();
00047 public: // static
00048     static PluginInformation* GetPluginInformation();
00049     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00050 protected: // virtual
00051     virtual void _execute(Entity* entity);
00052     virtual void _initBetweenReplications();
00053     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00054     virtual std::map<std::string, std::string>* _saveInstance();
00055     virtual bool _check(std::string* errorMessage);
00056 private: // methods
00057 private: // attributes 1:1
00058 private: // attributes 1:n
00059 };
00060
00061
00062 #endif /* DROPOFF_H */
00063
```

9.77 Dummy.cpp File Reference

```
#include "Dummy.h"
```

```
#include "Model.h"
Include dependency graph for Dummy.cpp:
```



9.78 Dummy.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Dummy.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Maio de 2019, 18:41
00012 */
00013
00014 #include "Dummy.h"
00015 #include "Model.h"
00016
00017 Dummy::Dummy(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Dummy>(), name) {
00018 }
00019
00020 std::string Dummy::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* Dummy::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     Dummy* newComponent = new Dummy(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030     return newComponent;
00031 }
00032
00033
00034 void Dummy::_execute(Entity* entity) {
00035     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00037         0.0);
00038 }
00039 bool Dummy::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
00046
00047 void Dummy::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Dummy::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool Dummy::_check(std::string* errorMessage) {
00057     bool resultAll = true;
```

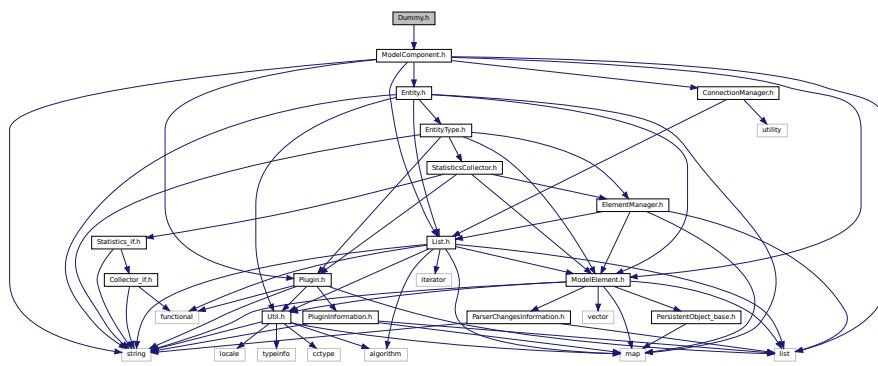
```

00058     //...
00059     return resultAll;
00060 }
00061
00062 PluginInformation* Dummy::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Dummy>(), &Dummy::LoadInstance);
00064     // ...
00065     return info;
00066 }

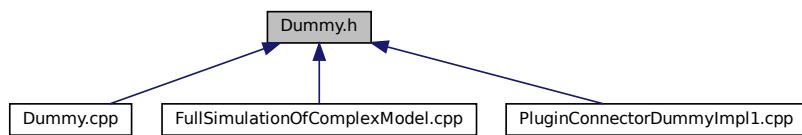
```

9.79 Dummy.h File Reference

#include "ModelComponent.h"
Include dependency graph for Dummy.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Dummy](#)

9.80 Dummy.h

```

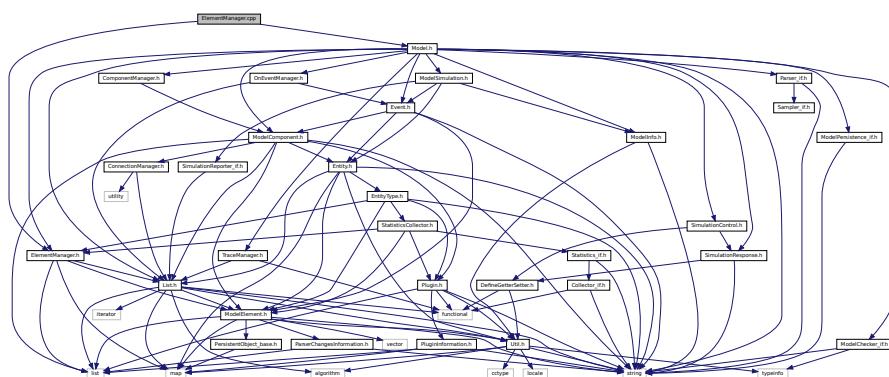
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Dummy.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 22 de Maio de 2019, 18:41
00012 */

```

```
00013
00014 #ifndef DUMMY_H
00015 #define DUMMY_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Dummy : public ModelComponent {
00020 public: // constructors
00021     Dummy(Model* model, std::string name = "");
00022     virtual ~Dummy() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00028 protected: // virtual
00029     virtual void _execute(Entity* entity);
00030     virtual void _initBetweenReplications();
00031     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032     virtual std::map<std::string, std::string>* _saveInstance();
00033     virtual bool _check(std::string* errorMessage);
00034 private: // methods
00035 private: // attributes 1:1
00036 private: // attributes 1:n
00037 };
00038
00039 #endif /* DUMMY_H */
```

9.81 ElementManager.cpp File Reference

```
#include "ElementManager.h"  
#include "Model.h"  
Include dependency graph for ElementManager.cpp:
```



9.82 ElementManager.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: ElementManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 12:48
00012 */
00013
00014 #include "ElementManager.h"
00015 #include "Model.h"
00016
00017 //using namespace GenesysKernel;
00018
00019 ElementManager::ElementManager(Model* model) {
```

```

00020     _parentModel = model;
00021     /* \todo: -- Sort methods for elements should be a decorator */
00022     _elements = new std::map<std::string, List<ModelElement*>>();
00023     //_elements->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00024     //    return a->getId() < b->getId();
00025     //});
00026
00027 }
00028
00029 bool ElementManager::insert(ModelElement* anElement) {
00030     std::string elementTypename = anElement->getclassname();
00031     bool res = insert(elementTypename, anElement);
00032     /*
00033     if (res)
00034         _parentModel->tracer()->trace(Util::TraceLevel::elementResult, "Element " + anElement->name()
00035         + " successfully inserted");
00036     else
00037         _parentModel->tracer()->trace(Util::TraceLevel::elementResult, "Element " + anElement->name()
00038         + " could not be inserted");
00039     */
00040     return res;
00041 }
00042
00043 bool ElementManager::insert(std::string elementTypename, ModelElement* anElement) {
00044     List<ModelElement*>* listElements = getElementList(elementTypename);
00045     if (listElements->find(anElement) == listElements->list()->end()) { //not found
00046         listElements->insert(anElement);
00047         this->_parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Element " +
00048             anElement->getclassname() + " \" " + anElement->getName() + " \" successfully inserted.");
00049         return true;
00050     }
00051     this->_parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Element \" " +
00052         anElement->getName() + " \" could not be inserted.");
00053     return false;
00054 }
00055
00056 void ElementManager::remove(ModelElement* anElement) {
00057     std::string elementTypename = anElement->getclassname();
00058     List<ModelElement*>* listElements = getElementList(elementTypename);
00059     listElements->remove(anElement);
00060     _parentModel->getTracer()->trace(Util::TraceLevel::elementResult, "Element successfully removed");
00061 }
00062
00063 void ElementManager::remove(std::string elementTypename, ModelElement* anElement) {
00064     List<ModelElement*>* listElements = getElementList(elementTypename);
00065     listElements->remove(anElement);
00066 }
00067
00068 bool ElementManager::check(std::string elementTypename, std::string elementName, std::string
expressionName, bool mandatory, std::string* errorMessage) {
00069     if (elementName == "" && !mandatory) {
00070         return true;
00071     }
00072     bool result = getElement(elementTypename, elementName) != nullptr;
00073     if (!result) {
00074         std::string msg = elementTypename + " \" " + elementName + " \" for '" + expressionName + "' is
not in the model.";
00075         errorMessage->append(msg);
00076     }
00077     return result;
00078 }
00079
00080 bool ElementManager::check(std::string elementTypename, ModelElement* anElement, std::string
expressionName, std::string* errorMessage) {
00081     bool result = anElement != nullptr;
00082     if (!result) {
00083         std::string msg = elementTypename + " for '" + expressionName + "' is null.";
00084         errorMessage->append(msg);
00085     } else {
00086         result = check(elementTypename, anElement->getName(), expressionName, true, errorMessage);
00087     }
00088     return result;
00089 }
00090
00091 unsigned int ElementManager::getNumberOfElements(std::string elementTypename) {
00092     List<ModelElement*>* listElements = getElementList(elementTypename);
00093     return listElements->size();
00094 }
00095
00096 unsigned int ElementManager::getNumberOfElements() {
00097     unsigned int total = 0;
00098     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin(); it !=
```

```

    _elements->end(); it++) {
00100         total += (*it).second->size();
00101     }
00102     return total;
00103 }
00104
00105 void ElementManager::show() {
00106     _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, "Model Elements:");
00107     //std::map<std::string, List<ModelElement*>>* _elements;
00108     std::string key;
00109     List<ModelElement*>* list;
00110     Util::IncIndent();
00111     {
00112         for (std::map<std::string, List<ModelElement*>>::iterator infraIt = _elements->begin();
00113             infraIt != _elements->end(); infraIt++) {
00114             key = (*infraIt).first;
00115             list = (*infraIt).second;
00116             _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, key + ": (" +
00117                 std::to_string(list->size()) + ")");
00118             Util::IncIndent();
00119             {
00120                 for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end(); it++) {
00121                     _parentModel->getTracer()->trace(Util::TraceLevel::toolDetailed, (*it)->show());
00122                 }
00123             }
00124         }
00125         Util::DecIndent();
00126     }
00127
00128 Model* ElementManager::getParentModel() const {
00129     return _parentModel;
00130 }
00131
00132 bool ElementManager::hasChanged() const {
00133     return _hasChanged;
00134 }
00135
00136 void ElementManager::setHasChanged(bool _hasChanged) {
00137     this->_hasChanged = _hasChanged;
00138 }
00139
00140 List<ModelElement*>* ElementManager::getElementList(std::string elementTypename) const {
00141     std::map<std::string, List<ModelElement*>>::iterator it = this->_elements->find(elementTypename);
00142     if (it == this->_elements->end()) {
00143         // list does not exists yet. Create it and set a valid iterator
00144         List<ModelElement*>* newList = new List<ModelElement*>();
00145         newList->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00146             return a->getId() < b->getId();
00147         });
00148         _elements->insert(std::pair<std::string, List<ModelElement*>>(elementTypename, newList));
00149         it = this->_elements->find(elementTypename);
00150     }
00151     List<ModelElement*>* infras = it->second;
00152     return infras;
00153 }
00154
00155 ModelElement* ElementManager::getElement(std::string elementTypename, Util::identification id) {
00156     List<ModelElement*>* list = getElementList(elementTypename);
00157     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00158         it++) {
00159         if ((*it)->getId() == id) { // found
00160             return (*it);
00161         }
00162     }
00163     return nullptr;
00164 }
00165 int ElementManager::getRankOf(std::string elementTypename, std::string name) {
00166     int rank = 0;
00167     List<ModelElement*>* list = getElementList(elementTypename);
00168     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00169         it++) {
00170         if ((*it)->getName() == name) { // found
00171             return rank;
00172         } else {
00173             rank++;
00174         }
00175     }
00176     return -1;
00177 }
00178 std::list<std::string>* ElementManager::getElementClassnames() const {
00179     std::list<std::string>* keys = new std::list<std::string>();
00180     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin(); it !=

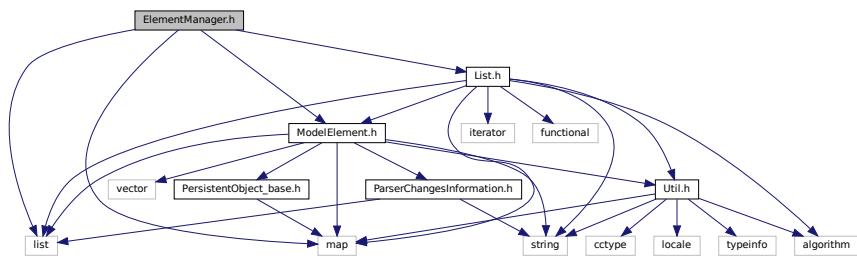
```

```

00181     _elements->end(); it++) {
00182         keys->insert(keys->end(), (*it).first);
00183     }
00184 }
00185
00186 ModelElement* ElementManager::getElement(std::string elementTypename, std::string name) {
00187     List<ModelElement*>* list = getElementList(elementTypename);
00188     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end();
00189     it++) {
00190         if ((*it)->getName() == name) { // found
00191             return (*it);
00192         }
00193     }
00194     return nullptr;
00195 }
```

9.83 ElementManager.h File Reference

```
#include <list>
#include <map>
#include "List.h"
#include "ModelElement.h"
Include dependency graph for ElementManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ElementManager](#)

9.84 ElementManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   ElementManager.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 7 de Novembro de 2018, 12:48
00012 */
00013
00014 #ifndef ELEMENTMANAGER_H
```

```

00015 #define ELEMENTMANAGER_H
00016
00017 #include <list>
00018 #include <map>
00019 #include "List.h"
00020 #include "ModelElement.h"
00021
00022 //namespace GenesysKernel {
00023     class Model;
00024
00025     class ElementManager {
00026     public:
00027         ElementManager(Model* model);
00028         virtual ~ElementManager() = default;
00029     public:
00030         bool insert(ModelElement* anElement);
00031         void remove(ModelElement* anElement);
00032         bool insert(std::string elementTypename, ModelElement* anElement);
00033         void remove(std::string elementTypename, ModelElement* anElement);
00034         bool check(std::string elementTypename, ModelElement* anElement, std::string expressionName,
00035                     std::string* errorMessage);
00036         bool check(std::string elementTypename, std::string elementName, std::string expressionName,
00037                     bool mandatory, std::string* errorMessage);
00038         void clear();
00039     public:
00040         ModelElement* getElement(std::string elementTypename, Util::identification id);
00041         ModelElement* getElement(std::string elementTypename, std::string name);
00042         unsigned int getNumberOfElements(std::string elementTypename);
00043         unsigned int getNumberOfElements();
00044         int getRankOf(std::string elementTypename, std::string name);
00045         std::list<std::string>* getElementClassnames() const;
00046
00047     //private:
00048     public:
00049         // \todo: MUST BE PRIVATE
00050         List<ModelElement*>* getElementList(std::string elementTypename) const;
00051     public:
00052         void show();
00053         Model* getParentModel() const;
00054         bool hasChanged() const;
00055         void setHasChanged(bool _hasChanged);
00056     private:
00057         std::map<std::string, List<ModelElement*>*>* _elements;
00058         Model* _parentModel;
00059         bool _hasChanged = false;
00060     };
00061 //namespace\\}
00062 #endif /* ELEMENTMANAGER_H */
00063

```

9.85 ElementManager_if.h File Reference

Classes

- class [ElementManager_if](#)

9.86 ElementManager_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   ElementManager_if.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 22 de Novembro de 2018, 01:06
00012 */
00013
00014 #ifndef ELEMENTMANAGER_IF_H
00015 #define ELEMENTMANAGER_IF_H
00016
00017 class ElementManager_if {
00018 public:
00019     ElementManager_if();
00020     ElementManager_if(const ElementManager_if& orig);

```

```

00021     virtual ~ElementManager_if();
00022 private:
00023 };
00024 };
00025
00026 #endif /* ELEMENTMANAGER_IF_H */
00027

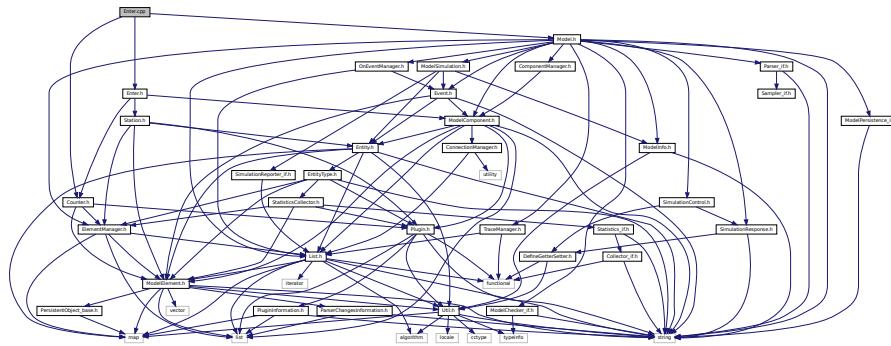
```

9.87 Enter.cpp File Reference

```

#include "Enter.h"
#include "Model.h"
#include "Counter.h"
Include dependency graph for Enter.cpp:

```



9.88 Enter.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Enter.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Enter.h"
00015 #include "Model.h"
00016 #include "Counter.h"
00017
00018 Enter::Enter(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Enter>(), name) {
00019 }
00020
00021 std::string Enter::show() {
00022     return ModelComponent::show() + ",station=" + this->_station->getName();
00023 }
00024
00025 ModelComponent* Enter::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Enter* newComponent = new Enter(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
00034
00035 void Enter::setStation(Station* _station) {
00036     this->_station = _station;
00037     _station->setEnterIntoStationComponent(this);
00038 }
00039
00040 Station* Enter::getStation() const {

```

```

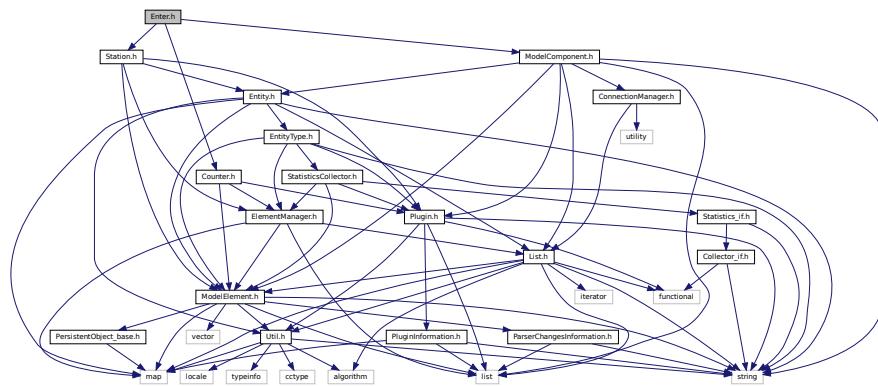
00041     return _station;
00042 }
00043
00044 void Enter::_execute(Entity* entity) {
00045     if (_reportStatistics)
00046         _numberIn->incCountValue();
00047     _station->enter(entity);
00048     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00049 }
00050
00051 bool Enter::_loadInstance(std::map<std::string, std::string>* fields) {
00052     bool res = ModelComponent::_loadInstance(fields);
00053     if (res) {
00054         std::string stationName = ((*fields->find("stationName"))).second;
00055         Station* station = dynamic_cast<Station*>
00056             (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00057         this->_station = station;
00058     }
00059     return res;
00060 }
00061 void Enter::_initBetweenReplications() {
00062     _numberIn->clear();
00063 }
00064
00065 std::map<std::string, std::string>* Enter::_saveInstance() {
00066     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00067     fields->emplace("stationName", "\"" + (this->_station->getName()) + "\"");
00068     return fields;
00069 }
00070
00071 bool Enter::_check(std::string* errorMessage) {
00072     bool resultAll = true;
00073     resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",
00074     errorMessage);
00075     return resultAll;
00076 }
00077 PluginInformation* Enter::GetPluginInformation() {
00078     PluginInformation* info = new PluginInformation(Util::TypeOf<Enter>(), &Enter::LoadInstance);
00079     info->setReceiveTransfer(true);
00080     info->insertDynamicLibFileDependence("station.so");
00081     return info;
00082 }
00083
00084 void Enter::_createInternalElements() {
00085     if (_reportStatistics) {
00086         if (_numberIn == nullptr) {
00087             _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00088             _childrenElements->insert({"CountNumberIn", _numberIn});
00089         }
00090     } else
00091         if (_numberIn != nullptr) {
00092             _removeChildrenElements();
00093         }
00094 }

```

9.89 Enter.h File Reference

```
#include "ModelComponent.h"
#include "Station.h"
#include "Counter.h"
```

Include dependency graph for Enter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Enter](#)

9.90 Enter.h

```

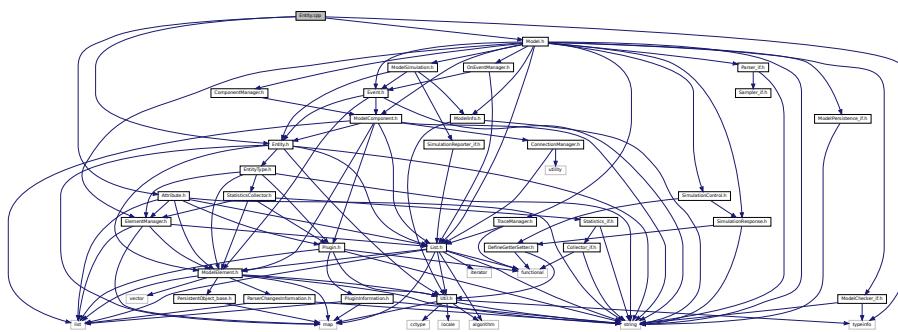
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Enter.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef ENTER_H
00015 #define ENTER_H
00016
00017 #include "ModelComponent.h"
00018 #include "Station.h"
00019 #include "Counter.h"
00098 class Enter : public ModelComponent {
00099 public:
00100     Enter(Model* model, std::string name = "");
00101     virtual ~Enter() = default;
00102 public:
00103     virtual std::string show();
00104 public:
00105     static PluginInformation* GetPluginInformation();
00106     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00107 protected:
00108     void setStation(Station* _station);
00109     Station* getStation() const;
00110 protected:
00111     virtual void _execute(Entity* entity);
00112     virtual void _initBetweenReplications();

```

```
00113     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00114     virtual std::map<std::string, std::string>* _saveInstance();
00115     virtual bool _check(std::string* errorMessage);
00116     virtual void _createInternalElements();
00117 private: // association
00118     Station* _station;
00119 private: // children elements
00120     Counter* _numberIn = nullptr;
00121 };
00122
00123 #endif /* ENTER_H */
00124
```

9.91 Entity.cpp File Reference

```
#include <typeinfo>
#include "Entity.h"
#include "Attribute.h"
#include "Model.h"
Include dependency graph for Entity.cpp:
```



9.92 Entity.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Entity.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 16:30
00012 */
00013
00014 #include <typeinfo>
00015 #include "Entity.h"
00016 #include "Attribute.h"
00017 #include "Model.h"
00018
00019 //using namespace GenesysKernel;
00020
00021 Entity::Entity(Model* model) : ModelElement(model, Util::TypeOf<Entity>()) {
00022     //_elements = elements;
00023     _entityNumber = Util::GetLastIdOfType(Util::TypeOf<Entity>());
00024     unsigned int numAttributes =
00025         _parentModel->getElements()->getNumberOfElements(Util::TypeOf<Attribute>());
00026     for (unsigned i = 0; i < numAttributes; i++) {
00027         std::map<std::string, double>* map = new std::map<std::string, double>();
00028         _attributeValues->insert(map);
00029     }
00030 }
00031 void Entity::setEntityTypeName(std::string entityTypeName) throw () {
00032     EntityType* entitytype = dynamic_cast<EntityType*>
00033     (_parentModel->getElements()->getElement(Util::TypeOf<EntityType>(), entityTypeName));
```

```

00033     if (entitytype != nullptr) {
00034         this->_entityType = entitytype;
00035     } else {
00036         throw std::invalid_argument("EntityType does not exist");
00037     }
00038 }
00039
00040 std::string Entity::getEntityType() const {
00041     return this->_entityType->getName();
00042 }
00043
00044 void Entity::setEntityType(EntityType* entityType) {
00045     _entityType = entityType;
00046 }
00047
00048 EntityType* Entity::getEntityType() const {
00049     return _entityType;
00050 }
00051
00052 std::string Entity::show() {
00053     std::string message = ModelElement::show();
00054     if (this->_entityType != nullptr) {
00055         message += ",entityType=\"" + this->_entityType->getName() + "\"";
00056     }
00057     message += ",attributes=\"";
00058     _attributeValues->front();
00059     for (unsigned int i = 0; i < _attributeValues->size(); i++) {
00060         std::map<std::string, double>* map = _attributeValues->current();
00061         std::string attributeName =
00062             _parentModel->getElements()->getElementList(Util::TypeOf<Attribute>())->getAtRank(i)->getName();
00063         message += attributeName + "=";
00064         if (map->size() == 0) { // scalar
00065             message += "NaN"; // std::to_string(map->begin()->second) + ",";
00066         } else if (map->size() == 1) { // scalar
00067             message += std::to_string(map->begin()->second) + ",";
00068         } else {
00069             // array or matrix
00070             message += "[";
00071             for (std::map<std::string, double>::iterator valIt = map->begin(); valIt != map->end();
00072                 valIt++) {
00073                 message += (*valIt).first + "=>" + std::to_string((*valIt).second) + ",";
00074             }
00075             message = message.substr(0, message.length() - 1);
00076             message += "]";
00077         }
00078         _attributeValues->next();
00079     }
00080     message = message.substr(0, message.length() - 1);
00081     message += "]";
00082     return message;
00083 }
00084
00085 double Entity::getAttributeValue(std::string attributeName) {
00086     return getAttributeValue("", attributeName);
00087 }
00088
00089 double Entity::getAttributeValue(std::string index, std::string attributeName) {
00090     int rank = _parentModel->getElements()->getRankOf(Util::TypeOf<Attribute>(), attributeName);
00091     if (rank >= 0) {
00092         std::map<std::string, double>* map = this->_attributeValues->getAtRank(rank);
00093         std::map<std::string, double>::iterator mapIt = map->find(index);
00094         if (mapIt != map->end()) { // found
00095             return (*mapIt).second;
00096         } else { // not found
00097             return 0.0;
00098         }
00099     }
00100     _parentModel->getTracer()->trace(Util::TraceLevel::errorRecover, "Attribute \""
00101                                         + attributeName +
00102                                         "\" not found");
00103     return 0.0; /* todo: !! Never should happen. check how to report */
00104 }
00105
00106 double Entity::getAttributeValue(Util::identification attributeID) {
00107     ModelElement* element = _parentModel->getElements()->getElement(Util::TypeOf<Attribute>(),
00108     attributeID);
00109     if (element != nullptr) {
00110         return getAttributeValue(index, element->getName());
00111     }
00112     return 0.0; // attribute not found
00113 }
00114 void Entity::setAttributeValue(std::string attributeName, double value) {
00115     setAttributeValue("", attributeName, value);

```

```

00116 }
00117
00118 void Entity::setAttributeValue(std::string index, std::string attributeName, double value) {
00119     int rank = _parentModel->getElements()->getRankOf(Util::TypeOf<Attribute>(), attributeName);
00120     if (rank >= 0) {
00121         std::map<std::string, double>* map = _attributeValues->getAtRank(rank);
00122         std::map<std::string, double>::iterator mapIt = map->find(index);
00123         if (mapIt != map->end()) { // found
00124             (*mapIt).second = value;
00125         } else { // not found
00126             map->insert({index, value}); // (map->end(), std::pair<std::string, double>(index,
00127             value));
00128         }
00129     } else
00129     _parentModel->getTracer()->trace(Util::TraceLevel::errorRecover, "Attribute \""
00130     attributeName + "\" not found");
00131 }
00132
00133 void Entity::setAttributeValue(Util::identification attributeID, double value) {
00134     setAttributeValue("", attributeID, value);
00135 }
00136
00137 void Entity::setAttributeValue(std::string index, Util::identification attributeID, double value) {
00138     std::string attrname = _parentModel->getElements()->getElement(Util::TypeOf<Attribute>(),
00139     attributeID)->getName();
00140     setAttributeValue(index, attrname, value);
00141 }
00142 Util::identification Entity::entityNumber() const {
00143     return _entityNumber;
00144 }
00145
00146 bool Entity::_loadInstance(std::map<std::string, std::string>* fields) {
00147     // never loads an entity
00148     return true;
00149 }
00150
00151 std::map<std::string, std::string>* Entity::_saveInstance() {
00152     return new std::map<std::string, std::string>();
00153 }
00154
00155 bool Entity::_check(std::string* errorMessage) {
00156     return true;
00157 }

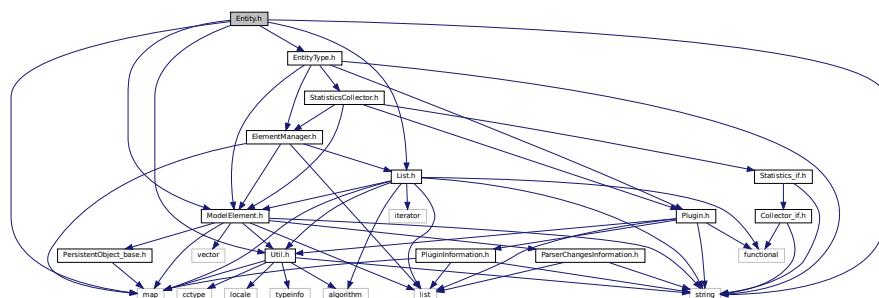
```

9.93 Entity.h File Reference

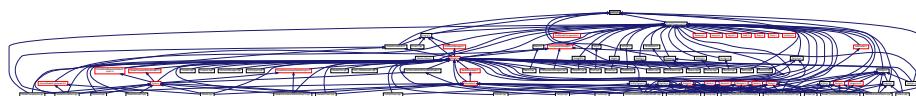
```

#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

9.94 Entity.h

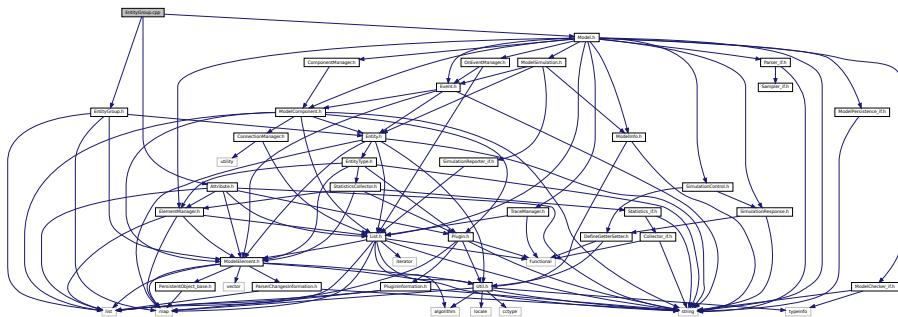
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Entity.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 16:30
00012 */
00013
00014 #ifndef ENTITY_H
00015 #define ENTITY_H
00016
00017 #include <string>
00018 #include <map>
00019
00020 #include "Util.h"
00021 #include "List.h"
00022 #include "ModelElement.h"
00023 #include "EntityType.h"
00024 //namespace GenesysKernel {
00025
00026     class Entity : public ModelElement {
00027     public:
00028         Entity(Model* model);
00029         virtual ~Entity() = default;
00030     public:
00031         virtual std::string show();
00032
00033     public: // g & s
00034         void setEntityType(std::string entityTypeName) throw(); // indirect access to EntityType
00035         std::string getEntityType() const;
00036         void setEntityType(EntityType* entityType); // direct access to EntityType
00037         EntityType* getEntityType() const;
00038     public:
00039         double getAttributeValue(std::string attributeName);
00040         double getAttributeValue(std::string index, std::string attributeName);
00041         double getAttributeValue(Util::identification attributeID);
00042         double getAttributeValue(std::string index, Util::identification attributeID);
00043         void setAttributeValue(std::string attributeName, double value);
00044         void setAttributeValue(std::string index, std::string attributeName, double value);
00045         void setAttributeValue(Util::identification attributeID, double value);
00046         void setAttributeValue(std::string index, Util::identification attributeID, double value);
00047         Util::identification entityNumber() const;
00048     protected:
00049         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050         virtual std::map<std::string, std::string>* _saveInstance();
00051         virtual bool _check(std::string* errorMessage);
00052     private:
00053         Util::identification _entityNumber;
00054         EntityType* _entityType = nullptr;
00055         List<std::map<std::string, double>*>* _attributeValues = new
00056             List<std::map<std::string, double*>*>();
00057     };
00058 //namespace\\}
00059 #endif /* ENTITY_H */
00060

```

9.95 EntityGroup.cpp File Reference

```
#include "EntityGroup.h"
#include "Model.h"
#include "Attribute.h"
Include dependency graph for EntityGroup.cpp:
```



9.96 EntityGroup.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Group.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 12 de Junho de 2019, 19:00
00012 */
00013
00014 #include "EntityGroup.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 EntityGroup::EntityGroup(Model* model, std::string name) : ModelElement(model,
00019     Util::TypeOf<EntityGroup>(), name) {
00020
00021 EntityGroup::~EntityGroup() {
00022     //parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatNumberInGroup);
00023 }
00024
00025 std::string EntityGroup::show() {
00026     return ModelElement::show() +
00027         ",entities=" + this->_list->show();
00028 }
00029
00030 void EntityGroup::insertElement(Entity* element) {
00031     _list->insert(element);
00032     this->_cstatNumberInGroup->getStatistics()->getCollector()->addValue(_list->size());
00033 }
00034
00035 void EntityGroup::removeElement(Entity* element) {
00036     //double tnow = this->_elements->getParentModel()->simulation()->getSimulatedTime();
00037     _list->remove(element);
00038     this->_cstatNumberInGroup->getStatistics()->getCollector()->addValue(_list->size());
00039 }
00040
00041 void EntityGroup::initBetweenReplications() {
00042     this->_list->clear();
00043     this->_cstatNumberInGroup->getStatistics()->getCollector()->clear();
00044 }
00045
00046 unsigned int EntityGroup::size() {
00047     return _list->size();
00048 }
00049
00050 Entity* EntityGroup::first() {
00051     return _list->front();
00052 }
```

```

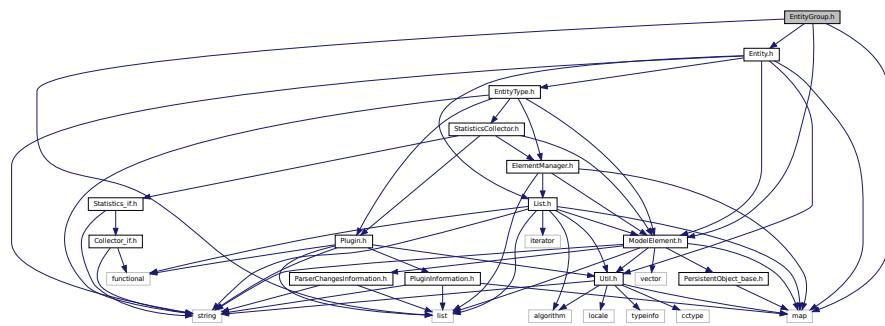
00053
00054 //List<Waiting*>* Group::getList() const {
00055 //    return _list;
00056 //}
00057
00058 PluginInformation* EntityGroup::GetPluginInformation() {
00059     PluginInformation* info = new PluginInformation(Util::TypeOf<EntityGroup>(),
00060     &EntityGroup::LoadInstance);
00061     return info;
00062 }
00063 ModelElement* EntityGroup::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00064     EntityGroup* newElement = new EntityGroup(model);
00065     try {
00066         newElement->_loadInstance(fields);
00067     } catch (const std::exception& e) {
00068     }
00069 }
00070     return newElement;
00071 }
00072
00073 bool EntityGroup::_loadInstance(std::map<std::string, std::string>* fields) {
00074     bool res = ModelElement::_loadInstance(fields);
00075     if (res) {
00076         try {
00077             } catch (...) {
00078         }
00079     }
00080     return res;
00081 }
00082
00083 std::map<std::string, std::string>* EntityGroup::_saveInstance() {
00084     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00085     //Util::TypeOf<Group>());
00086     return fields;
00087 }
00088 bool EntityGroup::_check(std::string* errorMessage) {
00089     std::string newNeededAttributeName = "Entity.Group";
00090     if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), newNeededAttributeName) ==
00091         nullptr) {
00092         new Attribute(_parentModel, newNeededAttributeName);
00093     }
00094     return true;
00095 }
00096 void EntityGroup::_createInternalElements() {
00097     if (_reportStatistics) {
00098         if (_cstatNumberInGroup == nullptr) {
00099             _cstatNumberInGroup = new StatisticsCollector(_parentModel, "NumberInGroup", this);
00100             _childrenElements->insert({"NumberInGroup", _cstatNumberInGroup});
00101         }
00102     } else
00103         if (_cstatNumberInGroup != nullptr) {
00104             _removeChildrenElements();
00105         }
00106 }
00107
00108

```

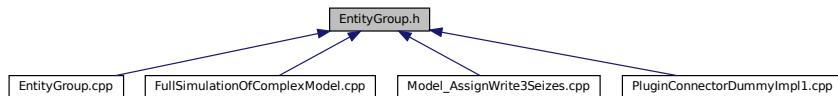
9.97 EntityGroup.h File Reference

```
#include "ModelElement.h"
#include "Entity.h"
#include <map>
#include <list>
```

Include dependency graph for EntityGroup.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EntityGroup

9.98 EntityGroup.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Group.h
00009  * Author: rlcancian
00010 *
00011  * Created on 12 de Junho de 2019, 19:00
00012 */
00013
00014 #ifndef ENTITYGROUP_H
00015 #define ENTITYGROUP_H
00016
00017 #include "ModelElement.h"
00018 #include "Entity.h"
00019 #include <map>
00020 #include <list>
00021
00022 /*
00023  The EntityGroup represents a set of entities grouped together somehow. The entities in this group
00024  are kept into a internal queue. Usually one entity representing all grouped entities is generated by
00025  ModelComponents and the EntityGroup it represented is accessed through an attribute.
00026 */
00027 class EntityGroup : public ModelElement {
00028 public:
00029     EntityGroup(Model* model, std::string name = "");
00030     virtual ~EntityGroup();
00031 public:
00032     static PluginInformation* GetPluginInformation();
00033     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00034 public:

```

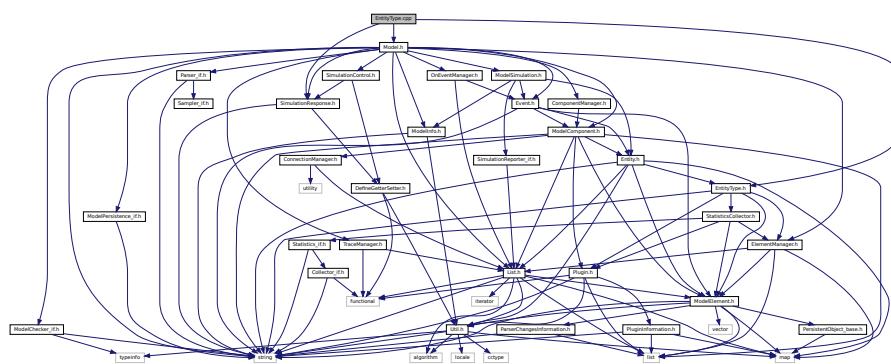
```

00035     void insertElement(Entity* element);
00036     void removeElement(Entity* element);
00037     unsigned int size();
00038     Entity* first();
00039 public:
00040     void initBetweenReplications();
00041 protected:
00042     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00043     virtual std::map<std::string, std::string>* _saveInstance();
00044     virtual bool _check(std::string* errorMessage);
00045     virtual void _createInternalElements();
00046 private:
00047     void _initCStats();
00048 private: //1::n
00049     List<Entity*>* _list = new List<Entity*>();
00050 private: // child inner element
00051     StatisticsCollector* _cstatNumberInGroup = nullptr;
00052 };
00053
00054 #endif /* ENTITYGROUP_H */
00055

```

9.99 EntityType.cpp File Reference

```
#include "EntityType.h"
#include "Model.h"
#include "SimulationResponse.h"
Include dependency graph for EntityType.cpp:
```



9.100 EntityType.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   EntityType.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 14 de Agosto de 2018, 19:24
00012 */
00013
00014 #include "EntityType.h"
00015 #include "Model.h"
00016 #include "SimulationResponse.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 EntityType::EntityType(Model* model, std::string name) : ModelElement(model,
00021     Util::TypeOf<EntityType>(), name) {
00022 }
00023 void EntityType::_initBetweenReplications() {
00024     _initialWaitingCost = 0.0;

```

```

00025     _initialVACost = 0.0;
00026     _initialNVACost = 0.0;
00027     _initialOtherCost = 0.0;
00028     for (std::list<StatisticsCollector*>::iterator it = this->_statisticsCollectors->list()->begin();
00029         it != this->_statisticsCollectors->list()->end(); it++) {
00030         (*it)->getStatistics()->getCollector()->clear();
00031     }
00032
00033 EntityType::~EntityType() {
00034     // remove all CStats
00035     for (std::list<StatisticsCollector*>::iterator it = this->_statisticsCollectors->list()->begin();
00036         it != this->_statisticsCollectors->list()->end(); it++) {
00037         _parentModel->getElements()->remove(Util::TypeOf<StatisticsCollector>(), (*it));
00038     }
00039
00040 std::string EntityType::show() {
00041     return ModelElement::show() +
00042             ",initialPicture=" + this->_initialPicture; // add more...
00043 }
00044
00045 void EntityType::setInitialWaitingCost(double _initialWaitingCost) {
00046     this->_initialWaitingCost = _initialWaitingCost;
00047 }
00048
00049 double EntityType::initialWaitingCost() const {
00050     return _initialWaitingCost;
00051 }
00052
00053 void EntityType::setInitialOtherCost(double _initialOtherCost) {
00054     this->_initialOtherCost = _initialOtherCost;
00055 }
00056
00057 double EntityType::initialOtherCost() const {
00058     return _initialOtherCost;
00059 }
00060
00061 void EntityType::setInitialNVACost(double _initialNVACost) {
00062     this->_initialNVACost = _initialNVACost;
00063 }
00064
00065 double EntityType::initialNVACost() const {
00066     return _initialNVACost;
00067 }
00068
00069 void EntityType::setInitialVACost(double _initialVACost) {
00070     this->_initialVACost = _initialVACost;
00071 }
00072
00073 double EntityType::initialVACost() const {
00074     return _initialVACost;
00075 }
00076
00077 void EntityType::setInitialPicture(std::string _initialPicture) {
00078     this->_initialPicture = _initialPicture;
00079 }
00080
00081 std::string EntityType::initialPicture() const {
00082     return _initialPicture;
00083 }
00084
00085 StatisticsCollector* EntityType::addGetStatisticsCollector(std::string name) {
00086     StatisticsCollector* cstat;
00087     for (std::list<StatisticsCollector*>::iterator it = _statisticsCollectors->list()->begin(); it != _statisticsCollectors->list()->end(); it++) {
00088         cstat = (*it);
00089         if (cstat->getName() == name) {
00090             return cstat;
00091         }
00092     }
00093     // not found. Create it, insert it into the list of cstats, into the model element manager, and
00094     // then return it
00095     cstat = new StatisticsCollector(_parentModel, name, this);
00096     _statisticsCollectors->insert(cstat); // \todo _statisticsCollectors list is probably redundant
00097     _childrenElements->insert({name, cstat});
00098     //_parentModel->insert(cstat); // unnecessary
00099     return cstat;
00100 }
00101 PluginInformation* EntityType::GetPluginInformation() {
00102     PluginInformation* info = new PluginInformation(Util::TypeOf<EntityType>(),
00103         &EntityType::LoadInstance);
00104     return info;
00105 }
```

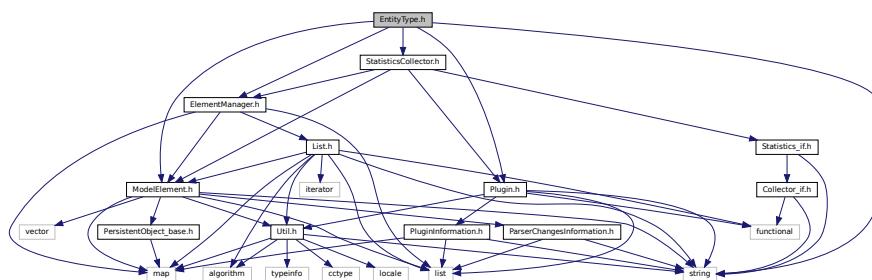
```

00106 ModelElement* EntityType::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00107     EntityType* newElement = new EntityType(model);
00108     try {
00109         newElement->_loadInstance(fields);
00110     } catch (const std::exception& e) {
00111     }
00112     return newElement;
00113 }
00114 }
00115
00116 bool EntityType::_loadInstance(std::map<std::string, std::string>* fields) {
00117     bool res = ModelElement::_loadInstance(fields);
00118     if (res) {
00119         this->_initialNVACost = std::stod(loadField(fields, "initialNVACost", "0.0"));
00120         this->_initialOtherCost = std::stod(loadField(fields, "initialOtherCost", "0.0"));
00121         this->_initialPicture = (loadField(fields, "initialpicture", "0.0"));
00122         this->_initialVACost = std::stod(loadField(fields, "initialVACost", "0.0"));
00123         this->_initialWaitingCost = std::stod(loadField(fields, "initialWaitingCost", "0.0"));
00124     }
00125     return res;
00126 }
00127
00128 std::map<std::string, std::string>* EntityType::_saveInstance() {
00129     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00130     //Util::TypeOf<EntityType>();
00131     if (_initialNVACost != 0.0) fields->emplace("initialNVACost",
00132         std::to_string(this->_initialNVACost));
00133     if (_initialOtherCost != 0.0) fields->emplace("initialOtherCost",
00134         std::to_string(this->_initialOtherCost));
00135     if (_initialPicture != "report") fields->emplace("initialPicture", this->_initialPicture);
00136     if (_initialVACost != 0.0) fields->emplace("initialVACost", std::to_string(this->_initialVACost));
00137     if (_initialWaitingCost != 0.0) fields->emplace("initialWaitingCost",
00138         std::to_string(this->_initialWaitingCost));
00139     return fields;
00140 }
00141
00142 bool EntityType::_check(std::string* errorMessage) {
00143     return true;
00144 }
00145
00146 void EntityType::_createInternalElements() {
00147     if (_reportStatistics) {
00148         if (_statisticsCollectors->size() > 0) {
00149             while (_statisticsCollectors->size() > 0) {
00150                 _parentModel->getElements()->remove(_statisticsCollectors->front());
00151                 _statisticsCollectors->front()->~StatisticsCollector();
00152                 _statisticsCollectors->pop_front();
00153             }
00154         }
00155     }
00156 }
00157
00158 }
```

9.101 EntityType.h File Reference

```
#include <string>
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
#include "Plugin.h"

Include dependency graph for EntityType.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [EntityType](#)

9.102 EntityType.h

```

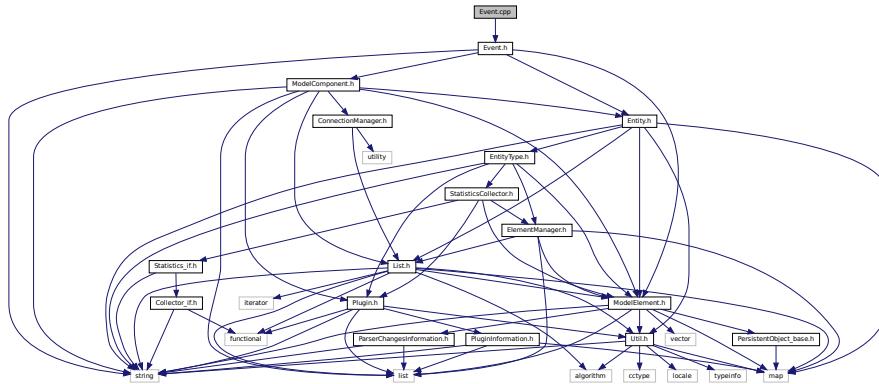
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   EntityType.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 19:24
00012 */
00013
00014 #ifndef ENTITYTYPE_H
00015 #define ENTITYTYPE_H
00016
00017 #include <string>
00018 #include "ModelElement.h"
00019 #include "StatisticsCollector.h"
00020 #include "ElementManager.h"
00021 #include "Plugin.h"
00022
00023 //##include "Model.h"
00024 //namespace GenesysKernel {
00025
00026     class EntityType : public ModelElement {
00027     public:
00028         //EntityType(Model* model);
00029         EntityType(Model* model, std::string name = "");
00030         virtual ~EntityType();
00031     public:
00032         virtual std::string show();
00033     public:
00034         static PluginInformation* GetPluginInformation();
00035         static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00036     public: //get & set
00037         void setInitialWaitingCost(double _initialWaitingCost);
00038         double initialWaitingCost() const;
00039         void setInitialOtherCost(double _initialOtherCost);
00040         double initialOtherCost() const;
00041         void setInitialNVACost(double _initialNVACost);
00042         double initialNVACost() const;
00043         void setInitialVACost(double _initialVACost);
00044         double initialVACost() const;
00045         void setInitialPicture(std::string _initialPicture);
00046         std::string initialPicture() const;
00047     public: //get
00048         StatisticsCollector* addGetStatisticsCollector(std::string name);
00049
00050     protected: // must be overridden by derived classes
00051         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052         virtual std::map<std::string, std::string>* _saveInstance();
00053         virtual bool _check(std::string* errorMessage);
00054         virtual void _initBetweenReplications();
00055         virtual void _createInternalElements();
00056     private:
00057         void _initCostsAndStatistics();
00058     private:
00059         std::string _initialPicture = "report";
00060         double _initialVACost = 0.0;
00061         double _initialNVACost = 0.0;
00062         double _initialOtherCost = 0.0;
00063         double _initialWaitingCost = 0.0;
00064     private: //1:

```

```
00065     List<StatisticsCollector*>* _statisticsCollectors = new List<StatisticsCollector*>();  
00066     };  
00067 //namespace\\}  
00068 #endif /* ENTITYTYPE_H */  
00069
```

9.103 Event.cpp File Reference

```
#include "Event.h"
Include dependency graph for Event.cpp:
```



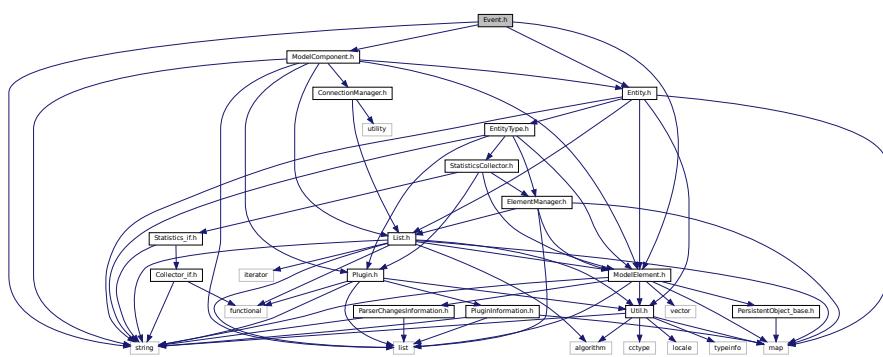
9.104 Event.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Event.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:41
00012 */
00013
00014 #include "Event.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 Event::Event(double time, Entity* entity, ModelComponent* component, unsigned int
    componentInputNumber) {
00019     _time = time;
00020     _entity = entity;
00021     _component = component;
00022     _componentInputNumber = componentInputNumber;
00023 }
00024
00025 Event::Event(double time, Entity* entity, Connection* connection) {
00026     _time = time;
00027     _entity = entity;
00028     _component = connection->first;
00029     _componentInputNumber = connection->second;
00030 }
00031
00032 std::string Event::show() {
00033     return "time=" + std::to_string(_time) +
00034         ",entity=" + std::to_string(_entity->entityNumber()) +
00035         ",comp=\"" + _component->getName() + "\""; //+std::to_string(_component->getId())+"}";
00036 }
00037
00038 unsigned int Event::getComponentInputNumber() const {
00039     return _componentInputNumber;
00040 }
00041
00042 double Event::getTime() const {
```

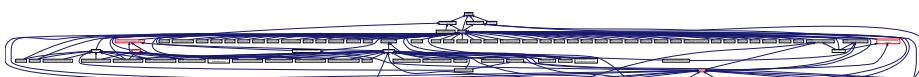
```
00043     return _time;
00044 }
00045
00046 ModelComponent* Event::getComponent() const {
00047     return _component;
00048 }
00049
00050 Entity* Event::getEntity() const {
00051     return _entity;
00052 }
00053
```

9.105 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

9.106 Event.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: Event.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 21 de Junho de 2018, 19:41  
00012 */  
00013  
00014 #ifndef EVENT_H  
00015 #define EVENT_H
```

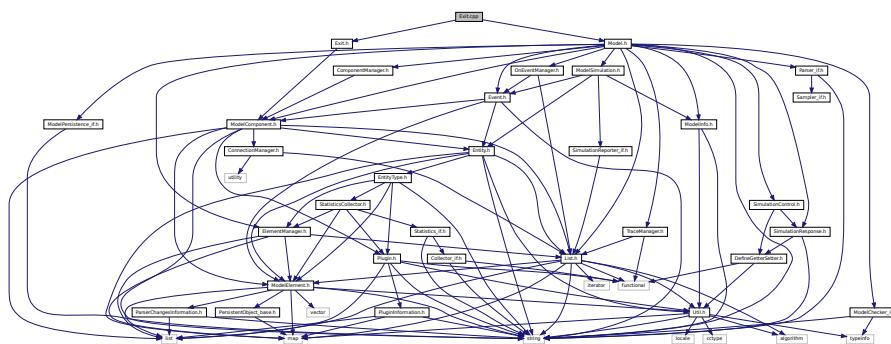
```

00016
00017 #include <string>
00018
00019 #include "ModelElement.h"
00020 #include "Entity.h"
00021 #include "ModelComponent.h"
00022
00023 //namespace GenesysKernel {
00024
00025     class Event {:// public ModelElement {
00026     public:
00027         Event(double time, Entity* entity, ModelComponent* component, unsigned int
00028               componentInputNumber = 0);
00029         Event(double time, Entity* entity, Connection* connection);
00030         virtual ~Event() = default;
00031     public:
00032         double getTime() const;
00033         ModelComponent* getComponent() const;
00034         Entity* getEntity() const;
00035         unsigned int getComponentInputNumber() const;
00036     public:
00037         std::string show();
00038     private:
00039         double _time;
00040         Entity* _entity;
00041         ModelComponent* _component;
00042         unsigned int _componentInputNumber;
00043     };
00044 //namespace\\}
00045 #endif /* EVENT_H */
00046

```

9.107 Exit.cpp File Reference

```
#include "Exit.h"
#include "Model.h"
Include dependency graph for Exit.cpp:
```



9.108 Exit.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Exit.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #include "Exit.h"
00015
00016 #include "Model.h"
00017
00018 Exit::Exit(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Exit>(), name) {

```

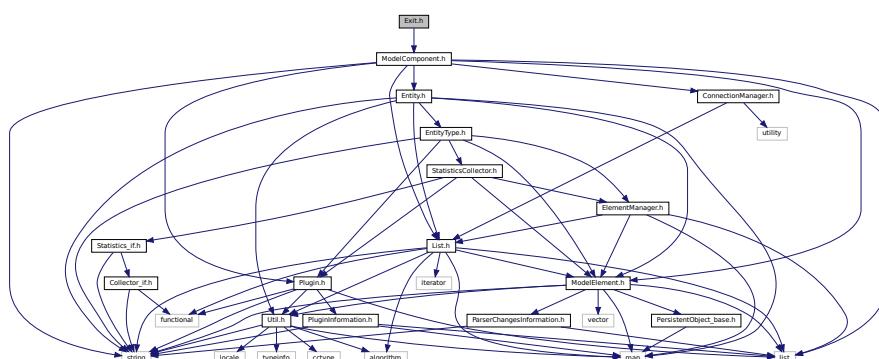
```

00019 }
00020
00021 std::string Exit::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Exit::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Exit* newComponent = new Exit(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
00034
00035 void Exit::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Exit::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Exit::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Exit::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Exit::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Exit::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Exit>(), &Exit::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

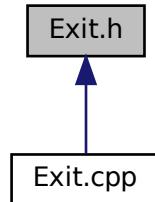
```

9.109 Exit.h File Reference

```
#include "ModelComponent.h"
Include dependency graph for Exit.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Exit](#)

9.110 Exit.h

```

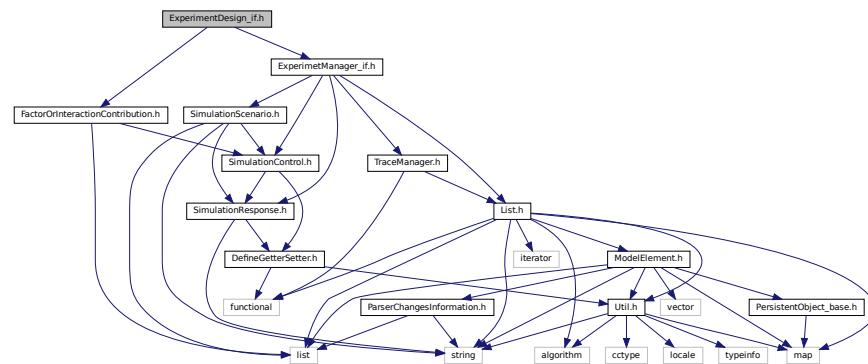
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Exit.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef EXIT_H
00015 #define EXIT_H
00016
00017 #include "ModelComponent.h"
00018
00037 class Exit : public ModelComponent {
00038 public: // constructors
00039     Exit(Model* model, std::string name = "");
00040     virtual ~Exit() = default;
00041 public: // virtual
00042     virtual std::string show();
00043 public: // static
00044     static PluginInformation* GetPluginInformation();
00045     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00046 protected: // virtual
00047     virtual void _execute(Entity* entity);
00048     virtual void _initBetweenReplications();
00049     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050     virtual std::map<std::string, std::string>* _saveInstance();
00051     virtual bool _check(std::string* errorMessage);
00052 private: // methods
00053 private: // attributes 1:1
00054 private: // attributes 1:n
00055 };
00056
00057
00058 #endif /* EXIT_H */
00059

```

9.111 ExperimentDesign_if.h File Reference

```
#include "FactorOrInteractionContribution.h"
```

```
#include "ExperimentManager_if.h"
Include dependency graph for ExperimentDesign_if.h:
```



This graph shows which files directly or indirectly include this file:



Classes

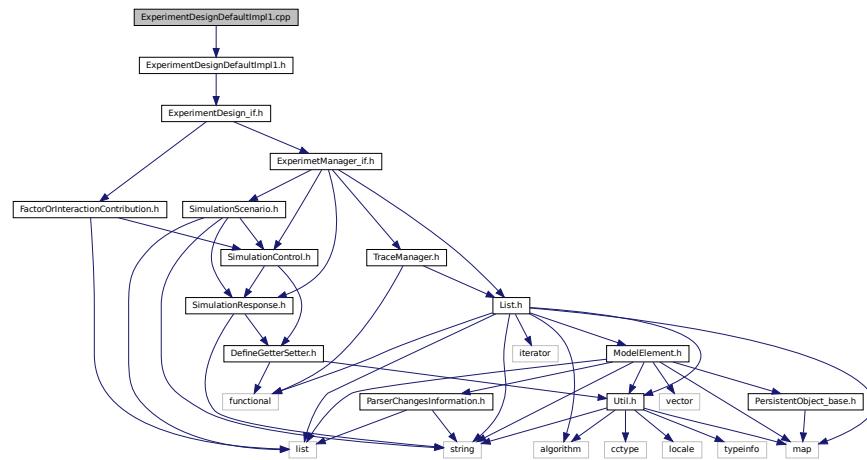
- class [ExperimentDesign_if](#)

9.112 ExperimentDesign_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File:   ExperimentDesign.h
00008  * Author: rafael.luiz.cancian
00009  *
00010  *
00011  * Created on 2 de Outubro de 2018, 22:47
00012 */
00013 #ifndef EXPERIMENTDESIGN_IF_H
00014 #define EXPERIMENTDESIGN_IF_H
00015
00016
00017 #include "FactorOrInteractionContribution.h"
00018 #include "ExperimentManager_if.h"
00019
00020 class ExperimentDesign_if {
00021 public:
00022     virtual ExperimentManager_if* getProcessAnalyser() const = 0;
00023     virtual bool generate2krScenarioExperiments() = 0;
00024     virtual bool calculateContributionAndCoefficients() = 0;
00025     virtual std::list<FactorOrInteractionContribution*>* getContributions() const = 0;
00026 };
00027
00028 #endif /* EXPERIMENTDESIGN_IF_H */
```

9.113 ExperimentDesignDefaultImpl1.cpp File Reference

```
#include "ExperimentDesignDefaultImpl1.h"
Include dependency graph for ExperimentDesignDefaultImpl1.cpp:
```



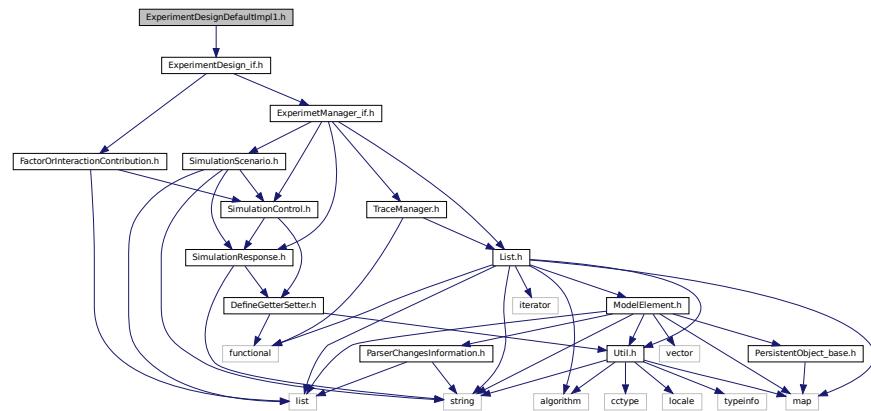
9.114 ExperimentDesignDefaultImpl1.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ExperimentDesignDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:19
00012 */
00013
00014 #include "ExperimentDesignDefaultImpl1.h"
00015
00016 ExperimentDesignDefaultImpl1::ExperimentDesignDefaultImpl1() {
00017 }
00018
00019 std::list<FactorOrInteractionContribution*>* ExperimentDesignDefaultImpl1::getContributions() const {
00020     return _contributions;
00021 }
00022
00023 bool ExperimentDesignDefaultImpl1::generate2krScenarioExperiments() {
00024     return true;
00025 }
00026
00027 bool ExperimentDesignDefaultImpl1::calculateContributionAndCoefficients() {
00028     return true;
00029 }
00030
00031 ExperimentManager_if* ExperimentDesignDefaultImpl1::getProcessAnalyser() const {
00032     return _processAnalyser;
00033 }
```

9.115 ExperimentDesignDefaultImpl1.h File Reference

```
#include "ExperimentDesign_if.h"
```

Include dependency graph for ExperimentDesignDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

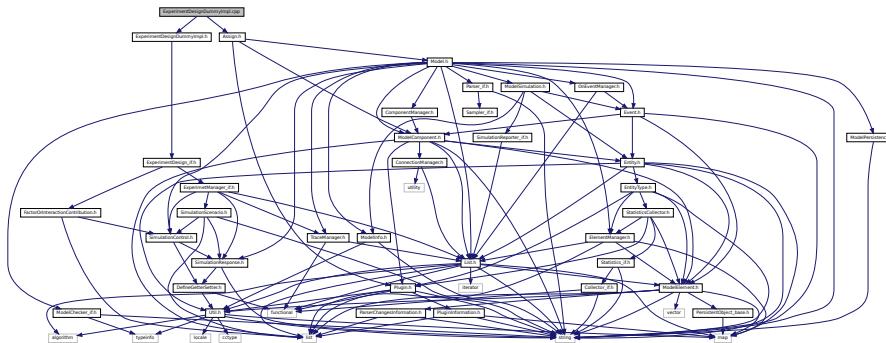
- class [ExperimentDesignDefaultImpl1](#)

9.116 ExperimentDesignDefaultImpl1.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ExperimentDesignDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:19
00012 */
00013
00014 #ifndef EXPERIMENTDESIGNDEFAULTIMPL1_H
00015 #define EXPERIMENTDESIGNDEFAULTIMPL1_H
00016
00017 #include "ExperimentDesign_if.h"
00018
00019 class ExperimentDesignDefaultImpl1 : public ExperimentDesign_if {
00020 public:
00021     ExperimentDesignDefaultImpl1();
00022     virtual ~ExperimentDesignDefaultImpl1() = default;
00023 public:
00024     virtual ExperimentManager_if* getProcessAnalyser() const;
00025 public:
00026     virtual bool generate2krScenarioExperiments();
00027     virtual bool calculateContributionAndCoefficients();
00028     virtual std::list<FactorOrInteractionContribution*>* getContributions() const;
00029 private:
00030     ExperimentManager_if* _processAnalyser; //= new
00031     Traits<ExperimentDesign_if>::ProcessAnalyserImplementation();
00032     std::list<FactorOrInteractionContribution*>* _contributions = new
00033     std::list<FactorOrInteractionContribution*>();
00034 };
00035#endif /* EXPERIMENTDESIGNDEFAULTIMPL1_H */
```

9.117 ExperimentDesignDummyImpl.cpp File Reference

```
#include "ExperimentDesignDummyImpl.h"
#include "Assign.h"
Include dependency graph for ExperimentDesignDummyImpl.cpp:
```



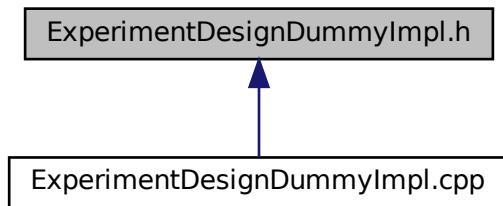
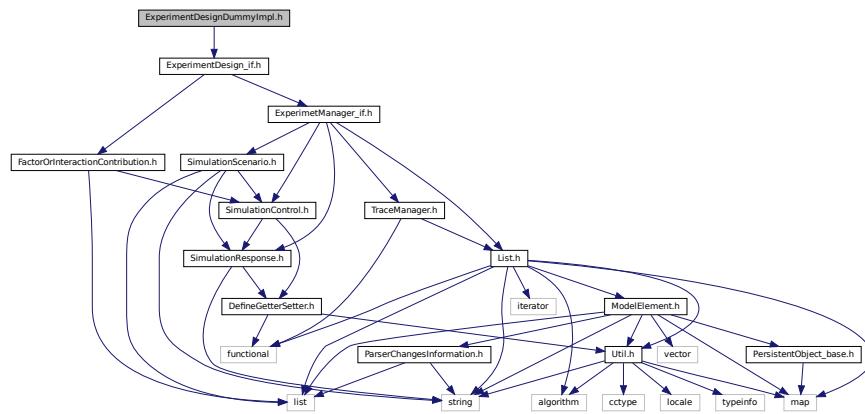
9.118 ExperimentDesignDummyImpl.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ExperimentDesignDummyImpl.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:32
00012 */
00013
00014 #include "ExperimentDesignDummyImpl.h"
00015 #include "Assign.h"
00016
00017 ExperimentDesignDummyImpl::ExperimentDesignDummyImpl() {
00018 }
00019
00020 std::list<FactorOrInteractionContribution*>* ExperimentDesignDummyImpl::getContributions() const {
00021     return _contributions;
00022 }
00023
00024 bool ExperimentDesignDummyImpl::generate2krScenarioExperiments() {
00025     return true;
00026 }
00027
00028 bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients() {
00029     return true;
00030 }
00031
00032 ExperimentManager_if* ExperimentDesignDummyImpl::getProcessAnalyser() const {
00033     return _processAnalyser;
00034 }
00035
```

9.119 ExperimentDesignDummyImpl.h File Reference

```
#include "ExperimentDesign_if.h"
```

Include dependency graph for ExperimentDesignDummyImpl.h:



Classes

- class [ExperimentDesignDummyImpl](#)

9.120 ExperimentDesignDummyImpl.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ExperimentDesignDummyImpl.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:32
00012 */
00013
00014 #ifndef EXPERIMENTDESIGNDUMMYIMPL_H
00015 #define EXPERIMENTDESIGNDUMMYIMPL_H
00016
00017 #include "ExperimentDesign_if.h"
00018
00022 class ExperimentDesignDummyImpl : public ExperimentDesign_if {
00023 public:
  
```

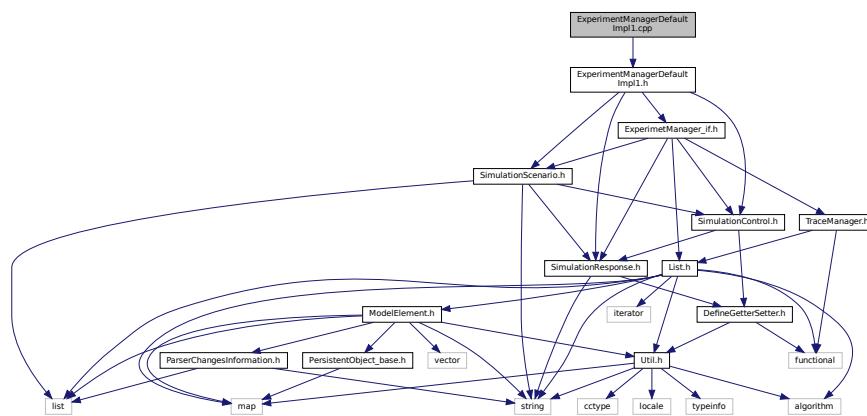
```

00024     ExperimentDesignDummyImpl();
00025     virtual ~ExperimentDesignDummyImpl() = default;
00026 public:
00027     virtual ExperimentManager_if* getProcessAnalyser() const;
00028 public:
00029     virtual bool generate2krScenarioExperiments();
00030     virtual bool calculateContributionAndCoefficients();
00031     virtual std::list<FactorOrInteractionContribution*>* getContributions() const;
00032 private:
00033     ExperimentManager_if* _processAnalyser; //= new
00034     Traits<ExperimentDesign_if>::ProcessAnalyserImplementation();
00035     std::list<FactorOrInteractionContribution*>* _contributions = new
00036     std::list<FactorOrInteractionContribution*>();
00037 };
00038 #endif /* EXPERIMENTDESIGNDUMMYIMPL_H */

```

9.121 ExperimentManagerDefaultImpl1.cpp File Reference

#include "ExperimentManagerDefaultImpl1.h"
Include dependency graph for ExperimentManagerDefaultImpl1.cpp:



9.122 ExperimentManagerDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ProcessAnalyserDefaultImpl1.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:45
00012 */
00013
00014 #include "ExperimentManagerDefaultImpl1.h"
00015
00016 ExperimentManagerDefaultImpl1::ExperimentManagerDefaultImpl1() {
00017 }
00018
00019 List<SimulationScenario*>* ExperimentManagerDefaultImpl1::getScenarios() const {
00020     // \todo: implement
00021 }
00022
00023 List<SimulationControl*>* ExperimentManagerDefaultImpl1::getControls() const {
00024     return _controls;
00025 }
00026
00027 List<SimulationResponse*>* ExperimentManagerDefaultImpl1::getResponses() const {
00028     // \todo: implement
00029 }
00030

```

```

00031 List<SimulationControl*>* ExperimentManagerDefaultImpl::extractControlsFromModel(std::string
00032     modelFilename) const {
00033     // \todo: implement
00034 }
00035 List<SimulationResponse*>* ExperimentManagerDefaultImpl::extractResponsesFromModel(std::string
00036     modelFilename) const {
00037     // \todo: implement
00038 }
00039 void ExperimentManagerDefaultImpl::startSimulationOfScenario(SimulationScenario* scenario) {
00040     // \todo: implement
00041 }
00042
00043 void ExperimentManagerDefaultImpl::startSimulation() {
00044     // \todo: implement
00045 }
00046
00047 void ExperimentManagerDefaultImpl::stopSimulation() {
00048     // \todo: implement
00049 }
00050
00051 void ExperimentManagerDefaultImpl::addTraceSimulationHandler(traceSimulationProcessListener
00052     traceSimulationProcessListener) {
00052 }

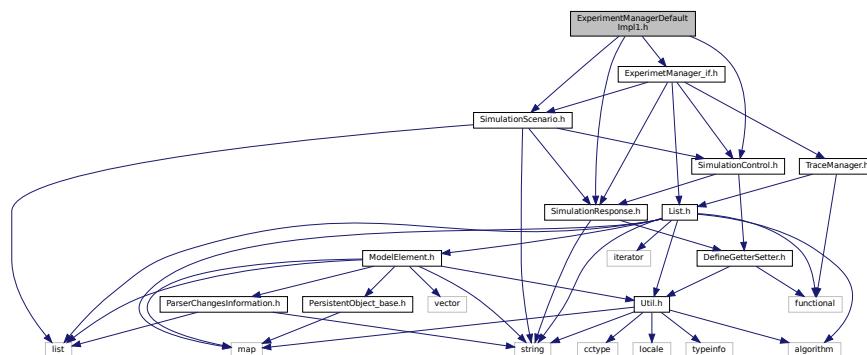
```

9.123 ExperimentManagerDefaultImpl1.h File Reference

```

#include "ExperimentManager_if.h"
#include "SimulationScenario.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"
Include dependency graph for ExperimentManagerDefaultImpl1.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class `ExperimentManagerDefaultImpl1`

9.124 ExperimentManagerDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: ProcessAnalyserDefaultImpl1.h
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:45
00012 */
00013
00014 #ifndef PROCESSANALYSERDEFAULTIMPL1_H
00015 #define PROCESSANALYSERDEFAULTIMPL1_H
00016
00017 #include "ExperimentManager_if.h"
00018 #include "SimulationScenario.h"
00019 #include "SimulationResponse.h"
00020 #include "SimulationControl.h"
00021
00022 class ExperimentManagerDefaultImpl1 : public ExperimentManager_if {
00023 public:
00024     ExperimentManagerDefaultImpl1();
00025     virtual ~ExperimentManagerDefaultImpl1() = default;
00026 public:
00027     virtual List<SimulationScenario*>* getScenarios() const;
00028     virtual List<SimulationControl*>* getControls() const;
00029     virtual List<SimulationResponse*>* getResponses() const;
00030     virtual List<SimulationControl*>* extractControlsFromModel(std::string modelName) const;
00031     virtual List<SimulationResponse*>* extractResponsesFromModel(std::string modelName) const;
00032     virtual void startSimulationOfScenario(SimulationScenario* scenario);
00033     virtual void startSimulation();
00034     virtual void stopSimulation();
00035     virtual void addTraceSimulationHandler(traceSimulationProcessListener
00036     traceSimulationProcessListener);
00037     List<SimulationControl*>* _controls = new List<SimulationControl*>();
00038 };
00039
00040 #endif /* PROCESSANALYSERDEFAULTIMPL1_H */
00041

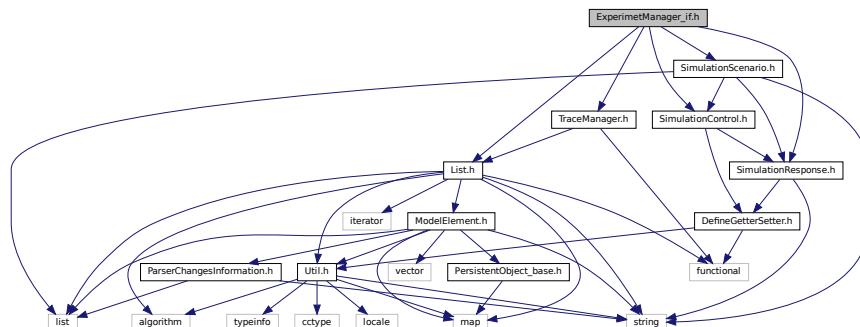
```

9.125 ExperimentManager_if.h File Reference

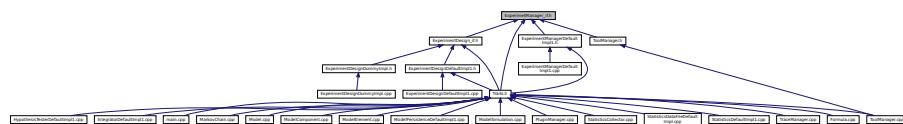
```

#include "List.h"
#include "SimulationScenario.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include "TraceManager.h"
Include dependency graph for ExperimentManager_if.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class `ExperimentManager_if`

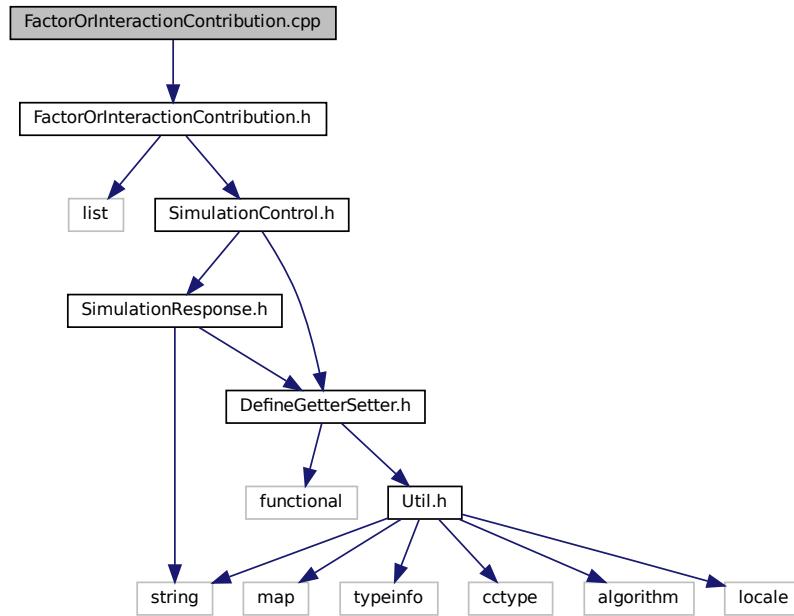
9.126 ExperimentManager_if.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ExperimentManager_if (old ProcessAnalyser_if.h)
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 14:26
00012 */
00013
00014 #ifndef EXPERIMENTMANAGER_IF_H
00015 #define EXPERIMENTMANAGER_IF_H
00016
00017 #include "List.h"
00018 #include "SimulationScenario.h"
00019 #include "SimulationControl.h"
00020 #include "SimulationResponse.h"
00021 #include "TraceManager.h"
00022
00023 class ExperimentManager_if {
00024 public:
00025     virtual List<SimulationScenario*>* getScenarios() const = 0;
00026     virtual List<SimulationControl*>* getControls() const = 0;
00027     virtual List<SimulationResponse*>* getResponses() const = 0;
00028     virtual List<SimulationControl*>* extractControlsFromModel(std::string modelName) const = 0;
00029     virtual List<SimulationResponse*>* extractResponsesFromModel(std::string modelName) const = 0;
00030     virtual void startSimulationOfScenario(SimulationScenario* scenario) = 0;
00031     virtual void startSimulation() = 0;
00032     virtual void stopSimulation() = 0;
00033     virtual void addTraceSimulationHandler(traceSimulationProcessListener
00034         traceSimulationProcessListener) = 0;
00035 };
00036
00037 #endif /* EXPERIMENTMANAGER_IF_H */
```

9.127 FactorOrInteractionContribution.cpp File Reference

```
#include "FactorOrInteractionContribution.h"
```

Include dependency graph for FactorOrInteractionContribution.cpp:



9.128 FactorOrInteractionContribution.cpp

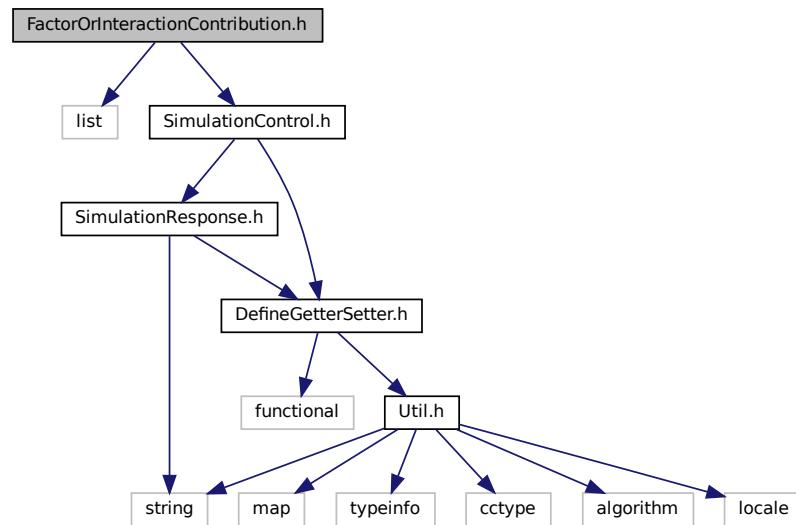
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: FactorOrInteractionContribution.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 20:08
00012 */
00013
00014 #include "FactorOrInteractionContribution.h"
00015
00016 FactorOrInteractionContribution::FactorOrInteractionContribution(double contribution, double
00017   modelCoefficient, std::list<SimulationControl*>* controls) {
00018   _contribution = contribution;
00019   _modelCoefficient = modelCoefficient;
00020   _controls = controls;
00021 }
00022 double FactorOrInteractionContribution::getModelCoefficient() const {
00023   return _modelCoefficient;
00024 }
00025
00026 std::list<SimulationControl*>* FactorOrInteractionContribution::getControls() const {
00027   return _controls;
00028 }
00029
00030 double FactorOrInteractionContribution::getContribution() const {
00031   return _contribution;
00032 }
  
```

9.129 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](#)

9.130 FactorOrInteractionContribution.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FactorOrInteractionContribution.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 20:08
00012 */
00013
00014 #ifndef FACTORORINTERACTIONCONTRIBUTION_H
00015 #define FACTORORINTERACTIONCONTRIBUTION_H
00016
00017 #include <list>
00018 #include "SimulationControl.h"
00019
00023 class FactorOrInteractionContribution {
00024 public:
00025     FactorOrInteractionContribution(double contribution, double modelCoefficient,
00026                                     std::list<SimulationControl*>* controls);

```

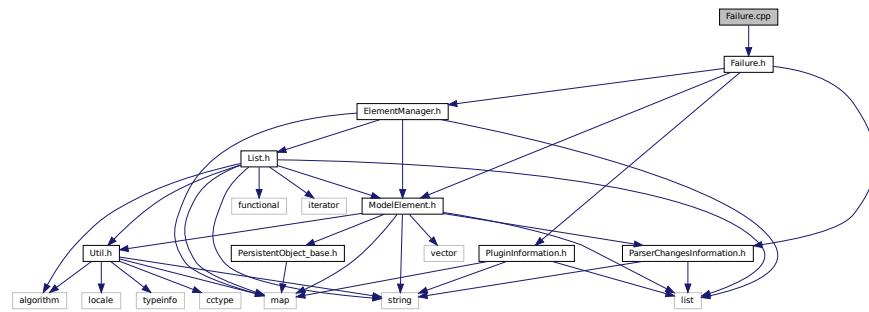
```

00026     ~FactorOrInteractionContribution();
00027 public:
00028     double getModelCoefficient() const;
00029     std::list<SimulationControl*>& getControls() const;
00030     double getContribution() const;
00031 private:
00032     double _contribution;
00033     double _modelCoefficient;
00034     std::list<SimulationControl*>& _controls;
00035 };
00036
00037 #endif /* FACTORORINTERACTIONCONTRIBUTION_H */
00038

```

9.131 Failure.cpp File Reference

#include "Failure.h"
Include dependency graph for Failure.cpp:



9.132 Failure.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Failure.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 20 de Failureembro de 2019, 20:07
00012 */
00013
00014 #include "Failure.h"
00015
00016 Failure::Failure(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Failure>(), name)
00017 {
00018 }
00019 std::string Failure::show() {
00020     return ModelElement::show() +
00021         "";
00022 }
00023
00024 PluginInformation* Failure::GetPluginInformation() {
00025     PluginInformation* info = new PluginInformation(Util::TypeOf<Failure>(), &Failure::LoadInstance);
00026     return info;
00027 }
00028
00029 ModelElement* Failure::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00030     Failure* newElement = new Failure(model);
00031     try {
00032         newElement->_loadInstance(fields);
00033     } catch (const std::exception& e) {
00034     }
00035
00036     return newElement;

```

```

00037 }
00038
00039 bool Failure::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelElement::_loadInstance(fields);
00041     if (res) {
00042         try {
00043             //this->_attributeName = (*fields->find("attributeName")).second;
00044             //this->_orderRule = static_cast<OrderRule>
00045             (std::stoi((*fields->find("orderRule")).second));
00046         } catch (...) {
00047     }
00048     return res;
00049 }
00050
00051 std::map<std::string, std::string>* Failure::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00053     //Util::TypeOf<Failure>());
00054     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055     //fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00056     return fields;
00057 }
00058 bool Failure::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     // resultAll |= ...
00061     return resultAll;
00062 }
00063
00064 ParserChangesInformation* Failure::_getParserChangesInformation() {
00065     ParserChangesInformation* changes = new ParserChangesInformation();
00066     //changes->getProductionToAdd()->insert(...);
00067     //changes->getTokensToAdd()->insert(...);
00068     return changes;
00069 }

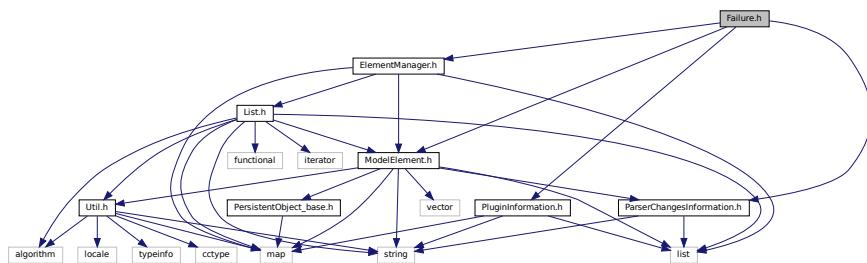
```

9.133 Failure.h File Reference

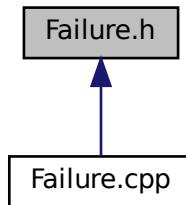
```

#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
Include dependency graph for Failure.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [Failure](#)

9.134 Failure.h

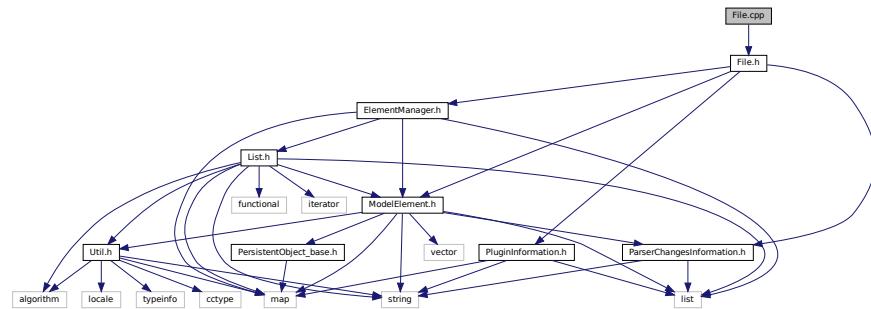
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Failure.h
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Failureembro de 2019, 20:07
00012 */
00013
00014 #ifndef FAILURE_H
00015 #define FAILURE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00067 class Failure : public ModelElement {
00068 public:
00069     Failure(Model* model, std::string name = "");
00070     virtual ~Failure() = default;
00071 public: // static
00072     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00073     static PluginInformation* GetPluginInformation();
00074 protected:
00075     virtual std::string show();
00076
00077 protected: // must be overriden by derived classes
00078     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00079     virtual std::map<std::string, std::string>* _saveInstance();
00080 protected: // could be overriden by derived classes
00081     virtual bool _check(std::string* errorMessage);
00082     virtual ParserChangesInformation* _getParserChangesInformation();
00083
00084 };
00085
00086 #endif /* FAILURE_H */
00087
  
```

9.135 File.cpp File Reference

```
#include "File.h"
```

Include dependency graph for File.cpp:



9.136 File.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: File.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Fileembro de 2019, 20:07
00012 */
00013
00014 #include "File.h"
00015
00016 File::File(Model* model, std::string name) : ModelElement(model, Util::TypeOf<File>(), name) {
00017     //elems = elems;
00018 }
00019
00020 std::string File::show() {
00021     return ModelElement::show() +
00022         "";
00023 }
00024
00025 PluginInformation* File::GetPluginInformation() {
00026     PluginInformation* info = new PluginInformation(Util::TypeOf<File>(), &File::LoadInstance);
00027     return info;
00028 }
00029
00030 ModelElement* File::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00031     File* newElement = new File(model);
00032     try {
00033         newElement->_loadInstance(fields);
00034     } catch (const std::exception& e) {
00035     }
00036     return newElement;
00037 }
00038
00039
00040 bool File::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelElement::_loadInstance(fields);
00042     if (res) {
00043         try {
00044             //this->_attributeName = (*fields->find("attributeName")).second;
00045             //this->_orderRule = static_cast<OrderRule>
00046             (std::stoi((*fields->find("orderRule")).second));
00047         } catch (...) {
00048     }
00049     return res;
00050 }
00051
00052 std::map<std::string, std::string>* File::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00054     //Util::TypeOf<File>());
00055 }
```

```

00054     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055     //fields->emplace("attributeName", "\"" +this->_attributeName+"\"");
00056     return fields;
00057 }
00058
00059 bool File::_check(std::string* errorMessage) {
00060     bool resultAll = true;
00061     // resultAll |= ...
00062     return resultAll;
00063 }
00064
00065 ParserChangesInformation* File::_getParserChangesInformation() {
00066     ParserChangesInformation* changes = new ParserChangesInformation();
00067     //changes->getProductionToAdd()->insert(...);
00068     //changes->getTokensToAdd()->insert(...);
00069     return changes;
00070 }
00071

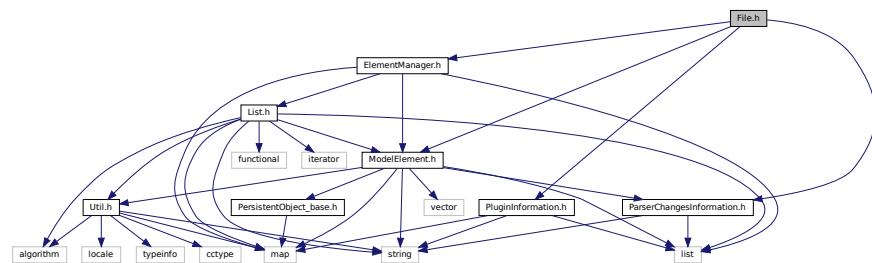
```

9.137 File.h File Reference

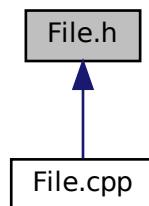
```

#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
Include dependency graph for File.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [File](#)

9.138 File.h

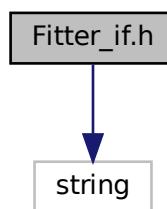
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: File.h
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Fileembro de 2019, 20:07
00012 */
00013
00014 #ifndef FILE_H
00015 #define FILE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00064 class File : public ModelElement {
00065 public:
00066     File(Model* model, std::string name = "");
00067     virtual ~File() = default;
00068 public: // static
00069     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00070     static PluginInformation* GetPluginInformation();
00071 public:
00072     virtual std::string show();
00073
00074 protected: // must be overridden by derived classes
00075     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00076     virtual std::map<std::string, std::string>* _saveInstance();
00077 protected: // could be overridden by derived classes
00078     virtual bool _check(std::string* errorMessage);
00079     virtual ParserChangesInformation* _getParserChangesInformation();
00080
00081 };
00082
00083 #endif /* FILE_H */
00084

```

9.139 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](#)

9.140 Fitter_if.h

```

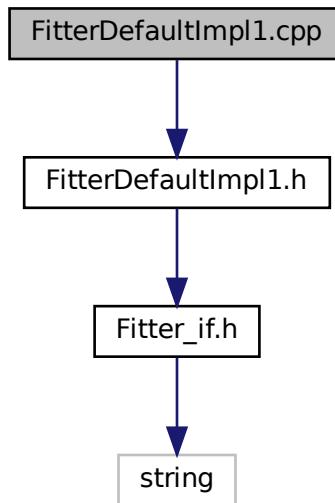
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Fitter_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 14:05
00012 */
00013
00014 #ifndef FITTER_IF_H
00015 #define FITTER_IF_H
00016
00017 #include <string>
00018
00019 class Fitter_if {
00020 public:
00021     virtual bool isNormalDistributed(double confidencelevel) = 0;
00022     virtual void fitUniform(double *sqrerror, double *min, double *max) = 0;
00023     virtual void fitTriangular(double *sqrerror, double *min, double *mo, double *max) = 0;
00024     virtual void fitNormal(double *sqrerror, double *avg, double *stddev) = 0;
00025     virtual void fitExpo(double *sqrerror, double *avg1) = 0;
00026     virtual void fitErlang(double *sqrerror, double *avg, double *m) = 0;
00027     virtual void fitBeta(double *sqrerror, double *alpha, double *beta, double *infLimit, double
00028                           *supLimit) = 0;
00029     virtual void fitWeibull(double *sqrerror, double *alpha, double *scale) = 0;
00030     virtual void fitAll(double *sqrerror, std::string *name) = 0;
00031 public:
00032     virtual void setDataFilename(std::string dataFilename) = 0;
00033     virtual std::string getDataFilename() = 0;
00034 };
00035 #endif /* FITTER_IF_H */
00036

```

9.141 FitterDefaultImpl1.cpp File Reference

```
#include "FitterDefaultImpl1.h"
```

Include dependency graph for FitterDefaultImpl1.cpp:



9.142 FitterDefaultImpl1.cpp

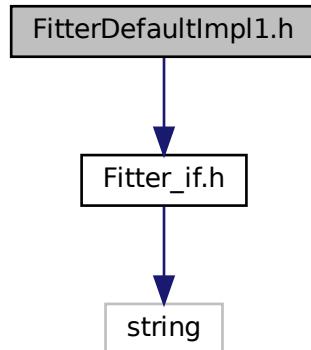
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:   FitterDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:48
00012 */
00013
00014 #include "FitterDefaultImpl1.h"
00015
00016 FitterDefaultImpl1::FitterDefaultImpl1() {
00017 }
00018
00019 bool FitterDefaultImpl1::isNormalDistributed(double confidencelevel) {
00020     return true;
00021 }
00022
00023 void FitterDefaultImpl1::fitUniform(double *sqrerror, double *min, double *max) {
00024
00025 }
00026
00027 void FitterDefaultImpl1::fitTriangular(double *sqrerror, double *min, double *mo, double *max) {
00028
00029 }
00030
00031 void FitterDefaultImpl1::fitNormal(double *sqrerror, double *avg, double *stddev) {
00032
00033 }
00034
00035 void FitterDefaultImpl1::fitExpo(double *sqrerror, double *avg1) {
00036 }
00037
00038 void FitterDefaultImpl1::fitErlang(double *sqrerror, double *avg, double *m) {
00039
00040 }
00041
00042 void FitterDefaultImpl1::fitBeta(double *sqrerror, double *alpha, double *beta, double *infLimit,
00043                                     double *supLimit) {
  
```

```
00044 }
00045
00046 void FitterDefaultImpl::fitWeibull(double *sqrerror, double *alpha, double *scale) {
00047
00048 }
00049
00050 void FitterDefaultImpl::fitAll(double *sqrerror, std::string *name) {
00051
00052 }
00053
00054 void FitterDefaultImpl::setDataFilename(std::string dataFilename) {
00055
00056 }
00057
00058 std::string FitterDefaultImpl::getDataFilename() {
00059     return ""; // \todo
00060 }
```

9.143 FitterDefaultImpl1.h File Reference

```
#include "Fitter_if.h"
Include dependency graph for FitterDefaultImpl1.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [FitterDefaultImpl1](#)

9.144 FitterDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: FitterDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:48
00012 */
00013
00014 #ifndef FITTERDEFAULTIMPL1_H
00015 #define FITTERDEFAULTIMPL1_H
00016
00017 #include "Fitter_if.h"
00018
00019 class FitterDefaultImpl1 : public Fitter_if {
00020 public:
00021     FitterDefaultImpl1();
00022     virtual ~FitterDefaultImpl1() = default;
00023 public:
00024     bool isNormalDistributed(double confidencelevel);
00025     void fitUniform(double *sqrerror, double *min, double *max);
00026     void fitTriangular(double *sqrerror, double *min, double *mo, double *max);
00027     void fitNormal(double *sqrerror, double *avg, double *stddev);
00028     void fitExpo(double *sqrerror, double *avg1);
00029     void fitErlang(double *sqrerror, double *avg, double *m);
00030     void fitBeta(double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit);
00031     void fitWeibull(double *sqrerror, double *alpha, double *scale);
00032     void fitAll(double *sqrerror, std::string *name);
00033 public:
00034     void setDataFilename(std::string dataFilename);
00035     std::string getDataFilename();
00036 private:
00037     std::string _dataFilename = "";
00038 };
00039
00040 #endif /* FITTERDEFAULTIMPL1_H */
00041

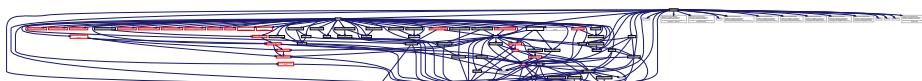
```

9.145 Formula.cpp File Reference

```

#include "Formula.h"
#include <iostream>
#include "ElementManager.h"
#include "Model.h"
#include "Traits.h"
Include dependency graph for Formula.cpp:

```



9.146 Formula.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Formula.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Junho de 2019, 00:56
00012 */
00013
00014 #include "Formula.h"
00015 #include <iostream>

```

```
00016 #include "ElementManager.h"
00017 #include "Model.h"
00018 #include "Traits.h"
00019
00020 Formula::Formula(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Formula>(), name)
00021     {  
00022         // _myPrivateParser = new Traits<Parser_if>::Implementation(_parentModel);  
00023     }
00024
00025     std::string Formula::show() {
00026         std::string expressions = "";
00027         unsigned int i = 0;
00028         // for (std::list<std::string>::iterator it = _formulaExpressions->list()->begin(); it !=  
         _formulaExpressions->list()->end(); it++) {  
00029             //expressions += "expression[" + std::to_string(i++) + "]=" + (*it) + "\"; "  
00030         //}
00031         return ModelElement::show() + expressions;
00032     }
00033
00034     unsigned int Formula::size() {
00035         return _formulaExpressions->size();
00036     }
00037
00038     void Formula::setExpression(std::string index, std::string formulaExpression) {
00039         std::map<std::string, std::string>::iterator mapIt = _formulaExpressions->find(index);
00040         if (mapIt != _formulaExpressions->end()) //found
00041             (*mapIt).second = formulaExpression;
00042         else //not found
00043             _formulaExpressions->insert({index, formulaExpression});
00044     }
00045
00046     std::string Formula::expression(std::string index) {
00047         std::map<std::string, std::string>::iterator it = _formulaExpressions->find(index);
00048         if (it == _formulaExpressions->end())
00049             return ""; // index does not exist. No formula expressions returned. \todo: Should it be  
traced?.
00050         else {
00051             return it->second;
00052         }
00053     }
00054
00055     void Formula::setExpression(std::string formulaExpression) {
00056         setExpression("", formulaExpression);
00057     }
00058
00059     std::string Formula::expression() {
00060         return expression("");
00061     }
00062
00063     double Formula::value(std::string index) {
00064         std::string strexpression = this->expression(index);
00065         double value = 0.0;
00066         try {
00067             value = _parentModel->parseExpression(strexpression);
00068         } catch (const std::exception& e) {
00069             _parentModel->getTracer()->traceError(e, "Error parsing formula \" " + strexpression + " \"");
00070         }
00071         return value;
00072     }
00073
00074     double Formula::value() {
00075         return value("");
00076     }
00077
00078     PluginInformation* Formula::GetPluginInformation() {
00079         PluginInformation* info = new PluginInformation(Util::TypeOf<Formula>(), &Formula::LoadInstance);
00080         return info;
00081     }
00082
00083     ModelElement* Formula::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00084         ModelElement* newElement = new Formula(model);
00085         try {
00086             newElement->_loadInstance(fields);
00087         } catch (const std::exception& e) {
00088
00089         }
00090         return newElement;
00091     }
00092
00093     bool Formula::_loadInstance(std::map<std::string, std::string>* fields) {
00094         return ModelElement::_loadInstance(fields);
00095     }
00096
00097     std::map<std::string, std::string>* Formula::_saveInstance() {
00098         std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00099         //fields->emplace("...", std::to_string(this->...));

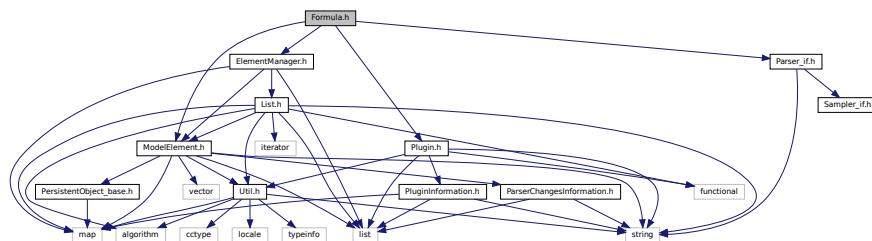
```

```

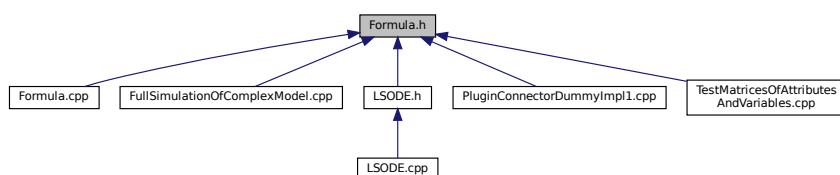
00100     return fields;
00101 }
00102
00103 bool Formula::_check(std::string* errorMessage) {
00104     std::string errorMsg = "";
00105     bool res, resAll = true;
00106     //unsigned int i = 0;
00107     for (std::map<std::string, std::string>::iterator it = _formulaExpressions->begin(); it != _formulaExpressions->end(); it++) {
00108         res = _parentModel->checkExpression((*it).second, "formula expression[" + (*it).first + "]", &errorMsg);
00109         if (!res) {
00110             _parentModel->getTracer()->trace(Util::TraceLevel::errorFatal, "Error parsing expression
00111 \\" + (*it).second + "\\"");
00112         }
00113     }
00114     return resAll;
00115 }
```

9.147 Formula.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
#include "Parser_if.h"
Include dependency graph for Formula.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Formula](#)

9.148 Formula.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Formula.h
00009  * Author:  rlcancian
00010 *
00011 * Created on 20 de Junho de 2019, 00:56
00012 */
00013
00014 #ifndef FORMULA_H
00015 #define FORMULA_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020 #include "Parser_if.h"
00021
00022 class Formula : public ModelElement {
00023 public:
00024     Formula(Model* model, std::string name = "");
00025     virtual ~Formula() = default;
00026 public: // virtual
00027     virtual std::string show();
00028 public:
00029     unsigned int size();
00030     void setExpression(std::string index, std::string formulaExpression);
00031     void setExpression(std::string formulaExpression);
00032     std::string expression(std::string index);
00033     std::string expression();
00034     double value();
00035     double value(std::string index);
00036 public: // statics
00037     static PluginInformation* GetPluginInformation();
00038     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00039 protected: // must be overriden by derived classes
00040     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00041     virtual std::map<std::string, std::string>* _saveInstance();
00042     virtual bool _check(std::string* errorMessage);
00043 private:
00044 private:
00045     std::map<std::string, std::string>* _formulaExpressions = new std::map<std::string,
00046         std::string>(); // map<index, formula>
00047 };
00048 #endif /* FORMULA_H */
00049

```

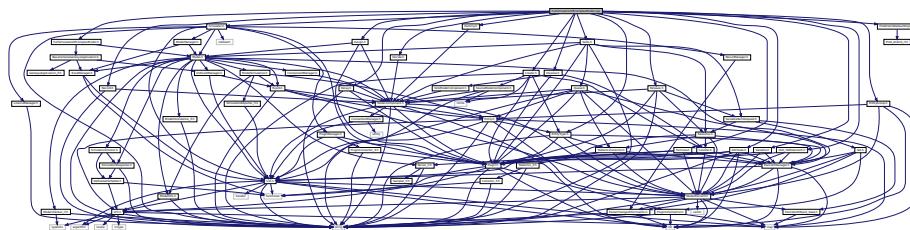
9.149 FullSimulationOfComplexModel.cpp File Reference

```

#include "FullSimulationOfComplexModel.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Dummy.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistribDefaultImpl.h"
#include "EntityGroup.h"
#include "Formula.h"

```

```
#include "OLD_ODElement.h"
Include dependency graph for FullSimulationOfComplexModel.cpp:
```



9.150 FullSimulationOfComplexModel.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 #include "FullSimulationOfComplexModel.h"
00008
00009 // GEnSys Simulator
00010 #include "Simulator.h"
00011
00012 // Model Components
00013 #include "Create.h"
00014 #include "Delay.h"
00015 #include "Dispose.h"
00016 #include "Seize.h"
00017 #include "Release.h"
00018 #include "Assign.h"
00019 #include "Record.h"
00020 #include "Decide.h"
00021 #include "Dummy.h"
00022
00023 // Model elements
00024 #include "EntityType.h"
00025 #include "Attribute.h"
00026 #include "Variable.h"
00027 #include "ProbDistribDefaultImpl.h"
00028 #include "EntityGroup.h"
00029 #include "Formula.h"
00030 #include "OLD_ODElement.h"
00031
00032 FullSimulationOfComplexModel::FullSimulationOfComplexModel() {
00033
00034 }
00035
00040 int FullSimulationOfComplexModel::main(int argc, char** argv) {
00041     Simulator* genesys = new Simulator();
00042     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00043     this->insertFakePluginsByHand(genesys);
00044     // creates an empty model
00045     Model* model = genesys->getModels()->newModel();
00046     // Handle traces and simulation events to output them
00047     TraceManager* tm = model->getTracer();
00048     this->setDefaultTraceHandlers(tm);
00049     // get easy access to classes used to insert components and elements into a model
00050     ComponentManager* components = model->getComponents();
00051     ElementManager* elements = model->getElements();
00052     //
00053     // build the simulation model
00054     //
00055     ModelInfo* infos = model->getInfos();
00056     infos->setAnalystName("Your name");
00057     infos->setProjectTitle("The title of the project");
00058     infos->setDescription("This simulation model tests the components and elements that have been
00059     implemented so far.");
00060     ModelSimulation* sim = model->getSimulation();
00061     sim->setReplicationLength(1e4);
00062     sim->setReplicationLengthTimeUnit(Util::TimeUnit::minute);
00063     sim->setNumberOfReplications(3000);
00064     tm->setTraceLevel(Util::TraceLevel::modelResult);
00065
00066     EntityType* entityTypel = new EntityType(model, "Representative_EntityType");
00067 }
```

```

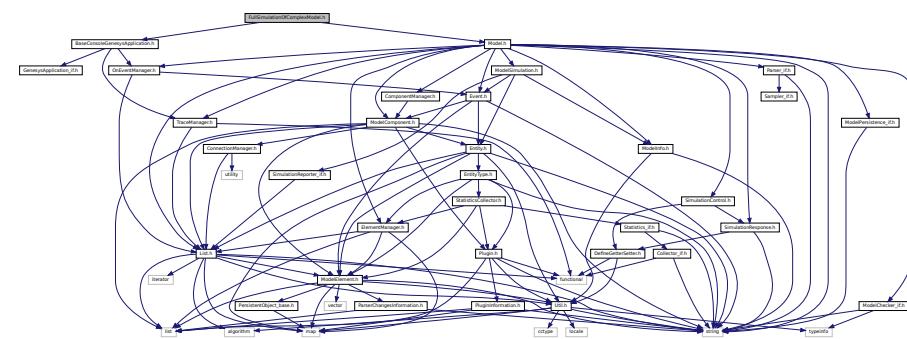
00068     Create* create1 = new Create(model);
00069     create1->setEntityType(entityType1);
00070     create1->setTimeBetweenCreationsExpression("EXPO(5)");
00071     create1-> setTimeUnit(Util::TimeUnit::minute);
00072     create1->setEntitiesPerCreation(1);
00073     components->insert(create1);
00074
00075     Attribute* attribute1 = new Attribute(model, "Attribute_1");
00076     Variable* variable1 = new Variable(model, "Variable_1");
00077
00078     Assign* assign1 = new Assign(model);
00079     Assign::Assignment* attrib2Assignment = new Assign::Assignment("Variable_1", "Variable_1 + 1");
00080     assign1->getAssignments()->insert(attrib2Assignment);
00081     Assign::Assignment* attrib1Assignment = new Assign::Assignment("Attribute_1", "Variable_1");
00082     assign1->getAssignments()->insert(attrib1Assignment);
00083
00084     Decide* decide1 = new Decide(model);
00085     decide1->getConditions()->insert("UNIF(0,1) > 0.5");
00086
00087     Resource* maquina1 = new Resource(model, "Máquina 1");
00088     maquina1->setCapacity(1);
00089
00090     Queue* filaSeize1 = new Queue(model);
00091     filaSeize1->setOrderRule(Queue::OrderRule::FIFO);
00092
00093     Seize* seize1 = new Seize(model);
00094     seize1->getSeizeRequests()->insert(new SeizableItemRequest(maquina1));
00095     seize1->setQueue(filaSeize1);
00096
00097     Delay* delay1 = new Delay(model);
00098     delay1->setDelayExpression("NORM(5, 3)");
00099     delay1-> setTimeUnit(Util::TimeUnit::minute);
00100
00101     Release* release1 = new Release(model);
00102     release1->getReleaseRequests()->insert(new SeizableItemRequest(maquina1));
00103
00104     Record* record1 = new Record(model);
00105     record1->setExpressionName("Tempo total no sistema");
00106     record1->setExpression("TNOW - Entity.ArrivalTime");
00107     record1->setFilename("./temp/TotalTimeInSystem.txt");
00108
00109     Dispose* dispose1 = new Dispose(model);
00110
00111     // connect model components to create a "workflow" -- should always start from a
00112     // SourceModelComponent and end at a SinkModelComponent (it will be checked)
00113     create1->getNextComponents()->insert(assign1);
00114     assign1->getNextComponents()->insert(decide1);
00115     decide1->getNextComponents()->insert(seize1);
00116     seize1->getNextComponents()->insert(delay1);
00117     delay1->getNextComponents()->insert(release1);
00118     release1->getNextComponents()->insert(record1);
00119     record1->getNextComponents()->insert(dispose1);
00120     // then save the model into a text file
00121     model->save("./models/AssignWrite3Seizes.txt");
00122     // execute the simulation
00123     model->getSimulation()->start();
00124
00125     return 0;
00126 }

```

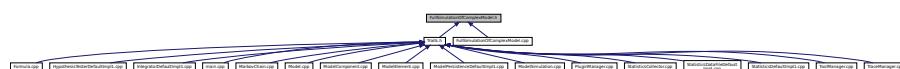
9.151 FullSimulationOfComplexModel.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
#include "Model.h"
```

Include dependency graph for FullSimulationOfComplexModel.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `FullSimulationOfComplexModel`

9.152 FullSimulationOfComplexModel.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: buildSimpleModel1.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 2 de Outubro de 2018, 19:18  
00012 */  
00013  
00014 #ifndef BUILDSIMULATIONMODEL02_H  
00015 #define BUILDSIMULATIONMODEL02_H  
00016  
00017 #include "BaseConsoleGenesysApplication.h"  
00018 #include "Model.h"  
00019  
00020 class FullSimulationOfComplexModel : public BaseConsoleGenesysApplication {  
00021 public:  
00022     FullSimulationOfComplexModel();  
00023 public:  
00024     virtual int main(int argc, char** argv);  
00025 };  
00026  
00027 #endif /* BUILDSIMULATIONMODEL02_H */  
00028
```

9.153 Functor.h File Reference

9.154 Functor.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 */
```

```

00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Functor.h
00009 * Author: joao vicentesouto
00010 *
00011 * Created on November 3, 2018, 11:04 AM
00012 */
00013
00014 #ifndef FUNCTOR_H
00015 #define FUNCTOR_H
00016
00017 #endif /* FUNCTOR_H */

```

9.155 GenesysApplication_if.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [GenesysApplication_if](#)

9.156 GenesysApplication_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: GenesysApplication_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Outubro de 2018, 19:14
00012 */
00013
00014 #ifndef GENESYSAPPLICATION_IF_H
00015 #define GENESYSAPPLICATION_IF_H
00016
00017 class GenesysApplication_if {
00018 public:
00019     virtual int main(int argc, char** argv) = 0;
00020 };
00021
00022 #endif /* GENESYSAPPLICATION_IF_H */
00023

```

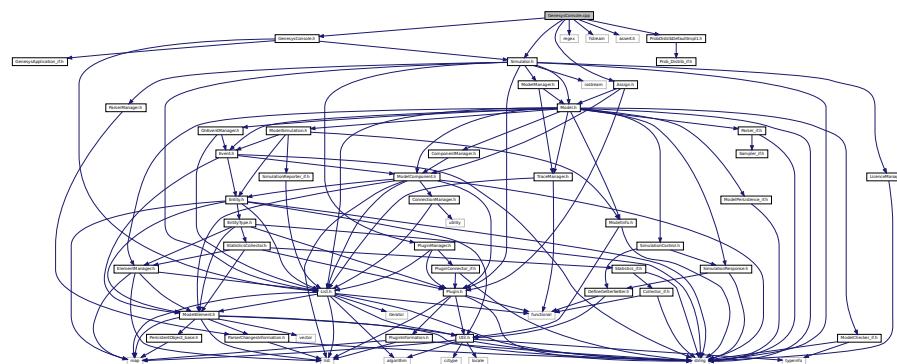
9.157 GenesysConsole.cpp File Reference

```

#include "GenesysConsole.h"
#include "Simulator.h"
#include "Assign.h"
#include <regex>
#include <fstream>
#include <assert.h>

```

```
#include "ProbDistribDefaultImpl1.h"
Include dependency graph for GenesysConsole.cpp:
```



9.158 GenesysConsole.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: GenesysShell.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:02
00012 */
00013
00014 #include "GenesysConsole.h"
00015 #include "Simulator.h"
00016 #include "Assign.h"
00017 #include <regex>
00018 #include <fstream>
00019 #include <assert.h>
00020
00021 #include "ProbDistribDefaultImpl1.h"
00022
00023 GenesysConsole::GenesysConsole() {
00024     _commands->setSortFunc([](const ShellCommand* a, const ShellCommand * b) {
00025         return a->shortname < b->shortname;
00026     });
00027     _commands->insert(new ShellCommand("q", "quit", "", "Quit ReGenesys shell. Same as exit.",
00028 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00029     _commands->insert(new ShellCommand("x", "exit", "", "Exit ReGenesys shell. Same as quit.",
00030 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00031     _commands->insert(new ShellCommand("h", "help", "", "Show help for commands.",
00032 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdHelp)));
00033     _commands->insert(new ShellCommand("ms", "modelsave", "<filename>", "Save model.",
00034 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelSave)));
00035     _commands->insert(new ShellCommand("ml", "modelload", "<filename>", "Load Model.",
00036 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelLoad)));
00037     _commands->insert(new ShellCommand("rf", "readfile", "<filename>", "Read and execute shell command
from file.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdScript)));
00038     _commands->insert(new ShellCommand("v", "version", "", "Show the version.",
00039 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdVersion)));
00040     _commands->insert(new ShellCommand("ss", "start", "", "Start simulation.",
00041 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStart)));
00042     _commands->insert(new ShellCommand("mc", "modelcheck", "", "Check model.",
00043 DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelCheck)));
00044     _commands->insert(new ShellCommand("mh", "modelshow", "", "Show model.",
DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelShow)));
00045     _commands->insert(new ShellCommand("ps", "step", "", "step simulation.",
DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStep)));
00046     _commands->insert(new ShellCommand("ts", "stop", "", "Stop simulation.",
DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStop)));
00047     _commands->insert(new ShellCommand("sr", "showreport", "", "Show simulation report.",
DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdShowReport)));
00048     _commands->insert(new ShellCommand("tl", "tracelevel", "<0|1|2|...|7>", "Set the trace level (the
bigger the most verbose).", DefineExecuterMember<GenesysConsole>(this,
&GenesysConsole::cmdTraceLevel)));
00049 }
00050
00051 void GenesysConsole::cmdTraceLevel() {
```

```

00044     Trace("Set trace level");
00045     try {
00046         int tlnum = std::stoi(_parameter);
00047         Util::TraceLevel tl = static_cast<Util::TraceLevel>(tlnum);
00048         _simulator->getModels()->current()->getTracer()->setTraceLevel(tl);
00049     } catch (...) {
00050         Trace("Error setting trace level");
00051     }
00052 }
00053
00054 void GenesysConsole::cmdModelCheck() {
00055     Trace("Check model");
00056     try {
00057         _simulator->getModels()->current()->check();
00058     } catch (...) {
00059         Trace("Error checking model");
00060     }
00061 }
00062
00063 void GenesysConsole::cmdStart() {
00064     Trace("Start simulation");
00065     try {
00066         _simulator->getModels()->current()->getSimulation()->start();
00067     } catch (...) {
00068         Trace("Error starting simulation");
00069     }
00070 }
00071
00072 void GenesysConsole::cmdStep() {
00073     Trace("Step simulation");
00074     try {
00075         _simulator->getModels()->current()->getSimulation()->step();
00076     } catch (...) {
00077         Trace("Error stepping simulation");
00078     }
00079 }
00080
00081 void GenesysConsole::cmdStop() {
00082     Trace("Stop simulation");
00083     try {
00084         _simulator->getModels()->current()->getSimulation()->stop();
00085     } catch (...) {
00086         Trace("Error stopping simulation");
00087     }
00088 }
00089
00090 void GenesysConsole::cmdShowReport() {
00091     Trace("Show report");
00092     try {
00093         _simulator->getModels()->current()->getSimulation()->getReporter()->showSimulationStatistics();
00094     } catch (...) {
00095         Trace("Error showing reports");
00096     }
00097 }
00098
00099 void GenesysConsole::cmdHelp() {
00100     ShellCommand* command;
00101     Trace("List of commands:");
00102     Trace(Util::SetW("Short", 6) + Util::SetW("Long", 12) + Util::SetW("Parameters", 15) +
00103 "Description");
00104     for (std::list<ShellCommand*>::iterator it = _commands->list()->begin(); it != _commands->list()->end(); it++) {
00105         //Trace("Unknown command. Type \"-h\" or \"help\" for help on possible commands.");
00106         command = (*it);
00107         Trace(Util::SetW(command->shortname, 6) + Util::SetW(command->longname, 12) +
00108             Util::SetW(command->parameters, 15) + command->description);
00109     }
00110 }
00111
00112 void GenesysConsole::cmdQuit() {
00113     Trace("Quiting/Exiting. Goodbye");
00114     exit(0);
00115 }
00116
00117 }
00118
00119 void GenesysConsole::cmdModelLoad() {
00120     Trace("Model load");
00121     try {
00122         std::string filename = _parameter;
00123         Model* model = new Model(this->_simulator);
00124         model->load(filename);
00125     } catch (...) {
00126         // _commands

```

```

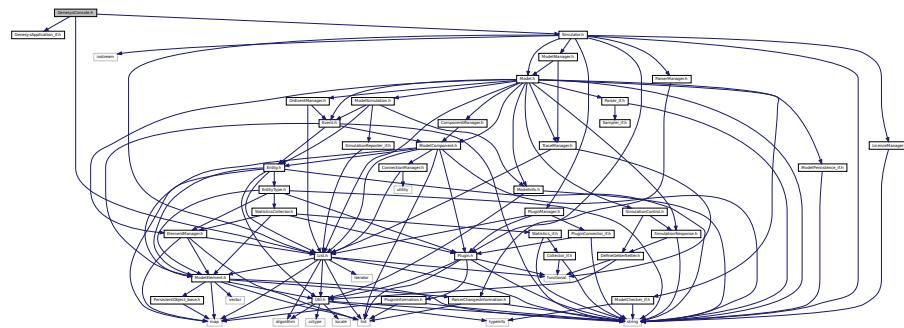
00127         Trace("    Error loading");
00128     }
00129 }
00130
00131 void GenesysConsole::cmdModelShow() {
00132     Trace("Model Show");
00133     try {
00134         _simulator->getModels()->current()->show();
00135     } catch (...) {
00136         // _commands
00137         Trace("    Error showing");
00138     }
00139 }
00140
00141 void GenesysConsole::cmdModelSave() {
00142     //this->_parameter;
00143 }
00144
00145 void GenesysConsole::cmdScript() {
00146     std::string filename = this->_parameter;
00147     //List<std::string>* arguments = new List<std::string>();
00148     std::ifstream commandfile;
00149     std::string inputText;
00150     try {
00151         commandfile.open(filename);
00152         while (getline(commandfile, inputText)) {
00153             this->tryExecuteCommand(inputText, "", "", " ");
00154         }
00155         commandfile.close();
00156     } catch (...) {
00157         Trace("    Error scripting");
00158     }
00159 }
00160
00161 void GenesysConsole::Trace(std::string message) {
00162     std::cout << message << std::endl;
00163 }
00164
00165 void GenesysConsole::run(List<std::string>* commandlineArgs) {
00166     Trace("ReGenesys Shell is running. Type your command. For help, type the command \"h\" or \"help\".");
00167     std::string inputText; //, shortPrefix, longPrefix, separator;
00168     while (true) {
00169         if (!commandlineArgs->empty()) {
00170             inputText = commandlineArgs->front();
00171             commandlineArgs->pop_front();
00172             tryExecuteCommand(inputText, "-", "--", "=");
00173         } else {
00174             std::cout << _prompt << " ";
00175             std::cin >> inputText;
00176             tryExecuteCommand(inputText, "", "", " ");
00177         }
00178     }
00179 }
00180 }
00181
00182 void GenesysConsole::tryExecuteCommand(std::string inputText, std::string shortPrefix, std::string
00183                                         longPrefix, std::string separator) {
00184     std::regex regex(R"([=]+)"); // split on space R"([\s]+)"
00185     std::sregex_token_iterator it(inputText.begin(), inputText.end(), regex, -1);
00186     std::vector<std::string> fields(it, {});
00187     std::string typedCommandStr = fields[0];
00188     if (fields.size() > 1) { // its a comnd and a parameter in the form command=parameter
00189         assert(fields.size() == 2);
00190         _parameter = fields[1];
00191     } else {
00192         _parameter = "";
00193     }
00194     ShellCommand* command;
00195     bool found;
00196     std::transform(typedCommandStr.begin(), typedCommandStr.end(), typedCommandStr.begin(),
00197                   ::tolower);
00198     if (typedCommandStr.substr(0, 1) != "#") {
00199         found = false;
00200         for (std::list<ShellCommand*>::iterator it = _commands->list()->begin(); it != _commands->list()->end(); it++) {
00201             //Trace("Unknown command. Type \"-h\" or \"help\" for help on possible commands.");
00202             command = (*it);
00203             if (typedCommandStr == shortPrefix + command->shortname || typedCommandStr == longPrefix +
00204                 command->longname) {
00205                 command->executer();
00206                 found = true;
00207                 break;
00208             }
00209         }
00210     }
00211     if (!found)
00212         Trace("Command \"" + typedCommandStr + "\" not found. Type \"h\" or \"help\" for help.");

```

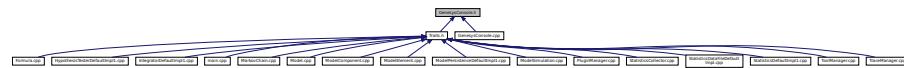
```
00209     }
00210 }
00211
00212 }
00213
00214 int GenesysConsole::main(int argc, char** argv) {
00215     //double res;
00216     //for (double x = -3.0; x <= 3.0; x += 0.05) {
00217     //    res = ProbDistrib::tStudent(x, 0, 1, 100);
00218     //    std::cout << x << ":" << res << std::endl;
00219     //}
00220     //return 0;
00221     List<std::string>* commandlineArgs = new List<std::string>();
00222     for (unsigned short i = 1; i < argc; i++) {
00223         std::string arg = argv[i];
00224         commandlineArgs->insert(arg);
00225     }
00226     //commandlineArgs->insert("-rf=temp/script.txt");
00227     //commandlineArgs->insert("-ml=models/genesysmodel.txt");
00228     this->run(commandlineArgs);
00229     return 0;
00230 }
```

9.159 GenesysConsole.h File Reference

```
#include "GenesysApplication_if.h"
#include "Simulator.h"
#include "List.h"
Include dependency graph for GenesysConsole.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class GenesysConsole

9.160 GenesysConsole.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*
```

```

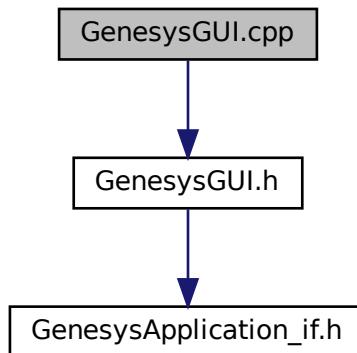
00008 * File: GenesysConsole.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:02
00012 */
00013
00014 #ifndef GENESYSCONSOLE_H
00015 #define GENESYSCONSOLE_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "Simulator.h"
00019 #include "List.h"
00020
00021 class GenesysConsole : public GenesysApplication_if {
00022 private:
00023     typedef std::function<void() > ExecuterMember;
00024
00025     template<typename Class>
00026     ExecuterMember DefineExecuterMember(Class * object, void (Class::*function)()) {
00027         return std::bind(function, object); //, std::placeholders::_1
00028     }
00029
00030     class ShellCommand {
00031     public:
00032
00033         ShellCommand(std::string shortname, std::string longname, std::string parameters, std::string
00034             description, ExecuterMember executer) {
00035             this->description = description;
00036             this->longname = longname;
00037             this->parameters = parameters;
00038             this->shortname = shortname;
00039             this->executer = executer;
00040         }
00041         std::string shortname;
00042         std::string longname;
00043         std::string parameters;
00044         std::string description;
00045         ExecuterMember executer;
00046     };
00047     public:
00048     GenesysConsole();
00049     virtual ~GenesysConsole() = default;
00050     public:
00051     virtual int main(int argc, char** argv);
00052     public: // commands
00053     void cmdScript();
00054     void cmdHelp();
00055     void cmdQuit();
00056     void cmdModelLoad();
00057     void cmdModelCheck();
00058     void cmdStart();
00059     void cmdStep();
00060     void cmdStop();
00061     void cmdShowReport();
00062     void cmdModelSave();
00063     void cmdModelShow();
00064     void cmdVersion();
00065     void cmdTraceLevel();
00066     private:
00067     void run(List<std::string>* arguments);
00068     void Trace(std::string message);
00069     void tryExecuteCommand(std::string inputText, std::string shortPrefix, std::string longPrefix,
00070         std::string separator);
00071     private:
00072     Simulator* _simulator = new Simulator();
00073     std::string _parameter;
00074     List<ShellCommand*>* _commands = new List<ShellCommand*>();
00075     std::string _prompt = "$ReGenesys> ";
00076 #endif /* GENESYSCONSOLE_H */
00077

```

9.161 GenesysGUI.cpp File Reference

```
#include "GenesysGUI.h"
```

Include dependency graph for GenesysGUI.cpp:



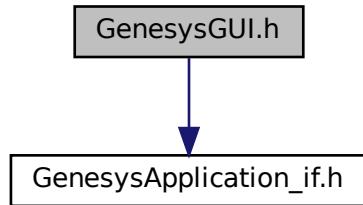
9.162 GenesysGUI.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysGUI.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:03
00012 */
00013
00014 #include "GenesysGUI.h"
00015 // #include "./qtGuiProject/QtRebornedGenesys/mainwindow.h"
00016 // #include <QApplication>
00017
00018 GenesysGUI::GenesysGUI() {
00019 }
00020
00021 int GenesysGUI::main(int argc, char** argv) {
00022     // QApplication a(argc, argv);
00023     // MainWindow w;
00024     // w.show();
00025     // return a.exec();
00026     return 0;
00027 }
```

9.163 GenesysGUI.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for GenesysGUI.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [GenesysGUI](#)

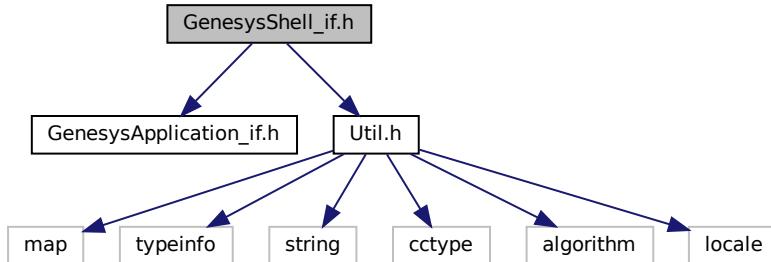
9.164 GenesysGUI.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   GenesysGUI.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 23 de Maio de 2019, 13:03
00012 */
00013
00014 #ifndef GENESYSGUI_H
00015 #define GENESYSGUI_H
00016
00017 #include "GenesysApplication_if.h"
00018
00019 // GUI PROJECT HAS BEEN MOVED TO A QT PROJECT ELSEWHERE
00020
00021 class GenesysGUI : public GenesysApplication_if {
00022 public:
00023     GenesysGUI();
00024     ~GenesysGUI() = default;
00025 public:
00026     virtual int main(int argc, char** argv);
00027 };
00028
00029 #endif /* GENESYSGUI_H */
00030
  
```

9.165 GenesysShell_if.h File Reference

```
#include "GenesysApplication_if.h"
#include "Util.h"
Include dependency graph for GenesysShell_if.h:
```



Classes

- class [GenesysShell_if](#)

9.166 GenesysShell_if.h

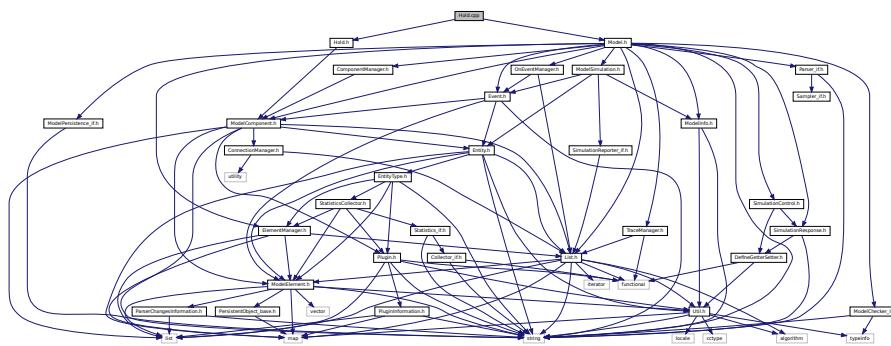
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysShell_if.h
00009  * Author: rlcancian
00010 *
00011  * Created on 24 de Maio de 2019, 11:02
00012 */
00013
00014 #ifndef GENESYSSHELL_IF_H
00015 #define GENESYSSHELL_IF_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "Util.h"
00019
00020 class GenesysShell_if : public GenesysApplication_if {
00021 public:
00022     virtual void openModel(std::string filename) = 0;
00023     virtual void saveModelAs(std::string filename) = 0;
00024     virtual void saveModel() = 0;
00025     virtual void listElements() = 0;
00026     virtual void listComponents() = 0;
00027     virtual void listHosts() = 0;
00028     virtual void listPlugins() = 0;
00029     virtual void deleteTraceFiles() = 0;
00030     virtual void traceLevel(Util::TraceLevel tracelevel) = 0;
00031     virtual void addPlugin(std::string filename) = 0;
00032     virtual void addFromFile(std::string filename) = 0;
00033     virtual void readCommandsFromFile(std::string filename) = 0;
00034     virtual void redirectTrace(std::string trace, std::string dest, std::string filename) = 0;
00035     virtual void closeModel() = 0;
00036     virtual void createModel() = 0;
00037     virtual void execLinuxCommand(std::string command) = 0;
00038     virtual void verboseMode(bool on) = 0;
00039     virtual void check() = 0;
00040     virtual void getGenesysInfo() = 0;
00041     virtual void getCommandLine() = 0;
00042     virtual void sendFile(std::string filename, std::string hostname, std::string portname) = 0;
00043     virtual void setActivationCode(std::string code) = 0;
  
```

```
00044     virtual void receiveFile(std::string filename) = 0;
00045     virtual void startSimulation() = 0;
00046     virtual void stepSimulation() = 0;
00047     virtual void stopSimulation() = 0;
00048     virtual void showInit() = 0;
00049     virtual void showHelp() = 0;
00050     virtual void showHostName() = 0;
00051     //virtual void execute() = 0;
00052     //property TraceHandler : TProcTrace write aTraceHandler;
00053 };
00054
00055 #endif /* GENESYSSHELL_IF_H */
00056
```

9.167 Hold.cpp File Reference

```
#include "Hold.h"  
#include "Model.h"  
Include dependency graph for Hold.cpp:
```



9.168 Hold.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Hold.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Hold.h"
00015 #include "Model.h"
00016
00017 Hold::Hold(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Hold>(), name) {
00018 }
00019
00020 std::string Hold::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* Hold::LoadInstance(Model* model, std::map<std::string, std::string*>* fields) {
00025     Hold* newComponent = new Hold(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029
00030     }
00031     return newComponent;
00032 }
00033
00034 void Hold::_execute(Entity* entity) {
00035     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
```

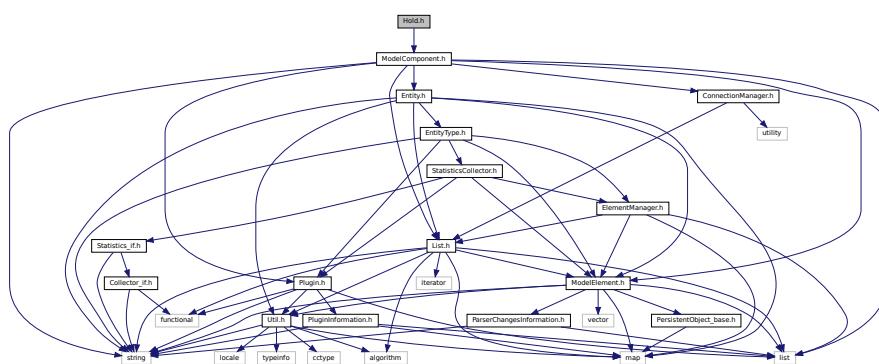
```

00036     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00037     0.0);
00038
00039 bool Hold::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
00046
00047 void Hold::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Hold::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool Hold::_check(std::string* errorMessage) {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
00061
00062 PluginInformation* Hold::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Hold>(), &Hold::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067

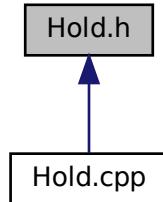
```

9.169 Hold.h File Reference

#include "ModelComponent.h"
Include dependency graph for Hold.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Hold](#)

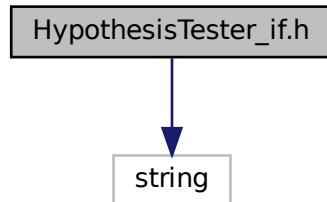
9.170 Hold.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Hold.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef HOLD_H
00015 #define HOLD_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Hold : public ModelComponent {
00020 public: // constructors
00021     Hold(Model* model, std::string name = "");
00022     virtual ~Hold() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00028 protected: // virtual
00029     virtual void _execute(Entity* entity);
00030     virtual void _initBetweenReplications();
00031     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032     virtual std::map<std::string, std::string>* _saveInstance();
00033     virtual bool _check(std::string* errorMessage);
00034 private: // methods
00035 private: // attributes 1:1
00036 private: // attributes 1:n
00037 };
00038
00039
00040 #endif /* HOLD_H */
00041
  
```

9.171 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](#)

9.172 HypothesisTester_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTester_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 19:04
00012 */
00013
00014 #ifndef HYPOTHESISTESTER_IF_H
00015 #define HYPOTHESISTESTER_IF_H
00016
00017 #include <string>
00018
00019 class HypothesisTester_if {
00020 public:
00021     enum H1Comparition {
00022         DIFFERENT = 1,
00023         LESS_THAN = 2,
00024         GREATER_THAN = 3
00025     };
00026     public:
00027         /* \todo: all "test" methods should return double p-value!! */
00028         virtual double testAverage(double confidencelevel, H1Comparition comp, std::string
00029             secondPopulationDataFilename = "") = 0;
00030         virtual double testProportion(double confidencelevel, H1Comparition comp, std::string
00031             secondPopulationDataFilename = "") = 0;
  
```

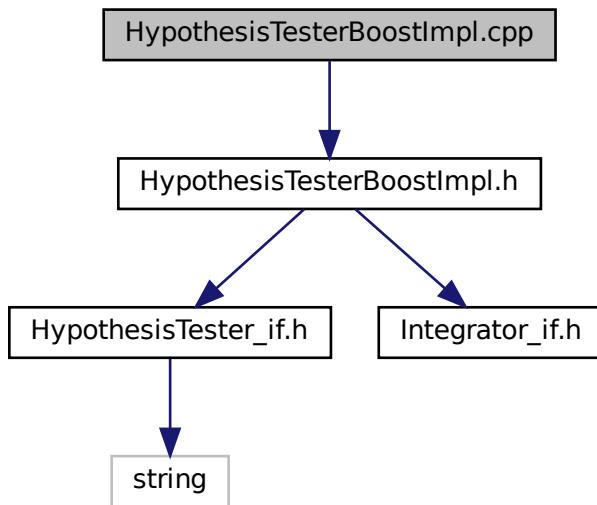
```

00034     virtual double testVariance(double confidencelevel, H1Comparition comp, std::string
00035         secondPopulationDataFilename = "") = 0;
00036     virtual void setDataFilename(std::string datafilename) = 0;
00037     virtual std::string getDataFilename() = 0;
00038 };
00039 #endif /* HYPOTHESISTESTER_IF_H */
00040

```

9.173 HypothesisTesterBoostImpl.cpp File Reference

#include "HypothesisTesterBoostImpl.h"
Include dependency graph for HypothesisTesterBoostImpl.cpp:



9.174 HypothesisTesterBoostImpl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterBoostImpl.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 13:31
00012 */
00013
00014 #include "HypothesisTesterBoostImpl.h"
00015
00016 HypothesisTesterBoostImpl::HypothesisTesterBoostImpl() {
00017 }
00018
00019 double HypothesisTesterBoostImpl::testAverage(double confidencelevel, H1Comparition comp, std::string
00020     secondPopulationDataFilename) {
00021     // \todo not implemented yet
00022 }
00023 double HypothesisTesterBoostImpl::testProportion(double confidencelevel, H1Comparition comp,
00024     std::string secondPopulationDataFilename) {
00025     // \todo not implemented yet

```

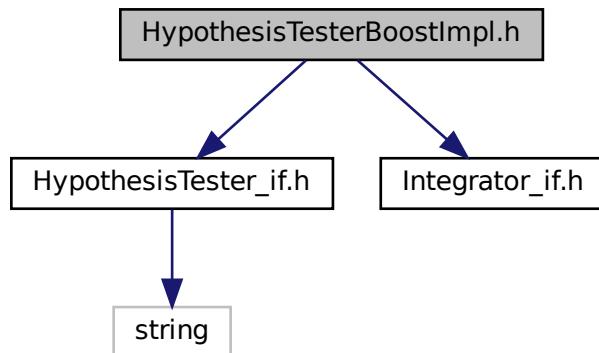
```

00025 }
00026
00027 double HypothesisTesterBoostImpl::testVariance(double confidencelevel, H1Comparition comp, std::string
    secondPopulationDataFilename) {
00028     // \todo not implemented yet
00029 }
00030 ;
00031
00032 void HypothesisTesterBoostImpl::setDataFilename(std::string datafilename) {
00033     // \todo not implemented yet
00034 }
00035
00036 std::string HypothesisTesterBoostImpl::getDataFilename() {
00037     return ""; // \todo not implemented yet
00038 }

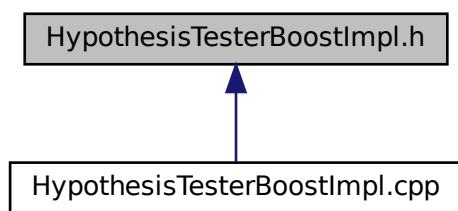
```

9.175 HypothesisTesterBoostImpl.h File Reference

```
#include "HypothesisTester_if.h"
#include "Integrator_if.h"
Include dependency graph for HypothesisTesterBoostImpl.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [HypothesisTesterBoostImpl](#)

9.176 HypothesisTesterBoostImpl.h

```

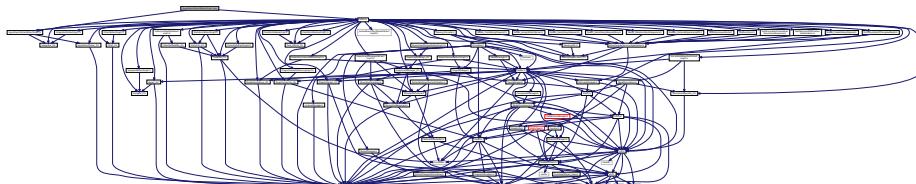
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: HypothesisTesterBoostImpl.h
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 13:31
00012 */
00013
00014 #ifndef HYPOTHESISTESTERBOOSTIMPL_H
00015 #define HYPOTHESISTESTERBOOSTIMPL_H
00016
00017 #include "HypothesisTester_if.h"
00018 #include "Integrator_if.h"
00019
00020 class HypothesisTesterBoostImpl : public HypothesisTester_if {
00021 public:
00022     HypothesisTesterBoostImpl();
00023     virtual ~HypothesisTesterBoostImpl() = default;
00024 public:
00025     virtual double testAverage(double confidencelevel, H1Comparition comp, std::string
secondPopulationDataFilename = "");
00026     virtual double testProportion(double confidencelevel, H1Comparition comp, std::string
secondPopulationDataFilename = "");
00027     virtual double testVariance(double confidencelevel, H1Comparition comp, std::string
secondPopulationDataFilename = "");
00028     virtual void setDataFilename(std::string dataFilename);
00029     virtual std::string getDataFilename();
00030 private:
00031     Integrator_if* _integrator;
00032 };
00033
00034 #endif /* HYPOTHESISTESTERBOOSTIMPL_H */
00035

```

9.177 HypothesisTesterDefaultImpl1.cpp File Reference

```
#include "HypothesisTesterDefaultImpl1.h"
#include "Traits.h"
```

Include dependency graph for HypothesisTesterDefaultImpl1.cpp:



9.178 HypothesisTesterDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:27
00012 */
00013
00014 #include "HypothesisTesterDefaultImpl1.h"
00015 #include "Traits.h"
00016
00017 HypothesisTesterDefaultImpl1::HypothesisTesterDefaultImpl1() {

```

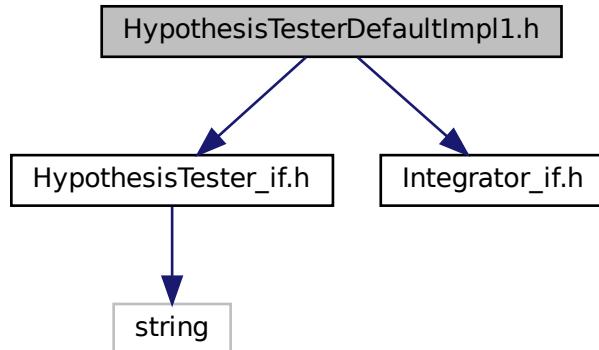
```

00018     _integrator = new Traits<HypothesisTester_if>::IntegratorImplementation();
00019 }
00020
00021 double HypothesisTesterDefaultImpl1::testAverage(double confidencelevel, H1Comparition comp,
00022         std::string secondPopulationDataFilename) {
00023     // \todo not implemented yet
00024 }
00025 double HypothesisTesterDefaultImpl1::testProportion(double confidencelevel, H1Comparition comp,
00026         std::string secondPopulationDataFilename) {
00027     // \todo not implemented yet
00028 }
00029 double HypothesisTesterDefaultImpl1::testVariance(double confidencelevel, H1Comparition comp,
00030         std::string secondPopulationDataFilename) {
00031     // \todo not implemented yet
00032 ;
00033
00034 void HypothesisTesterDefaultImpl1::setDataFilename(std::string dataFilename) {
00035     // \todo not implemented yet
00036 }
00037
00038 std::string HypothesisTesterDefaultImpl1::getDataFilename() {
00039     return ""; // \todo not implemented yet
00040 }

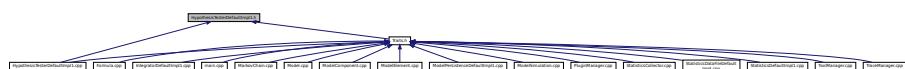
```

9.179 HypothesisTesterDefaultImpl1.h File Reference

```
#include "HypothesisTester_if.h"
#include "Integrator_if.h"
Include dependency graph for HypothesisTesterDefaultImpl1.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [HypothesisTesterDefaultImpl1](#)

9.180 HypothesisTesterDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: HypothesisTesterDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:27
00012 */
00013
00014 #ifndef HYPOTHESISTESTERDEFAULTIMPL1_H
00015 #define HYPOTHESISTESTERDEFAULTIMPL1_H
00016
00017 #include "HypothesisTester_if.h"
00018 #include "Integrator_if.h"
00019
00020 class HypothesisTesterDefaultImpl1 : public HypothesisTester_if {
00021 public:
00022     HypothesisTesterDefaultImpl1();
00023     virtual ~HypothesisTesterDefaultImpl1() = default;
00024 public:
00025     virtual double testAverage(double confidencelevel, H1Comparition comp, std::string secondPopulationDataFilename = "");
00026     virtual double testProportion(double confidencelevel, H1Comparition comp, std::string secondPopulationDataFilename = "");
00027     virtual double testVariance(double confidencelevel, H1Comparition comp, std::string secondPopulationDataFilename = "");
00028     virtual void setDataFilename(std::string dataFilename);
00029     virtual std::string getDataFilename();
00030 private:
00031     Integrator_if* _integrator;
00032 };
00033
00034 #endif /* HYPOTHESISTESTERDEFAULTIMPL1_H */
00035

```

9.181 Integrator_if.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [Integrator_if](#)

9.182 Integrator_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: Integrator_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 13:54
00012 */
00013
00014 #ifndef INTEGRATOR_IF_H
00015 #define INTEGRATOR_IF_H
00016

```

```

00021 class Integrator_if {
00022 public:
00023     virtual void setPrecision(double e) = 0;
00024     virtual double getPrecision() = 0;
00025     virtual double integrate(double min, double max, double (*f)(double, double), double p2) = 0;
00026     virtual double integrate(double min, double max, double (*f)(double, double, double), double p2,
00027         double p3) = 0;
00028     virtual double integrate(double min, double max, double (*f)(double, double, double, double),
00029         double p2, double p3, double p4) = 0;
00030     virtual double integrate(double min, double max, double (*f)(double, double, double, double,
00031         double), double p2, double p3, double p4, double p5) = 0;
00032 };
00030
00031 #endif /* INTEGRATOR_IF_H */
00032

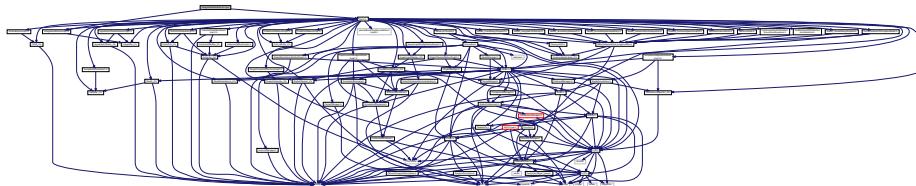
```

9.183 IntegratorDefaultImpl1.cpp File Reference

#include "IntegratorDefaultImpl1.h"

#include "Traits.h"

Include dependency graph for IntegratorDefaultImpl1.cpp:



9.184 IntegratorDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   IntegratorDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 2 de Agosto de 2018, 00:44
00012 */
00013
00014 #include "IntegratorDefaultImpl1.h"
00015 #include "Traits.h"
00016
00018
00019 IntegratorDefaultImpl1::IntegratorDefaultImpl1() {
00020     this->_precision = Traits<Integrator_if>::Precision;
00021 }
00022
00023 void IntegratorDefaultImpl1::setPrecision(double e) {
00024     _precision = e;
00025 }
00026
00027 double IntegratorDefaultImpl1::getPrecision() {
00028     return _precision;
00029 }
00030
00031 double IntegratorDefaultImpl1::integrate(double min, double max, double (*f)(double, double), double
00032     p2) {
00032     //Rosetta::GaussLegendreQuadrature<5>();
00033     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00034     double x, y, h = (max - min) / m;
00035     double I1 = 0.0;
00036     for (unsigned int j = 1; j <= m + 1; j++) {
00037         i = j - 1;
00038         if (i == 0 || i == m)
00039             c = 1;
00040         else
00041             c = 2;
00042         x = min + i*h;

```

```

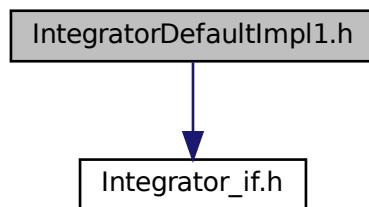
00043     y = f(x, p2);
00044     I1 += c*y;
00045 }
00046 return (h / 2)*I1;
00047 }
00048
00049 double IntegratorDefaultImpl1::integrate(double min, double max, double (*f)(double, double, double),
00050     double p2, double p3) {
00051     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00052     double x, y, h = (max - min) / m;
00053     double I1 = 0.0;
00054     for (unsigned int j = 1; j <= m + 1; j++) {
00055         i = j - 1;
00056         if (i == 0 || i == m)
00057             c = 1;
00058         else
00059             c = 2;
00060         x = min + i*h;
00061         y = f(x, p2, p3);
00062         I1 += c*y;
00063     }
00064     return (h / 2)*I1;
00065 }
00066
00067 double IntegratorDefaultImpl1::integrate(double min, double max, double (*f)(double, double, double,
00068     double), double p2, double p3, double p4) {
00069     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00070     double x, y, h = (max - min) / m;
00071     double I1 = 0.0;
00072     for (unsigned int j = 1; j <= m + 1; j++) {
00073         i = j - 1;
00074         if (i == 0 || i == m)
00075             c = 1;
00076         else
00077             c = 2;
00078         x = min + i*h;
00079         y = f(x, p2, p3, p4);
00080         I1 += c*y;
00081     }
00082     return (h / 2)*I1;
00083 }
00084
00085 double IntegratorDefaultImpl1::integrate(double min, double max, double (*f)(double, double, double,
00086     double, double), double p2, double p3, double p4, double p5) {
00087     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00088     double x, y, h = (max - min) / m;
00089     double I1 = 0.0;
00090     for (unsigned int j = 1; j <= m + 1; j++) {
00091         i = j - 1;
00092         if (i == 0 || i == m)
00093             c = 1;
00094         else
00095             c = 2;
00096         x = min + i*h;
00097         y = f(x, p2, p3, p4, p5);
00098         I1 += c*y;
00099     }
00100 /*
00101 double simpsonIntegral(double a, double b, int n, const std::function<double (double)> &f) {
00102     const double width = (b-a)/n;
00103
00104     double simpson_integral = 0;
00105     for(int step = 0; step < n; step++) {
00106         const double x1 = a + step*width;
00107         const double x2 = a + (step+1)*width;
00108
00109         simpson_integral += (x2-x1)/6.0*(f(x1) + 4.0*f(0.5*(x1+x2)) + f(x2));
00110     }
00111
00112     return simpson_integral;
00113 }
00114 */

```

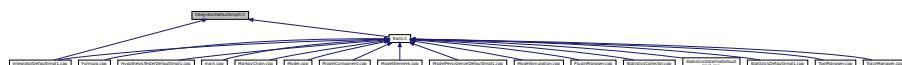
9.185 IntegratorDefaultImpl1.h File Reference

```
#include "Integrator_if.h"
```

Include dependency graph for IntegratorDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IntegratorDefaultImpl1](#)

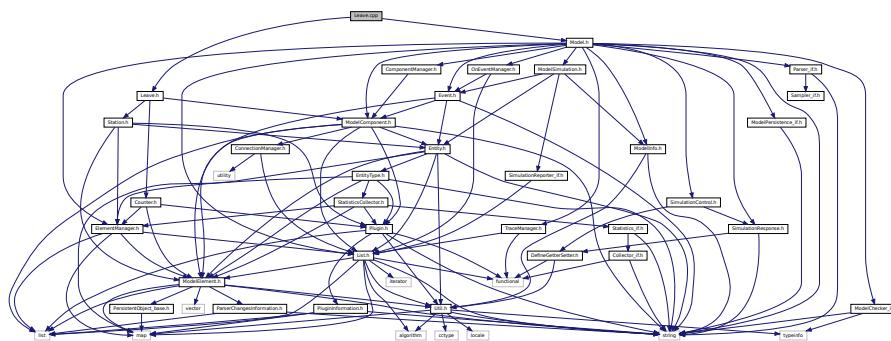
9.186 IntegratorDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   IntegratorDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 00:44
00012 */
00013
00014 #ifndef INTEGRATORDEFAULTIMPL1_H
00015 #define INTEGRATORDEFAULTIMPL1_H
00016
00017 #include "Integrator_if.h"
00018
00019 class IntegratorDefaultImpl1 : public Integrator_if {
00020 public:
00021     IntegratorDefaultImpl1();
00022     virtual ~IntegratorDefaultImpl1() = default;
00023 public:
00024     virtual void setPrecision(double e);
00025     virtual double getPrecision();
00026     virtual double integrate(double min, double max, double (*f)(double, double), double p2,
00027                               virtual double integrate(double min, double max, double (*f)(double, double, double),
00028                               double p3);
00029     virtual double integrate(double min, double max, double (*f)(double, double, double, double),
00030                               double p2, double p3, double p4);
00031     virtual double integrate(double min, double max, double (*f)(double, double, double, double,
00032                               double), double p2, double p3, double p4, double p5);
00033 private:
00034     double _precision;
00035 };
00036
00037 #endif /* INTEGRATORDEFAULTIMPL1_H */
  
```

9.187 Leave.cpp File Reference

```
#include "Leave.h"  
#include "Model.h"  
Include dependency graph for Leave.cpp:
```



9.188 Leave.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Leave.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Leave.h"
00015 #include "Model.h"
00016
00017 Leave::Leave(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Leave>(), name) {
00018 }
00019
00020 std::string Leave::show() {
00021     return ModelComponent::show() + ",station=" + this->_station->getName();
00022 }
00023
00024 ModelComponent* Leave::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     Leave* newComponent = new Leave(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030     return newComponent;
00031 }
00032 }
00033
00034 void Leave::setStation(Station* _station) {
00035     this->_station = _station;
00036 }
00037
00038 Station* Leave::getStation() const {
00039     return _station;
00040 }
00041
00042 void Leave::_execute(Entity* entity) {
00043     if (_reportStatistics)
00044         _numberIn->incCountValue();
00045     _station->leave(entity);
00046     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00047 }
00048
00049 bool Leave::_loadInstance(std::map<std::string, std::string>* fields) {
00050     bool res = ModelComponent::_loadInstance(fields);
00051     if (res) {
00052         std::string stationName = ((*fields->find("stationName"))).second;
```

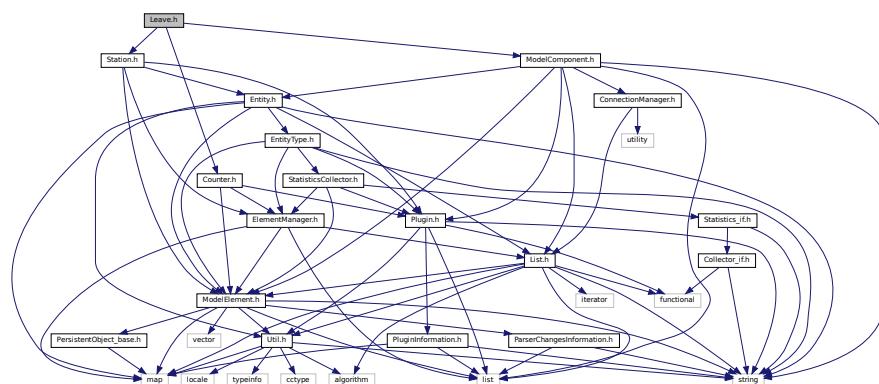
```

00053     Station* station = dynamic_cast<Station*>
00054     (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00055     this->_station = station;
00056 }
00057 }
00058
00059 void Leave::_initBetweenReplications() {
00060     _numberIn->clear();
00061 }
00062
00063 std::map<std::string, std::string>* Leave::_saveInstance() {
00064     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00065     fields->emplace("stationName", "\"" + (this->_station->getName()) + "\"");
00066     return fields;
00067 }
00068
00069 bool Leave::_check(std::string* errorMessage) {
00070     bool resultAll = true;
00071     resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",
00072         errorMessage);
00073 }
00074
00075 PluginInformation* Leave::GetPluginInformation() {
00076     PluginInformation* info = new PluginInformation(Util::TypeOf<Leave>(),
00077         &Leave::LoadInstance);
00078     info->insertDynamicLibFileDependence("station.so");
00079     return info;
00080 }
00081
00082 void Leave::_createInternalElements() {
00083     if (_reportStatistics)
00084         if (_numberIn == nullptr) {
00085             _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00086             _childrenElements->insert({ "CountNumberIn", _numberIn });
00087         } else
00088             if (_numberIn != nullptr)
00089                 _removeChildrenElements();
00090 }
00091

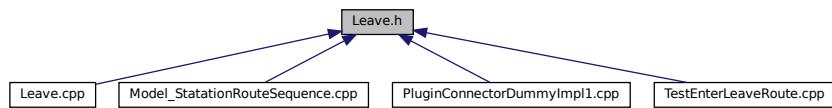
```

9.189 Leave.h File Reference

```
#include "ModelComponent.h"
#include "Station.h"
#include "Counter.h"
Include dependency graph for Leave.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Leave](#)

9.190 Leave.h

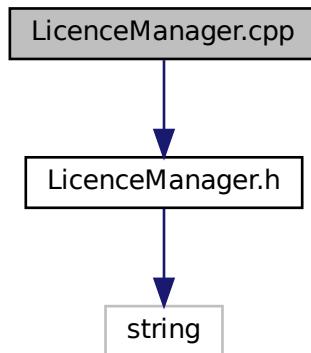
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Leave.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef LEAVE_H
00015 #define LEAVE_H
00016
00017 #include "ModelComponent.h"
00018 #include "Station.h"
00019 #include "Counter.h"
00020
00094 class Leave : public ModelComponent {
00095 public:
00096     Leave(Model* model, std::string name = "");
00097     virtual ~Leave() = default;
00098 public:
00099     virtual std::string show();
00100 public:
00101     static PluginInformation* GetPluginInformation();
00102     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00103 public:
00104     void setStation(Station* _station);
00105     Station* getStation() const;
00106 public:
00107 protected:
00108     virtual void _execute(Entity* entity);
00109     virtual void _initBetweenReplications();
00110     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00111     virtual std::map<std::string, std::string>* _saveInstance();
00112     virtual bool _check(std::string* errorMessage);
00113     virtual void _createInternalElements();
00114 private: // association
00115     Station* _station;
00116 private: // children elements
00117     Counter* _numberIn = nullptr;
00118 };
00119
00120 #endif /* LEAVE_H */
00121
  
```

9.191 LicenceManager.cpp File Reference

```
#include "LicenceManager.h"
```

Include dependency graph for LicenceManager.cpp:



9.192 LicenceManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: LicenceManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 29 de Maio de 2019, 11:45
00012 */
00013
00014 #include "LicenceManager.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 LicenceManager::LicenceManager(Simulator* simulator) {
00019     _simulator = simulator;
00020     this->setDefaultLicenceAndLimits();
00021 }
00022
00023 void LicenceManager::setDefaultLicenceAndLimits() {
00024     _licence = "LICENCE: Academic Mode. In academic mode this software has full functionality and
executing training-size simulation models. This software may be duplicated and used for educational
purposes only; any commercial application is a violation of the license agreement. Designed and
developed by prof. Dr. Ing Rafael Luiz Cancian, 2018-2021";
00025     _activationCode = "";
00026     _components = 50;
00027     _elements = 100;
00028     _entities = 200;
00029     _hosts = 1;
00030     _threads = 1;
00031 }
00032
00033 const std::string LicenceManager::showLicence() const {
00034     return _licence;
00035 }
00036
00037 const std::string LicenceManager::showLimits() const {
00038     std::string msg = "LIMITS: Based on your licence and activation code, your simulator is running
under the following limits" +
00039         std::string(": ") + std::to_string(_components) + " components" +
00040         ", " + std::to_string(_elements) + " elements" +
00041         ", " + std::to_string(_entities) + " entities" +
00042         ", " + std::to_string(_hosts) + " hosts" +
00043         ", " + std::to_string(_threads) + " threads.";
00044     return msg;
00045 }
00046
00047 const std::string LicenceManager::showActivationCode() const {
  
```

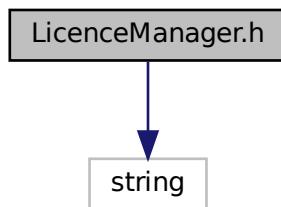
```

00048     std::string msg = "ACTIVATION CODE: Not found.";
00049     return msg;
00050 }
00051
00052 bool LicenceManager::lookforActivationCode() {
00053     // \todo: Not implemented yet
00054     return false;
00055 }
00056
00057 bool LicenceManager::insertActivationCode() {
00058     // \todo: Not implemented yet
00059     return false;
00060 }
00061
00062 void LicenceManager::removeActivationCode() {
00063     this->setDefaultLicenceAndLimits();
00064 }
00065
00066 unsigned int LicenceManager::getModelComponentsLimit() {
00067     return this->_components;
00068 }
00069
00070 unsigned int LicenceManager::getModelElementsLimit() {
00071     return this->_elements;
00072 }
00073
00074 unsigned int LicenceManager::getEntityLimit() {
00075     return this->_entities;
00076 }
00077
00078 unsigned int LicenceManager::getHostsLimit() {
00079     return this->_hosts;
00080 }
00081
00082 unsigned int LicenceManager::getThreadsLimit() {
00083     return this->_threads;
00084 }

```

9.193 LicenceManager.h File Reference

#include <string>
Include dependency graph for LicenceManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [LicenceManager](#)

9.194 LicenceManager.h

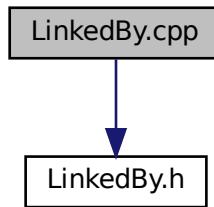
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: LicenceManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 29 de Maio de 2019, 11:45
00012 */
00013
00014 #ifndef LICENCEMANAGER_H
00015 #define LICENCEMANAGER_H
00016
00017 #include <string>
00018
00019 //namespace GenesysKernel {
00020     class Simulator;
00021
00022     class LicenceManager {
00023     public:
00024         LicenceManager(Simulator* simulator);
00025         virtual ~LicenceManager() = default;
00026         const std::string showLicence() const;
00027         const std::string showLimits() const;
00028         const std::string showActivationCode() const;
00029         bool lookforActivationCode();
00030         bool insertActivationCode();
00031         void removeActivationCode();
00032         unsigned int getModelComponentsLimit();
00033         unsigned int getModelElementsLimit();
00034         unsigned int getEntityLimit();
00035         unsigned int getHostsLimit();
00036         unsigned int getThreadsLimit();
00037     private:
00038         void setDefaultLicenceAndLimits();
00039     private:
00040         Simulator* _simulator;
00041         std::string _licence;
00042         std::string _activationCode;
00043         unsigned int _components, _elements, _entities, _hosts, _threads;
00044     };
00045 //namespace\\}
00046 #endif /* LICENCEMANAGER_H */
00047

```

9.195 LinkedBy.cpp File Reference

```
#include "LinkedBy.h"
Include dependency graph for LinkedBy.cpp:
```



9.196 LinkedBy.cpp

```
00001 /*
```

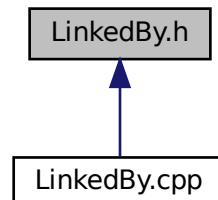
```

00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: LinkedBy.cpp
00008 * Author: rafael.luiz.cancian
00009 *
00010 *
00011 * Created on 22 de Agosto de 2018, 07:35
00012 */
00013
00014 #include "LinkedBy.h"
00015
00016 LinkedBy::LinkedBy() {
00017 }
00018
00019 LinkedBy::LinkedBy(const LinkedBy& orig) {
00020 }
00021
00022 LinkedBy::~LinkedBy() {
00023 }
00024
00025 void LinkedBy::addLink() {
00026     this->_linkedBy++;
00027 }
00028
00029 void LinkedBy::removeLink() {
00030     this->_linkedBy--;
00031 }
00032
00033 bool LinkedBy::isLinked() {
00034     return this->_linkedBy > 0;
00035 }

```

9.197 LinkedBy.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [LinkedBy](#)

9.198 LinkedBy.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: LinkedBy.h

```

```

00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Agosto de 2018, 07:35
00012 */
00013
00014 #ifndef LINKEDBYCOMPONENT_H
00015 #define LINKEDBYCOMPONENT_H
00016
00017 class LinkedBy {
00018 public:
00019     LinkedBy();
00020     LinkedBy(const LinkedBy& orig);
00021     virtual ~LinkedBy();
00022 public:
00023     void addLink();
00024     void removeLink();
00025     bool isLinked();
00026 private:
00027     unsigned int _linkedBy = 0;
00028 };
00029
00030 #endif /* LINKEDBYCOMPONENT_H */
00031

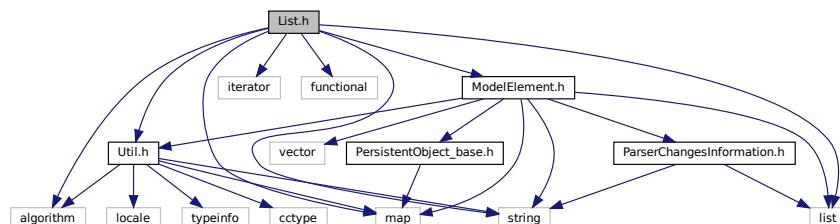
```

9.199 List.h File Reference

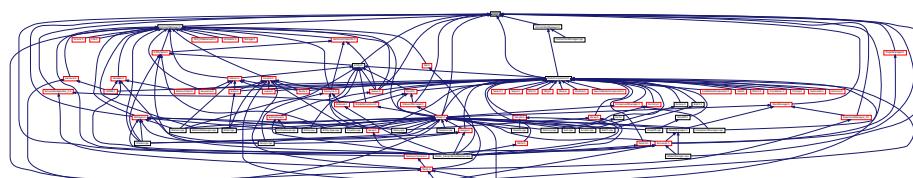
```

#include <string>
#include <list>
#include <map>
#include <iterator>
#include <functional>
#include <algorithm>
#include "Util.h"
#include "ModelElement.h"
Include dependency graph for List.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [List< T >](#)

9.200 List.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:55
00012 */
00013
00014 #ifndef LISTMANAGER_H
00015 #define LISTMANAGER_H
00016
00017 #include <string>
00018 #include <list>
00019 #include <map>
00020 #include <iterator>
00021 #include <functional>
00022 #include <algorithm>
00023 #include "Util.h"
00024 #include "ModelElement.h"
00025
00026 //class Simulator;
00027
00031 template <typename T>
00032 class List {
00033 public:
00034     using CompFunct = std::function<bool(const T, const T) >;
00035 public:
00036     List();
00037     virtual ~List() = default;
00038 public: // direct access to list
00039     unsigned int size();
00040     bool empty();
00041     void clear();
00042     void pop_front();
00043     template<class Compare>
00044     void sort(Compare comp);
00045     std::list<T>* list() const;
00046 public: // new methods
00047     T create();
00048     template<typename U>
00049     T create(U arg);
00050     std::string show();
00051     typename std::list<T>::iterator find(T element);
00052     //int rankOf(T element); // returns the position (1st position=0) of the element if found, or
00053     // negative value if not found
00054 public: // improved (easier) methods
00055     void insert(T element);
00056     void remove(T element);
00057     void setAtRank(unsigned int rank, T element);
00058     T getAtRank(unsigned int rank);
00059     T next();
00060     T front();
00061     T last();
00062     T previous();
00063     T current(); // get current element on the list (the last used)
00064     void setSortFunc(CompFunct _sortFunc);
00065 private:
00066     //std::map<Util::identitifcation, T*>* _map;
00067     std::list<T>* _list;
00068     CompFunct _sortFunc{[]} (const T, const T) {
00069         return false;
00070     };
00071     typename std::list<T>::iterator _it;
00072 };
00073 template <typename T>
00074 List<T>::List() {
00075     //_map = new std::map<Util::identitifcation, T>();
00076     _list = new std::list<T>();
00077     _it = _list->begin();
00078 }
00079
00080 template <typename T>
00081 std::list<T>* List<T>::list() const {
00082     return _list;
00083 }
00084
00085 template <typename T>
00086 unsigned int List<T>::size() {
00087     return _list->size();

```

```
00088 }
00089 //template <typename T>
00090 //List<T>::List(const List& orig) {
00091 //}
00093
00094 //template <typename T>
00095 //List<T>::~List() {
00096 //}
00097
00098 template <typename T>
00099 std::string List<T>::show() {
00100     int i = 0;
00101     std::string text = "{";
00102     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++, i++) {
00103         text += "[" + std::to_string(i) + "]=" + (*it)->show() + ",";
00104     }
00105     text += ")";
00106     return text;
00107 }
00108
00109 template <typename T>
00110 void List<T>::insert(T element) {
00111     _list->insert(std::upper_bound(_list->begin(), _list->end(), element, _sortFunc), element);
00112 }
00113
00114 template <typename T>
00115 bool List<T>::empty() {
00116     return _list->empty();
00117 }
00118
00119 template <typename T>
00120 void List<T>::pop_front() {
00121     typename std::list<T>::iterator itTemp = _list->begin();
00122     _list->pop_front();
00123     if (_it == itTemp) { /* \todo: :: check this */
00124         _it = _list->begin(); // if it points to the removed element, then changes to begin
00125     }
00126 }
00127
00128 template <typename T>
00129 void List<T>::remove(T element) {
00130     _list->remove(element);
00131     if ((*_it) == element) { /* \todo: :: check this */
00132         _it = _list->begin(); // if it points to the removed element, then changes to begin
00133     }
00134 }
00135
00136 template <typename T>
00137 T List<T>::create() {
00138     return new T();
00139 }
00140
00141 template <typename T>
00142 void List<T>::clear() {
00143     _list->clear();
00144 }
00145
00146 template <typename T>
00147 T List<T>::getAtRank(unsigned int rank) {
00148     unsigned int thisRank = 0;
00149     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00150         if (rank == thisRank) {
00151             return (*it);
00152         } else {
00153             thisRank++;
00154         }
00155     }
00156     return 0; /* \todo: Invalid return depends on T. If T is pointer, nullptr works fine. If T is
00157     double, it does not. I just let (*it), buut it is not nice*/
00158 }
00159
00160 template <typename T>
00161 void List<T>::setAtRank(unsigned int rank, T element) {
00162     unsigned int thisRank = 0;
00163     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00164         if (rank == thisRank) {
00165             *it = element;
00166             return;
00167         } else {
00168             thisRank++;
00169         }
00170     }
00171
00172 template <typename T>
00173 T List<T>::next() {
```

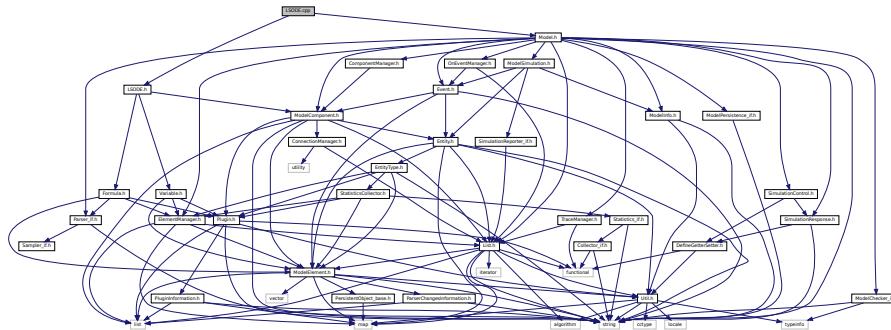
```

00174     _it++;
00175     if (_it != _list->end())
00176         return (*_it);
00177     else
00178         return nullptr;
00179
00180 }
00181
00182 template <typename T>
00183 typename std::list<T>::iterator List<T>::find(T element) {
00184     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00185         if ((*it) == element) {
00186             return it;
00187         }
00188     }
00189     return _list->end(); /* \todo:+: check nullptr or invalid iterator when not found */
00190     //return nullptr;
00191 }
00192
00193 /*
00194 template <typename T>
00195 int List<T>::rankOf(T element) {
00196     int rank = 0;
00197     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00198         if ((*it) == element) {
00199             return rank;
00200         } else
00201             rank++;
00202     }
00203     return -1; // not found -> negative rank
00204 }
00205 */
00206
00207 template <typename T>
00208 T List<T>::front() {
00209     _it = _list->begin();
00210     //if (_it != _list->end())
00211     return (*_it);
00212     //else
00213     //return dynamic_cast<T>(nullptr);
00214 }
00215
00216 template <typename T>
00217 T List<T>::last() {
00218     _it = _list->end();
00219     _it--;
00220     //if (_it != _list->end()) // \todo: CHECK!!!
00221     return (*_it);
00222     //else return nullptr;
00223 }
00224
00225 template <typename T>
00226 T List<T>::previous() {
00227     _it--; // \todo: CHECK!!!
00228     return (*_it);
00229 }
00230
00231 template <typename T>
00232 T List<T>::current() {
00233     /* \todo: To implement (i thing it's just to check). Must actualize _it on other methods when
00234     other elements are accessed */
00235     return (*_it);
00236 }
00237
00238 void List<T>::setSortFunc(CompFunct _sortFunc) {
00239     this->_sortFunc = _sortFunc;
00240 }
00241
00242 template <typename T>
00243 template<typename U>
00244 T List<T>::create(U arg) {
00245     return T(arg);
00246 }
00247
00248 template <typename T>
00249 template<class Compare>
00250 void List<T>::sort(Compare comp) {
00251     _list->sort(comp);
00252 }
00253
00254 #endif /* LISTMANAGER_H */
00255

```

9.201 LSODE.cpp File Reference

```
#include "LSODE.h"
#include "Model.h"
Include dependency graph for LSODE.cpp:
```



9.202 LSODE.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: LSODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 22:28
00012 */
00013
00014 #include "LSODE.h"
00015 #include "Model.h"
00016
00017 LSODE::LSODE(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<LSODE>(), name) {
00018 }
00019
00020 std::string LSODE::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* LSODE::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     LSODE* newComponent = new LSODE(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030     return newComponent;
00031 }
00032 }
00033
00034 void LSODE::setDiffEquations(Formula* formula) {
00035     _diffEquations = formula;
00036 }
00037
00038 Formula* LSODE::getDiffEquations() const {
00039     return _diffEquations;
00040 }
00041
00042 void LSODE:: setTimeVariable(Variable* timeVariable) {
00043     _timeVariable = timeVariable;
00044 }
00045
00046 Variable* LSODE::getTimeVariable() const {
00047     return _timeVariable;
00048 }
00049
00050 void LSODE::setStep(double step) {
00051     _step = step;
00052 }
00053
```

```

00054 double LSODE::getStep() const {
00055     return _step;
00056 }
00057
00058 void LSODE::setVariables(Variable* variables) {
00059     _variables = variables;
00060 }
00061
00062 Variable* LSODE::getVariables() const {
00063     return _variables;
00064 }
00065
00066 bool LSODE::_doStep() {
00067     double initTime, time, tnow, eqResult, halfStep;
00068     //std::list<std::string>* eqs = _diffEquations->formulaExpressions()->list();
00069     unsigned int i, numEqs = _diffEquations->size();
00070     double k1[numEqs], k2[numEqs], k3[numEqs], k4[numEqs], valVar[numEqs];
00071     time = _timeVariable->value();
00072     initTime = time;
00073     tnow = _parentModel->simulation()->simulatedTime();
00074     bool res = time + _step <= tnow + 1e-15; // \todo: numerical error treatment by just adding 1e-15
00075     if (res) {
00076         halfStep = _step * 0.5;
00077         for (i = 0; i < numEqs; i++) { // (std::list<std::string>::iterator it = eqs->begin(); it != eqs->end(); it++)
00078             std::string expression = _diffEquations->expression(std::to_string(i));
00079             valVar[i] = _variables->value(std::to_string(i));
00080             eqResult = _parentModel->parseExpression(expression);
00081             k1[i] = eqResult;
00082         }
00083         time += halfStep;
00084         _timeVariable->setValue(time);
00085         for (i = 0; i < numEqs; i++) {
00086             _variables->setValue(std::to_string(i), valVar[i] + k1[i] * halfStep);
00087         }
00088         for (i = 0; i < numEqs; i++) {
00089             eqResult = _parentModel->parseExpression(_diffEquations->expression(std::to_string(i)));
00090             k2[i] = eqResult;
00091         }
00092         for (i = 0; i < numEqs; i++) {
00093             _variables->setValue(std::to_string(i), valVar[i] + k2[i] * halfStep);
00094         }
00095         for (i = 0; i < numEqs; i++) {
00096             eqResult = _parentModel->parseExpression(_diffEquations->expression(std::to_string(i)));
00097             k3[i] = eqResult;
00098         }
00099         for (i = 0; i < numEqs; i++) {
00100             _variables->setValue(std::to_string(i), valVar[i] + k3[i] * halfStep);
00101         }
00102         for (i = 0; i < numEqs; i++) {
00103             eqResult = _parentModel->parseExpression(_diffEquations->expression(std::to_string(i)));
00104             k4[i] = eqResult;
00105         }
00106         for (i = 0; i < numEqs; i++) {
00107             eqResult = _variables->value(std::to_string(i)) + (_step / 6) * (k1[i] + 2 * (k2[i] +
00108                 k3[i]) + k4[i]);
00109             _variables->setValue(std::to_string(i), eqResult);
00110         }
00111         time = initTime + _step;
00112         _timeVariable->setValue(time);
00113     }
00114     return res;
00115 }
00116 void LSODE::_execute(Entity* entity) {
00117     //_parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00118     //for (std::list<std::string>::iterator it =
00119     //    _diffEquations->getFormulaExpressions()->list()->begin(); it != _diffEquations->getFormulaExpressions()->list()->end(); it++) {
00120     //    double value = _parentModel->parseExpression((*it));
00121     //    _parentModel->tracer()->trace("Expression \" " + (*it) + "\" evaluates to " +
00122     //        std::to_string(value));
00123     //}
00124     //while (_doStep()) { // execute solve ODE step by step until reach TNOW
00125     //    std::string message = "time=" + std::to_string(_timeVariable->value());
00126     //    for (unsigned int i = 0; i < _variables->dimensionSizes()->front(); i++) {
00127     //        message += ",y[" + std::to_string(i) + "]=" +
00128     //            std::to_string(_variables->value(std::to_string(i)));
00129     //    }
00130     //    _parentModel->tracer()->trace(message);
00131     //}
00132     _parentModel->sendEntityToComponent(entity, nextComponents()->frontConnection(), 0.0);
00133 }
00134 bool LSODE::_loadInstance(std::map<std::string, std::string>* fields) {
00135     bool res = ModelComponent::_loadInstance(fields);
00136     if (res) {

```

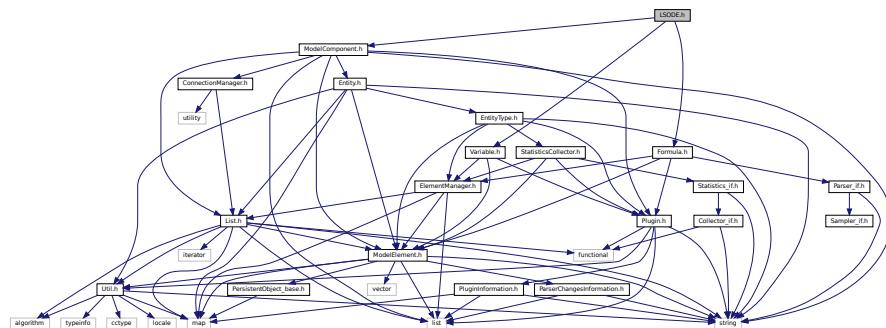
```

00135     //...
00136 }
00137     return res;
00138 }
00139
00140 void LSODE::_initBetweenReplications() {
00141 }
00142
00143 std::map<std::string, std::string>* LSODE::_saveInstance() {
00144     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00145     //...
00146     return fields;
00147 }
00148
00149 bool LSODE::_check(std::string* errorMessage) {
00150     bool resultAll = true;
00151     //...
00152     return resultAll;
00153 }
00154
00155 PluginInformation* LSODE::GetPluginInformation() {
00156     PluginInformation* info = new PluginInformation(Util::TypeOf<LSODE>(), &LSODE::LoadInstance);
00157     // ...
00158     return info;
00159 }

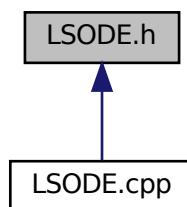
```

9.203 LSODE.h File Reference

```
#include "ModelComponent.h"
#include "Formula.h"
#include "Variable.h"
Include dependency graph for LSODE.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [LSODE](#)

9.204 LSODE.h

```

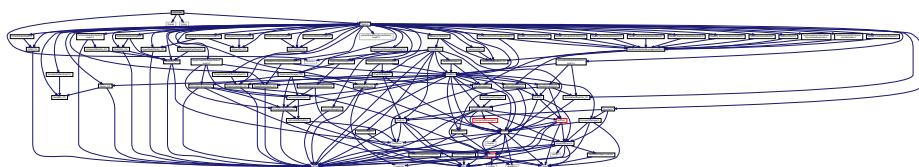
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    LSODE.h
00009  * Author:  rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 22:28
00012 */
00013
00014 #ifndef LSODE_H
00015 #define LSODE_H
00016
00017 #include "ModelComponent.h"
00018 #include "Formula.h"
00019 #include "Variable.h"
00020
00021 class LSODE : public ModelComponent {
00022 public: // constructors
00023     LSODE(Model* model, std::string name = "");
00024     virtual ~LSODE() = default;
00025 public: // virtual
00026     virtual std::string show();
00027 public: // static
00028     static PluginInformation* GetPluginInformation();
00029     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00030 public: // g&s
00031     void setDiffEquations(Formula* formula);
00032     Formula* getDiffEquations() const;
00033     void setTimeVariable(Variable* _timeVariable);
00034     Variable* getTimeVariable() const;
00035     void setStep(double _step);
00036     double getStep() const;
00037     void setVariables(Variable* _variables);
00038     Variable* getVariables() const;
00039 protected: // virtual
00040     virtual void _execute(Entity* entity);
00041     virtual void _initBetweenReplications();
00042     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00043     virtual std::map<std::string, std::string>* _saveInstance();
00044     virtual bool _check(std::string* errorMessage);
00045     //virtual void _createInternalElements();
00046 private: // methods
00047     bool _doStep();
00048 private: // attributes 1:1
00049     Formula* _diffEquations;
00050     Variable* _variables;
00051     Variable* _timeVariable;
00052     double _step;
00053 private: // attributes 1:n
00054 };
00055
00056 #endif /* LSODE_H */
00057

```

9.205 main.cpp File Reference

```
#include <iostream>
#include <thread>
#include <limits>
```

```
#include "Traits.h"  
Include dependency graph for main.cpp:
```



Functions

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

9.205.1 Function Documentation

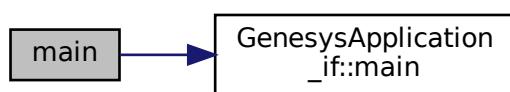
```
9.205.1.1 main() int main (  
    int argc,  
    char ** argv )
```

Definition at line 22 of file [main.cpp](#).

```
00022     {  
00023         // do not change it. Set you own application in Traits file =>  
00024         Traits<GenesysApplication_if>::Application  
00025         GenesysApplication_if *app = new Traits<GenesysApplication_if>::Application();  
00026         int res = app->main(argc, argv);  
00027         // that's all folks!  
00028         std::cout << "Press ENTER to quit...";  
00029         std::ignore(std::numeric_limits<std::streamsize> ::max(), '\n');  
00030         return res;  
00030 }
```

References [GenesysApplication_if::main\(\)](#).

Here is the call graph for this function:



9.206 main.cpp

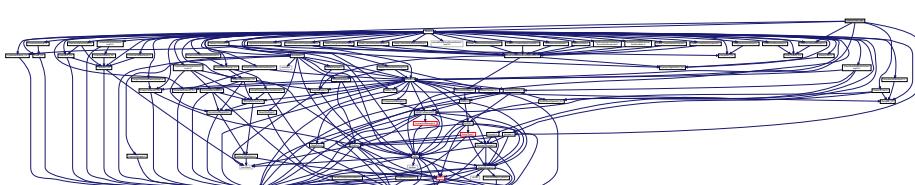
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: main.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:47
00012 */
00013
00014 #include <iostream>
00015 #include <thread>
00016 #include <limits>
00017 #include "Traits.h"
00018
00019 /*
00020 * This is the MAIN application of GenESyS. It just calls the Application specified on the
00021 * configuration file.
00022 */
00023 int main(int argc, char** argv) {
00024     // do not change it. Set your own application in Traits file =>
00025     Traits<GenesysApplication_if>::Application
00026     GenesysApplication_if *app = new Traits<GenesysApplication_if>::Application();
00027     int res = app->main(argc, argv);
00028     // that's all folks!
00029     std::cout << "Press ENTER to quit..." ;
00030     std::cin.ignore(std::numeric_limits<std::streamsize> ::max(), '\n');
00031     return res;
00032 }
```

9.207 MarkovChain.cpp File Reference

```
#include "MarkovChain.h"
#include "Model.h"
#include "Variable.h"
#include "ProbDistribDefaultImpl.h"
#include "Simulator.h"
#include "Traits.h"
```

Include dependency graph for MarkovChain.cpp:



9.208 MarkovChain.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: MarkovChain.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Outubro de 2019, 18:26
00012 */
00013
00014 #include "MarkovChain.h"
00015
00016 #include "Model.h"
```

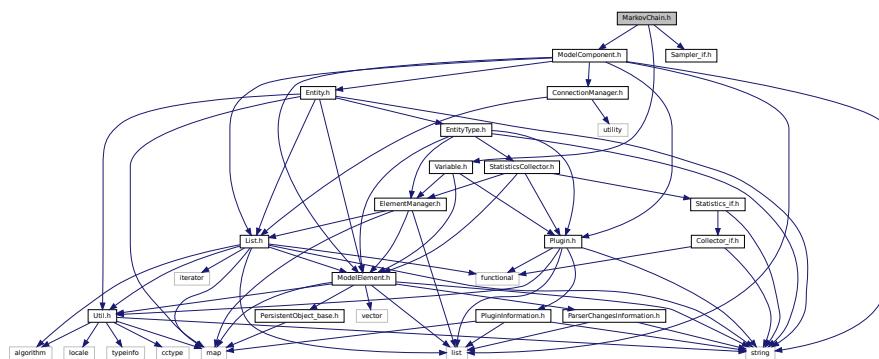
```
00017 #include "Variable.h"
00018 #include "ProbabilistDefaultImpl.h"
00019 #include "Simulator.h"
00020 #include "Traits.h"
00021
00022 MarkovChain::MarkovChain(Model* model, std::string name) : ModelComponent(model,
00023     Util::TypeOf<MarkovChain>(), name) {
00024     _sampler = new Traits<Sampler_if>::Implementation();
00025 }
00026 std::string MarkovChain::show() {
00027     return ModelComponent::show() + "";
00028 }
00029
00030 ModelComponent* MarkovChain::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00031     MarkovChain* newComponent = new MarkovChain(model);
00032     try {
00033         newComponent->_loadInstance(fields);
00034     } catch (const std::exception& e) {
00035     }
00036     return newComponent;
00037 }
00038 }
00039
00040 void MarkovChain::setTransitionProbabilityMatrix(Variable* _transitionMatrix) {
00041     this->_transitionProbMatrix = _transitionMatrix;
00042 }
00043
00044 Variable* MarkovChain::getTransitionMatrix() const {
00045     return _transitionProbMatrix;
00046 }
00047
00048 Variable* MarkovChain::getCurrentState() const {
00049     return _currentState;
00050 }
00051
00052 void MarkovChain::setCurrentState(Variable* _currentState) {
00053     this->_currentState = _currentState;
00054 }
00055
00056 void MarkovChain::setInitialized(Variable* _initialDistribution) {
00057     this->_initialDistribution = _initialDistribution;
00058 }
00059
00060 Variable* MarkovChain::getInitialState() const {
00061     return _initialDistribution;
00062 }
00063
00064 void MarkovChain::setInitialized(bool _initialized) {
00065     this->_initialized = _initialized;
00066 }
00067
00068 bool MarkovChain::isInitialized() const {
00069     return _initialized;
00070 }
00071
00072 void MarkovChain::_execute(Entity* entity) {
00073     //parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00074     unsigned int size;
00075     double rnd, sum, value;
00076     if (!_initialized) {
00077         // define the initial state based on initial probabilities
00078         size = _initialDistribution->dimensionSizes()->front();
00079         rnd = _sampler->random(); //parentSimulator()->tools()->sampler()->random();
00080         double sum = 0.0;
00081         for (unsigned int i = 0; i < size; i++) {
00082             value = _initialDistribution->value(std::to_string(i));
00083             sum += value;
00084             if (sum > rnd) {
00085                 _currentState->setValue(i); // _currentState = i;
00086                 break;
00087             }
00088         }
00089         _parentModel->tracer()->trace("Initial current state=" +
00090             std::to_string(_currentState->value()));
00091         _initialized = true;
00092     } else {
00093         size = _transitionProbMatrix->dimensionSizes()->front();
00094         rnd = _sampler->random(); //parentSimulator()->tools()->sampler()->random();
00095         sum = 0.0;
00096         for (unsigned int i = 0; i < size; i++) {
00097             std::string index = std::to_string(static_cast<unsigned int> (_currentState->value()) +
00098                 "," + std::to_string(i));
00099             value = _transitionProbMatrix->value(index);
00100             sum += value;
00101             if (sum > rnd) {
00102                 _currentState->setValue(i);
00103             }
00104         }
00105     }
00106 }
```

```

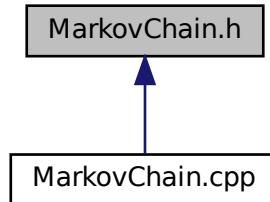
00101         break;
00102     }
00103 }
00104     _parentModel->tracer()->trace("Current state=" + std::to_string(_currentState->value()));
00105 }
00106     _parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00107 }
00108
00109 bool MarkovChain::_loadInstance(std::map<std::string, std::string>* fields) {
00110     bool res = ModelComponent::_loadInstance(fields);
00111     if (res) {
00112         //...
00113     }
00114     return res;
00115 }
00116
00117 void MarkovChain::_initBetweenReplications() {
00118     this->_initialized = false;
00119 }
00120
00121 std::map<std::string, std::string>* MarkovChain::_saveInstance() {
00122     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00123     //...
00124     return fields;
00125 }
00126
00127 bool MarkovChain::_check(std::string* errorMessage) {
00128     bool resultAll = true;
00129     //...
00130     return resultAll;
00131 }
00132
00133 PluginInformation* MarkovChain::GetPluginInformation() {
00134     PluginInformation* info = new PluginInformation(Util::TypeOf<MarkovChain>(),
00135     &MarkovChain::LoadInstance);
00136     // ...
00137     return info;
00138 }
```

9.209 MarkovChain.h File Reference

```
#include "ModelComponent.h"
#include "Variable.h"
#include "Sampler_if.h"
Include dependency graph for MarkovChain.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [MarkovChain](#)

9.210 MarkovChain.h

```

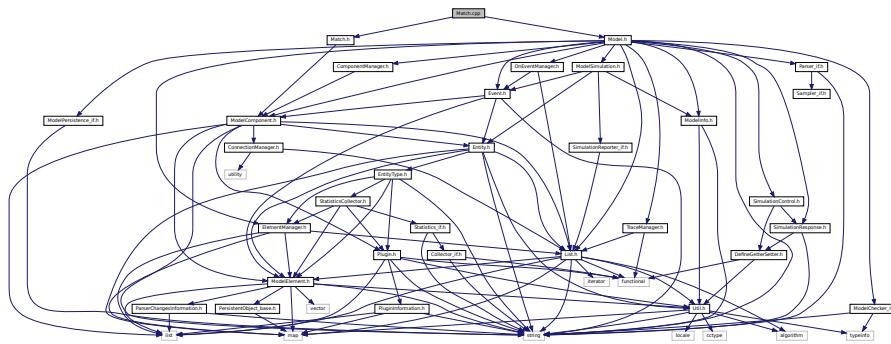
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /**
00008  * File:   MarkovChain.h
00009  * Author: rlcancian
00010 *
00011 * Created on 24 de Outubro de 2019, 18:26
00012 */
00013
00014 #ifndef MARKOVCHAIN_H
00015 #define MARKOVCHAIN_H
00016
00017 #include "ModelComponent.h"
00018 #include "Variable.h"
00019 #include "Sampler_if.h"
00020
00021 class MarkovChain : public ModelComponent {
00022 public: // constructors
00023     MarkovChain(Model* model, std::string name = "");
00024     virtual ~MarkovChain() = default;
00025 public: // virtual
00026     virtual std::string show();
00027 public: // static
00028     static PluginInformation* GetPluginInformation();
00029     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00030 public: // get and set
00031     void setTransitionProbabilityMatrix(Variable* _transitionMatrix);
00032     Variable* getTransitionMatrix() const;
00033     Variable* getCurrentState() const;
00034     void setInitialDistribution(Variable* _initialDistribution);
00035     Variable* getInitialState() const;
00036     void setInitialized(bool _initialized);
00037     bool isInitialized() const;
00038     void setCurrentState(Variable* _currentState);
00039 protected: // virtual
00040     virtual void _execute(Entity* entity);
00041     virtual void _initBetweenReplications();
00042     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00043     virtual std::map<std::string, std::string>* _saveInstance();
00044     virtual bool _check(std::string* errorMessage);
00045 private: // methods
00046 private: // attributes 1:1
00047     Variable* _transitionProbMatrix;
00048     Variable* _initialDistribution;
00049     Variable* _currentState;
00050     bool _initialized = false;

```

```
00051 private: // attributes 1:n  
00052     Sampler_if* _sampler;  
00053 };  
00054  
00055 #endif /* MARKOVCHAIN_H */  
00056
```

9.211 Match.cpp File Reference

```
#include "Match.h"  
#include "Model.h"  
Include dependency graph for Match.cpp:
```



9.212 Match.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Match.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Match.h"
00015
00016 #include "Model.h"
00017
00018 Match::Match(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Match>(), name) {
00019 }
00020
00021 std::string Match::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Match::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Match* newComponent = new Match(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
00034
00035 void Match::_execute(Entity* entity) {
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
0.0);
00038 }
00039
00040 bool Match::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
```

```

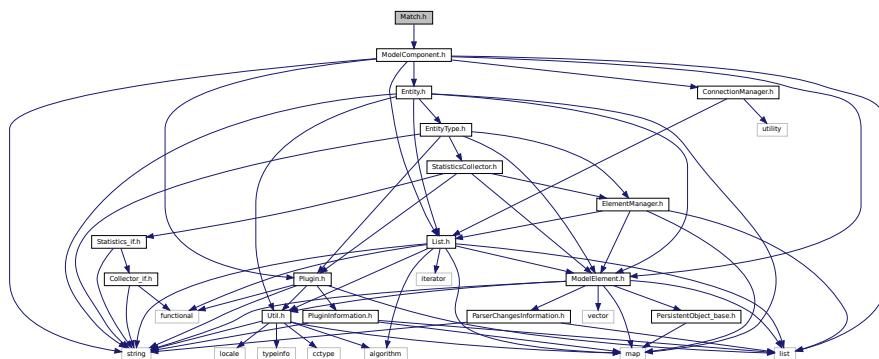
00043     //...
00044 }
00045     return res;
00046 }
00047
00048 void Match::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Match::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Match::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Match::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Match>(), &Match::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068

```

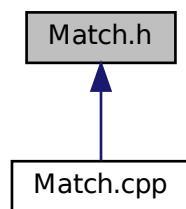
9.213 Match.h File Reference

#include "ModelComponent.h"

Include dependency graph for Match.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Match

9.214 Match.h

```

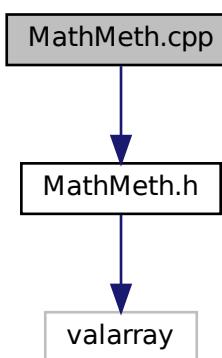
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Match.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef MATCH_H
00015 #define MATCH_H
00016
00017 #include "ModelComponent.h"
00018
00047 class Match : public ModelComponent {
00048 public: // constructors
00049     Match(Model* model, std::string name = "");
00050     virtual ~Match() = default;
00051 public: // virtual
00052     virtual std::string show();
00053 public: // static
00054     static PluginInformation* GetPluginInformation();
00055     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00056 protected: // virtual
00057     virtual void _execute(Entity* entity);
00058     virtual void _initBetweenReplications();
00059     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00060     virtual std::map<std::string, std::string>* _saveInstance();
00061     virtual bool _check(std::string* errorMessage);
00062 private: // methods
00063 private: // attributes 1:1
00064 private: // attributes 1:n
00065 };
00066
00067
00068 #endif /* MATCH_H */
00069

```

9.215 MathMeth.cpp File Reference

#include "MathMeth.h"

Include dependency graph for MathMeth.cpp:

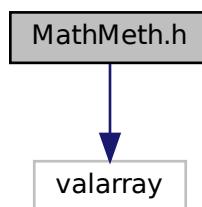


9.216 MathMeth.cpp

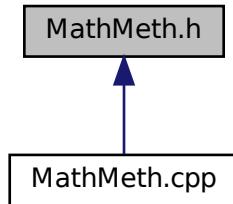
```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: MathMeth.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 17 de Outubro de 2019, 17:46
00012 */
00013
00014 #include "MathMeth.h"
00015
00016 MathMeth::MathMeth() {
00017 }
00018
00019 /*
00020 void MathMeth::RK4step(int ndep, double dx, double &x, valarray<double> &Y, valarray<double>
00021 (*F)(double, valarray<double>)) {
00022     valarray<double> dY1(ndep), dY2(ndep), dY3(ndep), dY4(ndep);
00023
00024     dY1 = F(x, Y) * dx;
00025     dY2 = F(x + 0.5 * dx, Y + 0.5 * dY1) * dx;
00026     dY3 = F(x + 0.5 * dx, Y + 0.5 * dY2) * dx;
00027     dY4 = F(x + dx, Y + dY3) * dx;
00028
00029     Y += (dY1 + 2.0 * dY2 + 2.0 * dY3 + dY4) / 6.0;
00030 }
00031 */
```

9.217 MathMeth.h File Reference

```
#include <valarray>
Include dependency graph for MathMeth.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [MathMeth](#)

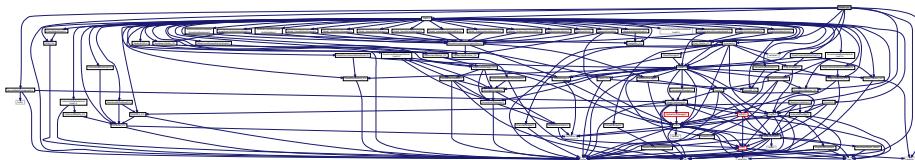
9.218 MathMeth.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    MathMeth.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 17 de Outubro de 2019, 17:46
00012 */
00013
00014 #ifndef MATHMETH_H
00015 #define MATHMETH_H
00016
00017 #include <valarray>
00018
00019 class MathMeth {
00020 public:
00021     MathMeth();
00022 public:
00023     //void RK4step(int ndep, double dx, double &x, valarray<double> &Y, valarray<double> (*F)(double,
00024     valarray<double>));
00024 private:
00025
00026 };
00027
00028 #endif /* MATHMETH_H */
```

9.219 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)

9.219.1 Function Documentation

9.219.1.1 EventCompare() bool EventCompare (

```
    const Event * a,
    const Event * b )
```

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

```
00028
00029     return a->getTime() < b->getTime();
00030 }
```

References [Event::getTime\(\)](#).

Here is the call graph for this function:



9.220 Model.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SimulationModel.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 15:01
00012 */
00013
00014 #include <typeinfo>
00015 #include <iostream>
00016 #include <algorithm>
```

```

00017 #include <string>
00018
00019 #include "Model.h"
00020 #include "SourceModelComponent.h"
00021 #include "Simulator.h"
00022 #include "StatisticsCollector.h"
00023 #include "Traits.h"
00024 // #include "Access.h"
00025
00026 //using namespace GenesysKernel;
00027
00028 bool EventCompare(const Event* a, const Event * b) {
00029     return a->getTime() < b->getTime();
00030 }
00031
00032 Model::Model(Simulator* simulator) {
00033     _parentSimulator = simulator; // a simulator is the "parent" of a model
00034     // 1:1 associations (no Traits)
00035     _traceManager = simulator->getTracer(); // every model starts with the same tracer, unless a
00036     // specific one is set
00037     _modelInfo = new ModelInfo();
00038     _eventManager = new OnEventManager(); // should be on .h (all that does not depends on THIS)
00039     _elementManager = new ElementManager(this);
00040     _componentManager = new ComponentManager(this);
00041     _simulation = new ModelSimulation(this);
00042     // 1:1 associations (Traits)
00043     //Sampler_if* sampler = new Traits<Sampler_if>::Implementation();
00044     _parser = new Traits<Parser_if>::Implementation(this, new Traits<Sampler_if>::Implementation());
00045     _modelChecker = new Traits<ModelChecker_if>::Implementation(this);
00046     _modelPersistence = new Traits<ModelPersistence_if>::Implementation(this);
00047     // 1:n associations
00048     _futureEvents = new List<Event*>();
00049     //_events->setSortFunc(&EventCompare); // It works too
00050     _futureEvents->setSortFunc([](const Event* a, const Event * b) {
00051         return a->getTime() < b->getTime();
00052     });
00053     // for process analyser
00054     _responses = new List<SimulationResponse*>();
00055     _controls = new List<SimulationControl*>();
00056     // insert controls
00057     _controls->insert(new SimulationControl("ModelSimulation", "NumberOfReplications",
00058         DefineGetterMember<ModelSimulation>(this->_simulation,
00059             &ModelSimulation::getNumberOfReplications),
00060             DefineSetterMember<ModelSimulation>(this->_simulation,
00061                 &ModelSimulation::setNumberOfReplications));
00062     _controls->insert(new SimulationControl("ModelSimulation", "ReplicationLength",
00063         DefineGetterMember<ModelSimulation>(this->_simulation,
00064             &ModelSimulation::getReplicationLength),
00065             DefineSetterMember<ModelSimulation>(this->_simulation,
00066                 &ModelSimulation::setReplicationLength));
00067     _controls->insert(new SimulationControl("ModelSimulation", "WarmupPeriod",
00068         DefineGetterMember<ModelSimulation>(this->_simulation, &ModelSimulation::getWarmUpPeriod),
00069             DefineSetterMember<ModelSimulation>(this->_simulation, &ModelSimulation::setWarmUpPeriod));
00070 void Model::sendEntityToComponent(Entity* entity, Connection* connection, double timeDelay) {
00071     this->sendEntityToComponent(entity, connection->first, timeDelay, connection->second);
00072 }
00073
00074 void Model::sendEntityToComponent(Entity* entity, ModelComponent* component, double timeDelay,
00075     unsigned int componentInputNumber) {
00076     this->getOnEvents()->NotifyEntityMoveHandlers(new
00077         SimulationEvent(_simulation->getCurrentReplicationNumber(), new
00078             Event(_simulation->getSimulatedTime(), entity, component, componentInputNumber))); // \todo: Event
00079     // should include information about "from component" and timeDelay, but it doesn't
00080     // schedule to send it
00081     Event* newEvent = new Event(this->getSimulation()->getSimulatedTime() + timeDelay, entity,
00082         component, componentInputNumber);
00083     this->getFutureEvents()->insert(newEvent);
00084 }
00085
00086
00087
00088
00089 bool Model::save(std::string filename) {
00090     bool res = this->_modelPersistence->save(filename);
00091     if (res) {
00092         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model successfully saved");
00093     } else {
00094         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model could not be saved");
00095     }
00096 }
00097
00098
00099
00100 bool Model::load(std::string filename) {
00101     this->clear();

```

```

00102     bool res = this->_modelPersistence->load(filename);
00103     if (res)
00104         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model successfully loaded");
00105     else
00106         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model could not be loaded");
00107     return res;
00108 }
00109
00110 double Model::parseExpression(const std::string expression) {
00111     try {
00112         return _parser->parse(expression);
00113     } catch (...) {
00114         return 0.0; // \todo: HOW SAY THERE WAS AN ERROR?
00115     }
00116 }
00117
00118 bool Model::checkExpression(const std::string expression, const std::string expressionName,
00119     std::string* errorMessage) {
00120     bool result;
00121     parseExpression(expression, &result, errorMessage);
00122     if (!result) {
00123         std::string msg = "Expression \"\" + expression + "\" for '" + expressionName + "' is
00124         incorrect. ";
00125         errorMessage->append(msg);
00126     }
00127     return result;
00128 }
00129
00130 double Model::parseExpression(const std::string expression, bool* success, std::string* errorMessage)
00131 {
00132     double value = _parser->parse(expression, success, errorMessage);
00133     return value;
00134 }
00135
00136 void Model::show() {
00137     getTracer()->trace(Util::TraceLevel::report, "Simulation Model:");
00138     Util::IncIndent();
00139     {
00140         getTracer()->trace(Util::TraceLevel::report, "Information:");
00141         Util::IncIndent();
00142         getTracer()->trace(Util::TraceLevel::report, this->getInfos()->show());
00143         Util::DecIndent();
00144         _showConnections();
00145         _showComponents();
00146         _showElements();
00147         _showSimulationControls();
00148         _showSimulationResponses();
00149     }
00150     Util::DecIndent();
00151     getTracer()->trace(Util::TraceLevel::report, "End of Simulation Model");
00152 }
00153
00154 bool Model::insert(ModelElement* elemOrComp) {
00155     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00156     if (comp == nullptr) // it's a ModelElement
00157         return this->getElements()->insert(elemOrComp);
00158     else // it's a ModelComponent
00159         return this->getComponents()->insert(comp);
00160 }
00161
00162 void Model::remove(ModelElement* elemOrComp) {
00163     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00164     if (comp == nullptr) // it's a ModelElement
00165         this->getElements()->remove(elemOrComp);
00166     else // it's a ModelComponent
00167         this->getComponents()->remove(comp);
00168 }
00169
00170 void Model::_showElements() const {
00171     getTracer()->trace(Util::TraceLevel::report, "Elements:");
00172     Util::IncIndent();
00173     {
00174         std::string elementType;
00175         ModelElement* element;
00176         std::list<std::string>* elementTypes = getElements()->getElementClassnames();
00177         for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end(); typeIt++) {
00178             elementType = (*typeIt);
00179             List<ModelElement*>* em = getElements()->getElementList(elementType);
00180             getTracer()->trace(Util::TraceLevel::report, elementType + ":");
00181             Util::IncIndent();
00182             {
00183                 for (std::list<ModelElement*>::iterator it = em->list()->begin(); it != em->list()->end(); it++) {
00184                     element = (*it);
00185                     getTracer()->trace(Util::TraceLevel::report, element->show());
00186                 }
00187             }
00188         }
00189     }
00190 }

```

```

00184         }
00185         Util::DecIndent();
00186     }
00187 }
00188 Util::DecIndent();
00189 }
00190
00191 void Model::_showConnections() const {
00192 // \todo
00193 }
00194
00195 void Model::_showComponents() const {
00196     getTracer()->trace(Util::TraceLevel::report, "Components:");
00197     Util::IncIndent();
00198     for (std::list<ModelComponent*>::iterator it = _components->begin(); it != _components->end(); it++) {
00199         getTracer()->trace(Util::TraceLevel::report, (*it)->show());
00200     }
00201     Util::DecIndent();
00202 }
00203
00204 void Model::_showSimulationControls() const {
00205     getTracer()->trace(Util::TraceLevel::report, "Simulation Controls:");
00206     Util::IncIndent();
00207     for (std::list<SimulationControl*>::iterator it = _controls->list()->begin(); it != _controls->list()->end(); it++) {
00208         getTracer()->trace(Util::TraceLevel::report, (*it)->show());
00209     }
00210     Util::DecIndent();
00211 }
00212
00213 void Model::_showSimulationResponses() const {
00214     getTracer()->trace(Util::TraceLevel::report, "Simulation Responses:");
00215     Util::IncIndent();
00216     for (std::list<SimulationResponse*>::iterator it = _responses->list()->begin(); it != _responses->list()->end(); it++) {
00217         getTracer()->trace(Util::TraceLevel::report, (*it)->show());
00218     }
00219     Util::DecIndent();
00220 }
00221
00222 void Model::clear() {
00223     this->_componentManager->clear();
00224     this->elementManager->clear();
00225 //Util::ResetAllIds(); // \todo: To implement
00226 }
00227
00228 void Model::_createModelInternalElements() {
00229     getTracer()->trace(Util::TraceLevel::modelInternal, "Creating internal elements");
00230     Util::IncIndent();
00231
00232     for (std::list<ModelComponent*>::iterator it = _componentManager->begin(); it != _componentManager->end(); it++) {
00233         getTracer()->trace(Util::TraceLevel::modelInternal, "Internals for " + (*it)->getClassName() +
00234 " \" " + (*it)->getName() + "\" ");
00235         Util::IncIndent();
00236         ModelComponent::CreateInternalElements((*it));
00237         Util::DecIndent();
00238     }
00239
00240     std::list<ModelElement*>* modelElements;
00241     unsigned int originalSize = getElements()->elementClassnames()->size(), pos = 1;
00242     //for (std::list<std::string>::iterator itty = elements()->elementClassnames()->begin(); itty != elements()->elementClassnames()->end(); itty++) {
00243     std::list<std::string>::iterator itty = getElements()->elementClassnames()->begin();
00244     while (itty != getElements()->elementClassnames()->end() && pos <= originalSize) {
00245         //try {
00246             modelElements = getElements()->elementList((itty))->list();
00247             //} catch (const std::exception& e) {
00248             // \todo Is there a better solution to iterate over a changing sorted list??
00249             // ops. Sorted list has changed and iteration fails. Starts iterating again
00250             // itty = elements()->elementClassnames()->begin();
00251             // modelElements = elements()->elementList((itty))->list();
00252             // tracer()->trace(Util::TraceLevel::modelInternal, "Creating internal elements");
00253             for (std::list<ModelElement*>::iterator itel = modelElements->begin(); itel != modelElements->end(); itel++) {
00254                 getTracer()->trace(Util::TraceLevel::modelInternal, "Internals for " +
00255 (*itel)->getClassName() + " \" " + (*itel)->getName() + "\" "); // (" + std::to_string(pos) + "/" +
00256 std::to_string(originalSize) + ")");
00257                 Util::IncIndent();
00258                 ModelElement::CreateInternalElements((*itel));
00259                 Util::DecIndent();
00260             }
00261             if (originalSize == getElements()->elementClassnames()->size()) {
00262                 itty++;
00263                 pos++;
00264             }
00265         }
00266     }
00267 }

```

```

00262     } else {
00263         originalSize = getElements()->getElementClassnames()->size();
00264         itty = getElements()->getElementClassnames()->begin();
00265         pos = 1;
00266         getTracer()->trace(Util::TraceLevel::modelInternal, "Restarting to create internal
elements (due to previous creations)");
00267     }
00268 }
00269 Util::DecIndent();
00270 }
00271
00272 bool Model::check() {
00273     getTracer()->trace(Util::TraceLevel::modelInternal, "Checking model consistency");
00274     Util::IncIndent();
00275     // before checking the model, creates all necessary internal ModelElements
00276     _createModelInternalElements();
00277     bool res = this->_modelChecker->checkAll();
00278     Util::DecIndent();
00279     if (res) {
00280         getTracer()->trace(Util::TraceLevel::modelResult, "End of Model checking: Success");
00281     } else {
00282         //std::exception e = new std::exception();
00283         //getTrace()->traceError();
00284         getTracer()->trace(Util::TraceLevel::modelResult, "End of Model checking: Failed");
00285     }
00286     return res;
00287 }
00288
00289 //bool Model::verifySymbol(std::string componentName, std::string expressionName, std::string
expression, std::string expressionResult, bool mandatory) {
00290 //    return this->_modelChecker->verifySymbol(componentName, expressionName, expression,
expressionResult, mandatory);
00291 //}
00292
00293 void Model::removeEntity(Entity* entity, bool collectStatistics) {
00294     /* \todo: -: event onEntityRemove */
00295     /* \todo: -: collectStatistics */
00296     std::string entId = std::to_string(entity->entityNumber());
00297     this->getElements()->remove(Util::TypeOf<Entity>(), entity);
00298     getTracer()->trace("Entity " + entId + " was removed from the system");
00299 }
00300
00301 List<Event*>* Model::getFutureEvents() const {
00302     return _futureEvents;
00303 }
00304
00305 void Model::setTracer(TraceManager * _traceManager) {
00306     this->_traceManager = _traceManager;
00307 }
00308
00309 TraceManager * Model::getTracer() const {
00310     return _traceManager;
00311 }
00312
00313 bool Model::hasChanged() const {
00314     bool changed = _hasChanged;
00315     changed &= this->_componentManager->hasChanged();
00316     changed &= this->elementManager->hasChanged();
00317     changed &= this->_modelInfo->hasChanged();
00318     changed &= this->_modelPersistence->hasChanged();
00319     return changed;
00320 }
00321
00322 ComponentManager * Model::getComponents() const {
00323     return _componentManager;
00324 }
00325
00326 List<SimulationControl*>* Model::getControls() const {
00327     return _controls;
00328 }
00329
00330 List<SimulationResponse*>* Model::getResponses() const {
00331     return _responses;
00332 }
00333
00334 OnEventManager * Model::getOnEvents() const {
00335     return _eventManager;
00336 }
00337
00338 ElementManager * Model::getElements() const {
00339     return _elementManager;
00340 }
00341
00342 ModelInfo * Model::getInfos() const {
00343     return _modelInfo;
00344 }
00345

```

```

00346 Simulator * Model::getParentSimulator() const {
00347     return _parentSimulator;
00348 }
00349
00350 ModelSimulation * Model::getSimulation() const {
00351     return _simulation;
00352 }
00353
00354 Util::identification Model::getId() const {
00355     return _id;
00356 }
00357

```

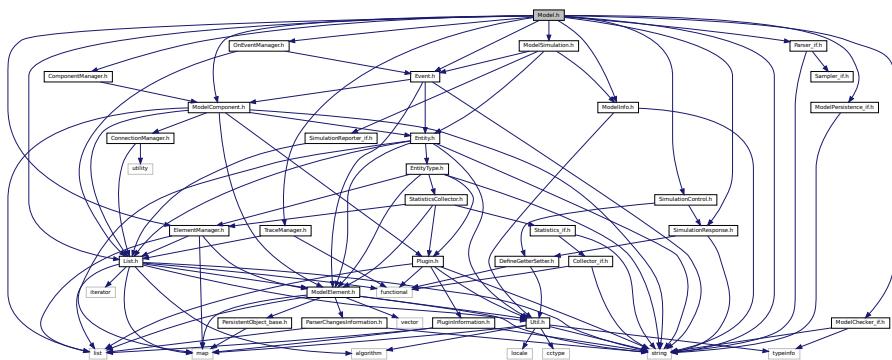
9.221 Model.h File Reference

```

#include <string>
#include "List.h"
#include "ModelComponent.h"
#include "Event.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "ModelPersistence_if.h"
#include "ElementManager.h"
#include "ComponentManager.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](#)

9.222 Model.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationModel.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 15:01
00012 */
00013
00014 #ifndef SIMULATIONMODEL_H
00015 #define SIMULATIONMODEL_H
00016
00017 #include <string>
00018
00019 #include "List.h"
00020 #include "ModelComponent.h"
00021 #include "Event.h"
00022 #include "ModelChecker_if.h"
00023 #include "Parser_if.h"
00024 #include "ModelPersistence_if.h"
00025 #include "ElementManager.h"
00026 #include "ComponentManager.h"
00027 #include "TraceManager.h"
00028 #include "OnEventManager.h"
00029 #include "ModelInfo.h"
00030 #include "ModelSimulation.h"
00031 //for PAN
00032 #include "SimulationResponse.h"
00033 #include "SimulationControl.h"
00034
00035 //namespace GenesysKernel {
00036     class Simulator;
00037
00044     class Model {
00045         public:
00046             Model(Simulator* simulator);
00047             virtual ~Model() = default;
00048         public: // model control
00049             //void showReports();
00050             bool save(std::string filename);
00051             bool load(std::string filename);
00052             bool check();
00053             void clear();
00054             void show();
00055             bool insert(ModelElement* elemOrComp);
00056             void remove(ModelElement* elemOrComp);
00057             //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.
00058             void removeEntity(Entity* entity, bool collectStatistics);
00059             void sendEntityToComponent(Entity* entity, Connection* connection, double timeDelay);
00060             void sendEntityToComponent(Entity* entity, ModelComponent* component, double timeDelay,
unsigned int componentInputNumber = 0);
00061             double parseExpression(const std::string expression);
00062             double parseExpression(const std::string expression, bool* success, std::string*
errorMessage);
00063             bool checkExpression(const std::string expression, const std::string expressionName,
std::string* errorMessage);
00064         public: // only gets
00065             Util::identification getId() const;
00066             bool hasChanged() const;
00067             // 1:1
00068             OnEventManager* getOnEvents() const;
00069             ElementManager* getElements() const;
00070             ComponentManager* getComponents() const;
00071             ModelInfo* getInfos() const;
00072             Simulator* getParentSimulator() const;
00073             ModelSimulation* getSimulation() const;
00074             // 1:n
00075             //List<ModelComponent*>* getComponents() const; // Returns the list of components (such as
Create, Delay, Dispose, etc.) that make up the simulation model.
00076             List<Event*>* getFutureEvents() const;
00077             List<SimulationControl*>* getControls() const;
00078             List<SimulationResponse*>* getResponses() const;
00079         public: // gets and sets
00080             void setTracer(TraceManager* _traceManager);
00081             TraceManager* getTracer() const;
00082             /*
00083             * PRIVATE
00084             */

```

```

00085     private:
00086         void _showConnections() const;
00087         void _showComponents() const;
00088         void _showElements() const;
00089         void _showSimulationControls() const;
00090         void _showSimulationResponses() const;
00091         void _createModelInternalElements();
00092     private:
00093         bool _hasChanged = false;
00094     private: // read only public access (gets)
00095         Util::identification _id;
00096         Simulator* _parentSimulator;
00097         // 1:1 (associated classes)
00098         TraceManager* _traceManager;
00099         OnEventManager* _eventManager;
00100         ElementManager* _elementManager;
00101         ComponentManager* _componentManager;
00102         ModelInfo* _modelInfo;
00103         ModelSimulation* _simulation;
00104         // 1:n
00105         //List<ModelComponent*>* _components;
00106         List<Event*>* _futureEvents;
00107         // for process analyser
00108         List<SimulationResponse*>* _responses;
00109         List<SimulationControl*>* _controls;
00110
00111     private: // no public access (no gets / sets)
00112         ModelChecker_if* _modelChecker;
00113         ModelPersistence_if* _modelPersistence;
00114         Parser_if* _parser;
00115     };
00116 //namespace\\}
00117 #endif /* SIMULATIONMODEL_H */
00118

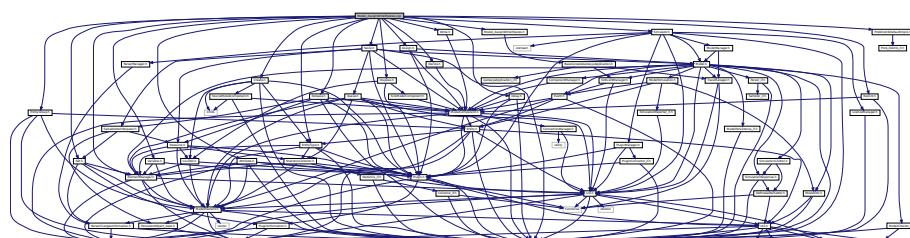
```

9.223 Model_AssignWrite3Seizes.cpp File Reference

```

#include "Model_AssignWrite3Seizes.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Write.h"
#include "ElementManager.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistribDefaultImpl1.h"
#include "EntityGroup.h"
#include "Set.h"
Include dependency graph for Model_AssignWrite3Seizes.cpp:

```



9.224 Model_AssignWrite3Seizes.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FourthExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 20:56
00012 */
00013
00014 #include "Model_AssignWrite3Seizes.h"
00015
00016 // GEnSyS Simulator
00017 #include "Simulator.h"
00018
00019 // Model Components
00020 #include "Create.h"
00021 #include "Delay.h"
00022 #include "Dispose.h"
00023 #include "Seize.h"
00024 #include "Release.h"
00025 #include "Assign.h"
00026 #include "Record.h"
00027 #include "Decide.h"
00028 #include "Write.h"
00029
00030 // Model elements
00031 #include "ElementManager.h"
00032 #include "EntityType.h"
00033 #include "Attribute.h"
00034 #include "Variable.h"
00035 #include "ProbDistribDefaultImpl1.h"
00036 #include "EntityGroup.h"
00037 #include "Set.h"
00038
00039 Model_AssignWrite3Seizes::Model_AssignWrite3Seizes() {
00040 }
00041
00042 int Model_AssignWrite3Seizes::main(int argc, char** argv) {
00043     Simulator* genesys = new Simulator();
00044     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed); //modelResult);
//componentArrival);
00045     this->setDefaultTraceHandlers(genesys->getTracer());
00046     this->insertFakePluginsByHand(genesys);
00047
00048     Model* model = genesys->getModels()->newModel();
00049     this->setDefaultEventHandlers(model->getOnEvents());
00050
00051     // build the simulation model
00052     ModelSimulation* sim = model->getSimulation();
00053     sim->setNumberOfReplications(5);
00054     sim->setReplicationLength(100);
00055     EntityType* part = new EntityType(model, "Part");
00056     // model->insert(part);
00057     Create* createl = new Create(model);
00058     createl->setEntityType(part);
00059     createl-> setTimeBetweenCreationsExpression("norm(1.5,0.5)");
00060     createl-> setTimeUnit(Util::TimeUnit::second);
00061     createl-> setEntitiesPerCreation(1);
00062     // model->insert(createl);
00063     Assign* assignl = new Assign(model);
00064     assignl->getAssignments()->insert(new Assign::Assignment("varNextIndex", "varNextIndex + 1"));
00065     assignl->getAssignments()->insert(new Assign::Assignment("index", "varNextIndex"));
// model->insert(assignl);
00066     Attribute* attrl = new Attribute(model, "index");
00067     // model->insert(attrl);
00068     Variable* varl = new Variable(model, "varNextIndex");
// model->insert(varl);
00069     Write* writel = new Write(model);
00070     writel->setWriteToType(Write::WriteToType::SCREEN);
00071     writel->writeElements()->insert(new WriteElement("Atributo index: "));
00072     writel->writeElements()->insert(new WriteElement("index", true, true));
00073     writel->writeElements()->insert(new WriteElement("Variável nextIndex: "));
00074     writel->writeElements()->insert(new WriteElement("varNextIndex", true, true));
00075     writel->writeElements()->insert(new WriteElement("Quantidade ocupada das máquinas: "));
00076     writel->writeElements()->insert(new WriteElement("NR( Machine_1 )", true));
00077     writel->writeElements()->insert(new WriteElement(" ", ""));
00078     writel->writeElements()->insert(new WriteElement("NR(Machine_2)", true));
00079     writel->writeElements()->insert(new WriteElement(" ", ""));
00080     writel->writeElements()->insert(new WriteElement("NR(Machine_3)", true, true));
00081     writel->writeElements()->insert(new WriteElement("Estado das máquinas: "));
00082     writel->writeElements()->insert(new WriteElement("STATE(Machine_1)", true));
00083     writel->writeElements()->insert(new WriteElement("STATE(Machine_1)", true));

```

```

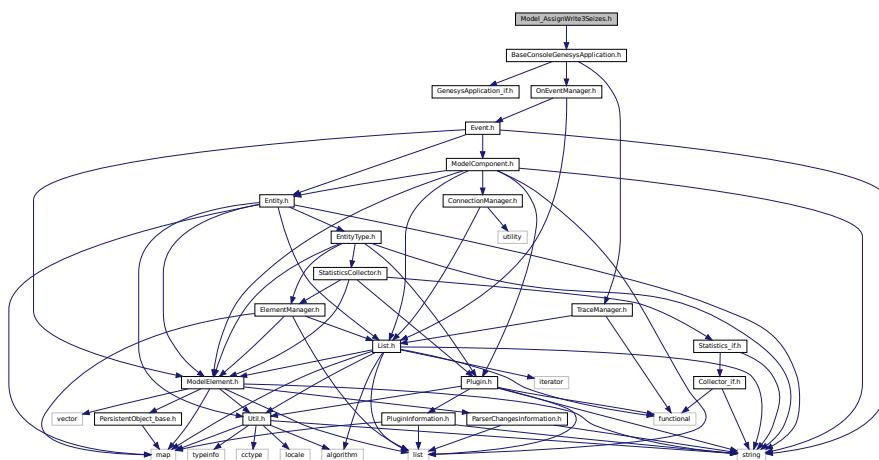
00085     write1->writeElements()->insert(new WriteElement(" ", ""));
00086     write1->writeElements()->insert(new WriteElement("STATE(Machine_2)", true));
00087     write1->writeElements()->insert(new WriteElement(" ", ""));
00088     write1->writeElements()->insert(new WriteElement("STATE(Machine_3)", true, true));
00089     write1->writeElements()->insert(new WriteElement("Quantidade de máquinas ocupadas no Set: "));
00090     write1->writeElements()->insert(new WriteElement("SETSUM(Machine_Set)", true, true));
00091     write1->writeElements()->insert(new WriteElement("Quantidade de entidades na fila 3: "));
00092     write1->writeElements()->insert(new WriteElement("NO(Queue_Seize_3)", true, true));
00093     write1->writeElements()->insert(new WriteElement("Somatório do atributo 'index' das entidades na
00094     fila 3: "));
00094     write1->writeElements()->insert(new WriteElement("SAQUE(Queue_Seize_3,index)", true, true));
00095     write1->writeElements()->insert(new WriteElement("Valor do atributo 'index' da 2ª entidade na fila
00096     3: "));
00096     write1->writeElements()->insert(new WriteElement("AQUE(Queue_Seize_3,2,index)", true, true));
00097     write1->writeElements()->insert(new WriteElement("Tempo médio das entidades na fila 3: "));
00098     write1->writeElements()->insert(new WriteElement("TAVG(Queue_Seize_3.TimeInQueue)", true, true));
00099     // model->insert(write1);
00100     //
00101     Resource* machine1 = new Resource(model, "Machine_1");
00102     machine1->setCapacity(1);
00103     // model->insert(machine1);
00104     Resource* machine2 = new Resource(model, "Machine_2");
00105     machine2->setCapacity(2);
00106     // model->insert(machine2);
00107     Resource* machine3 = new Resource(model, "Machine_3");
00108     machine3->setCapacity(3);
00109     // model->insert(machine3);
00110     Set* machSet = new Set(model, "Machine_Set");
00111     machSet->setSetOfType(Util::TypeOf<Resource>());
00112     machSet->getElementSet()->insert(machine1);
00113     machSet->getElementSet()->insert(machine2);
00114     machSet->getElementSet()->insert(machine3);
00115     // model->insert(machSet);
00116     Decide* decide1 = new Decide(model);
00117     decide1->getConditions()->insert("NR(Machine_1) < MR(Machine_1)");
00118     decide1->getConditions()->insert("NR(Machine_2) < MR(Machine_2)");
00119     // model->insert(decide1);
00120     Queue* queueSeize1 = new Queue(model, "Queue_Seize_1");
00121     queueSeize1->setOrderRule(Queue::OrderRule::FIFO);
00122     // model->insert(queueSeize1);
00123     Seize* seize1 = new Seize(model);
00124     seize1->getSeizeRequests()->insert(new SeizableItemRequest(machine1));
00125     seize1->setQueue(queueSeize1);
00126     // model->insert(seize1);
00127     Delay* delay1 = new Delay(model);
00128     delay1->setDelayExpression("norm(15,1)");
00129     delay1->setDelayTimeUnit(Util::TimeUnit::second);
00130     // model->insert(delay1);
00131     Release* release1 = new Release(model);
00132     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine1));
00133     // model->insert(release1);
00134     Queue* queueSeize2 = new Queue(model, "Queue_Seize_2");
00135     queueSeize2->setOrderRule(Queue::OrderRule::FIFO);
00136     // model->insert(queueSeize2);
00137     Seize* seize2 = new Seize(model);
00138     seize2->getSeizeRequests()->insert(new SeizableItemRequest(machine2));
00139     seize2->setQueue(queueSeize2);
00140     // model->insert(seize2);
00141     Delay* delay2 = new Delay(model);
00142     delay2->setDelayExpression("norm(15,1)");
00143     delay2->setDelayTimeUnit(Util::TimeUnit::second);
00144     // model->insert(delay2);
00145     Release* release2 = new Release(model);
00146     release2->getReleaseRequests()->insert(new SeizableItemRequest(machine2));
00147     // model->insert(release2);
00148     Queue* queueSeize3 = new Queue(model, "Queue_Seize_3");
00149     queueSeize3->setOrderRule(Queue::OrderRule::FIFO);
00150     // model->insert(queueSeize3);
00151     Seize* seize3 = new Seize(model);
00152     seize3->getSeizeRequests()->insert(new SeizableItemRequest(machine3));
00153     seize3->setQueue(queueSeize3);
00154     // model->insert(seize3);
00155     Delay* delay3 = new Delay(model);
00156     delay3->setDelayExpression("norm(15,1)");
00157     delay3->setDelayTimeUnit(Util::TimeUnit::second);
00158     // model->insert(delay3);
00159     Release* release3 = new Release(model);
00160     release3->getReleaseRequests()->insert(new SeizableItemRequest(machine3));
00161     // model->insert(release3);
00162     Dispose* dispose1 = new Dispose(model);
00163     // model->insert(dispose1);
00164     //
00165     createl->getNextComponents()->insert(assign1);
00166     assign1->getNextComponents()->insert(write1);
00167     write1->getNextComponents()->insert(decide1);
00168     decide1->getNextComponents()->insert(seize1);
00169     decide1->getNextComponents()->insert(seize2);

```

```
00170     decide1->getNextComponents()->insert(seize3);
00171     seize1->getNextComponents()->insert(delay1);
00172     delay1->getNextComponents()->insert(release1);
00173     release1->getNextComponents()->insert(dispose1);
00174     seize2->getNextComponents()->insert(delay2);
00175     delay2->getNextComponents()->insert(release2);
00176     release2->getNextComponents()->insert(dispose1);
00177     seize3->getNextComponents()->insert(delay3);
00178     delay3->getNextComponents()->insert(release3);
00179     release3->getNextComponents()->insert(dispose1);
00180     //
00181     model->save("./temp/forthExampleOfSimulation.txt");
00182     sim->start();
00183     return 0;
00184 }
00185 }
```

9.225 Model_AssignWrite3Seizes.h File Reference

```
#include "BaseConsoleGenesysApplication.h"  
Include dependency graph for Model_AssignWrite3Seizes.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Model AssignWrite3Seizes

9.226 Model AssignWrite3Seizes.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: FourthExampleOfSimulation.h  
00009 * Author: rlcancian
```

```

00010  /*
00011  * Created on 24 de Setembro de 2019, 20:56
00012  */
00013
00014 #ifndef FOURTHEXAMPLEOFSIMULATION_H
00015 #define FOURTHEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_AssignWrite3Seizes : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_AssignWrite3Seizes();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* FOURTHEXAMPLEOFSIMULATION_H */
00027

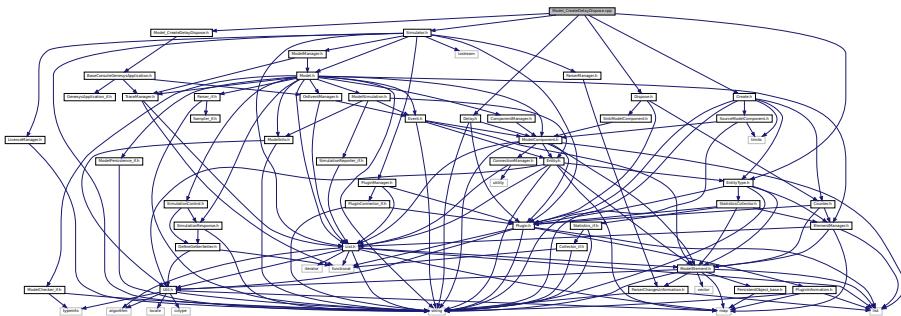
```

9.227 Model_CreateDelayDispose.cpp File Reference

```

#include "Model_CreateDelayDispose.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "EntityType.h"
Include dependency graph for Model_CreateDelayDispose.cpp:

```



9.228 Model_CreateDelayDispose.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: FirstExampleOfSimulation.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 3 de Setembro de 2019, 18:34
00012 */
00013
00014 #include "Model_CreateDelayDispose.h"
00015
00016 // you have to included need libs
00017
00018 // GEnSyS Simulator
00019 #include "Simulator.h"
00020
00021 // Model Components
00022 #include "Create.h"
00023 #include "Delay.h"
00024 #include "Dispose.h"
00025
00026 // Model model

```

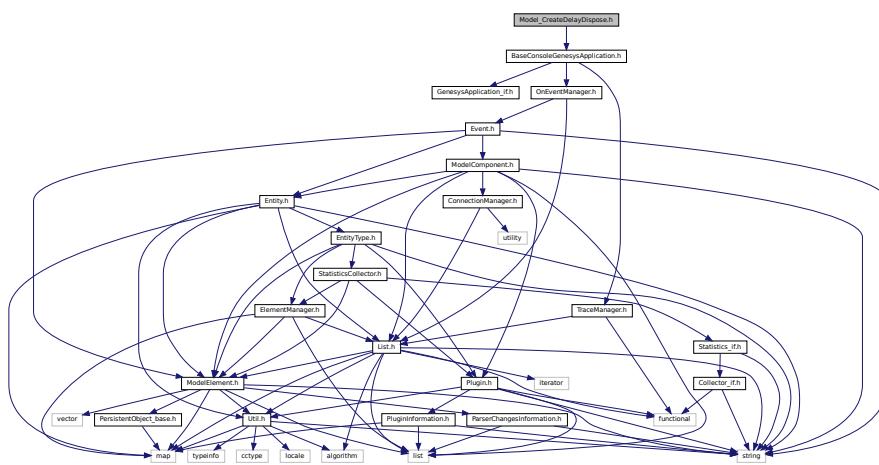
```

00027 #include "EntityType.h"
00028
00029 Model_CreateDelayDispose::Model_CreateDelayDispose() {
00030 }
00031
00036 int Model_CreateDelayDispose::main(int argc, char** argv) {
00037     Simulator* genesys = new Simulator();
00038     // Handle traces and simulation events to output them
00039     this->setDefaultTraceHandlers(genesys->getTracer());
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::componentDetailed);
00041     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00042     this->insertFakePluginsByHand(genesys);
00043     Model* model = genesys->getModels()->newModel();
00044     //
00045     // build the simulation model
00046     // if no ModelInfo is provided, then the model will be simulated once (one replication) and the
replication length will be 3600 seconds (simulated time)
00047     model->getSimulation()->setReplicationLength(60);
00048     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00049     EntityType* entityType1 = new EntityType(model, "Type_of_Representative_Entity");
00050     // create a ModelComponent of type Create, used to insert entities into the model
00051     Create* create1 = new Create(model);
00052     create1->setEntityType(entityType1);
00053     create1-> setTimeBetweenCreationsExpression("1.5"); // create one new entity every 1.5 seconds
00054     // create a ModelComponent of type Delay, used to represent a time delay
00055     Delay* delay1 = new Delay(model);
00056     // if nothing else is set, the delay will take 1 second
00057     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00058     Dispose* dispose1 = new Dispose(model); // insert the component into the model
00059     // connect model components to create a "workflow" -- should always start from a
SourceModelComponent and end at a SinkModelComponent (it will be checked)
00060     create1->getNextComponents()->insert(delay1);
00061     delay1->getNextComponents()->insert(dispose1);
00062     // save the model into a text file
00063     model->save("./models/Model_CreateDelayDispose.txt");
00064     // execute the simulation until completed and show the report
00065     model->getSimulation()->start();
00066     genesys->~Simulator();
00067     return 0;
00068 };
00069

```

9.229 Model_CreateDelayDispose.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
Include dependency graph for Model_CreateDelayDispose.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Model_CreateDelayDispose

9.230 Model_CreateDelayDispose.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: FirstExampleOfSimulation.h
00009  * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 18:34
00012 */
00013
00014 #ifndef FIRSTEXAMPLEOFSIMULATION_H
00015 #define FIRSTEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_CreateDelayDispose : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_CreateDelayDispose();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* FIRSTEXAMPLEOFSIMULATION_H */
00027

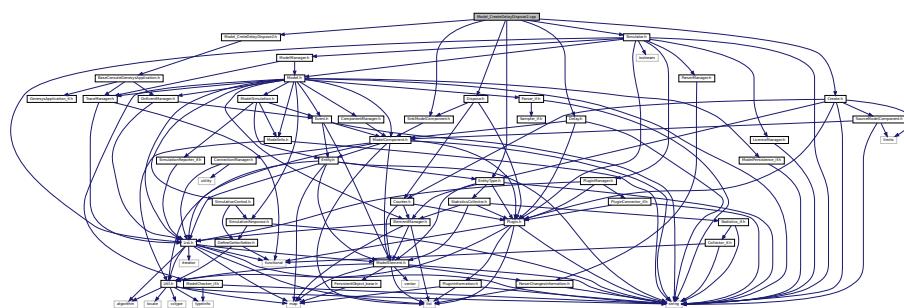
```

9.231 Model_CreateDelayDispose2.cpp File Reference

```

#include "Model_CreteDelayDispose2.h"
#include "SinkModelComponent.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "EntityType.h"
Include dependency graph for Model_CreateDelayDispose2.cpp:

```



9.232 Model_CreateDelayDispose2.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*

```

```

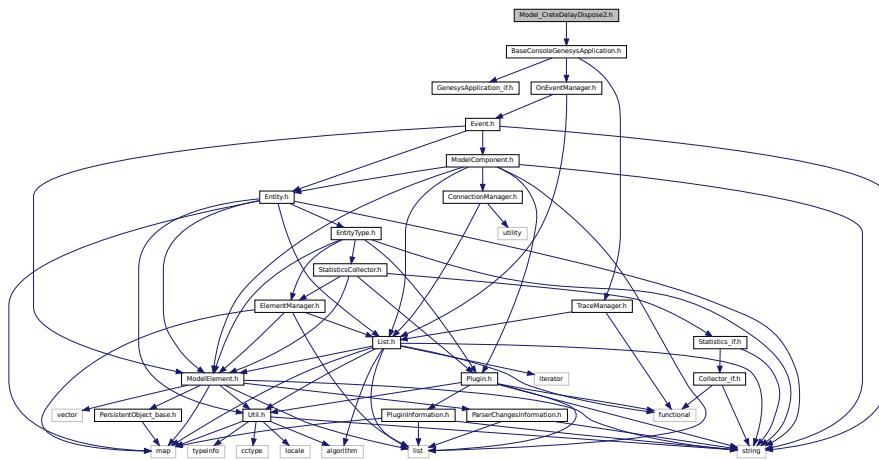
00008 * File: SecondExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 18:59
00012 */
00013
00014 #include "Model_CreteDelayDispose2.h"
00015 #include "SinkModelComponent.h"
00016
00017 // you have to included need libs
00018
00019 // GEnSyS Simulator
00020 #include "Simulator.h"
00021
00022 // Model Components
00023 #include "Create.h"
00024 #include "Delay.h"
00025 #include "Dispose.h"
00026
00027 // Model elements
00028 #include "EntityType.h"
00029
00030 Model_CreteDelayDispose2::Model_CreteDelayDispose2() {
00031 }
00036 int Model_CreteDelayDispose2::main(int argc, char** argv) {
00037     Simulator* genesys = new Simulator();
00038     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of
00039     tracing
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::componentArrival);
00041     // Handle traces and simulation events to output them
00042     this->setDefaultTraceHandlers(genesys->getTracer());
00043     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00044     this->insertFakePluginsByHand(genesys);
00045     Model* model = genesys->getModels()->newModel();
00046     // build the simulation model
00047     ModelInfo* infos = model->getInfos();
00048     infos->setAnalystName("Your name");
00049     infos->setProjectTitle("The title of the project");
00050     infos->setDescription("This simulation model tests one of the most basic models possible.");
00051
00052     ModelSimulation* sim = model->getSimulation();
00053     sim->setReplicationLength(30);
00054     sim->setReplicationLengthTimeUnit(Util::TimeUnit::minute); // each replication will last 30
00055     minutes (simulated time)
00056     sim->setNumberOfReplications(3); // replicates the simulation 3 times
00057     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00058     EntityType* entityTypel = new EntityType(model, "Type_of_Representative_Entity");
00059     // create a ModelComponent of type Create, used to insert entities into the model
00060     Create* createel = new Create(model);
00061     createel->setEntityType(entityTypel);
00062     createel-> setTimeBetweenCreationsExpression("Expo(2)");
00063     createel->setTimeUnit(Util::TimeUnit::minute);
00064     createel->setEntitiesPerCreation(1);
00065     // create a ModelComponent of type Delay, used to represent a time delay
00066     Delay* delayl = new Delay(model);
00067     delayl->setDelayExpression("NORM(1,0.2)");
00068     delayl->setDelayTimeUnit(Util::TimeUnit::minute);
00069     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00070     Dispose* disposel = new Dispose(model);
00071     // connect model components to create a "workflow"
00072     createel->getNextComponents()->insert(delayl);
00073     delayl->getNextComponents()->insert(disposel);
00074     model->save("./models/Model_CreteDelayDispose2.txt");
00075     // execute the simulation
00076     sim->start();
00077
00078     return 0;
00079 };
00080

```

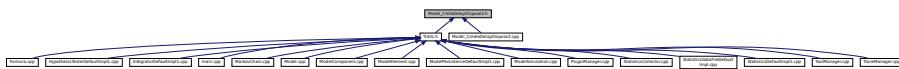
9.233 Model_CreteDelayDispose2.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Include dependency graph for Model_CreteDelayDispose2.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Model_CreateDelayDispose2](#)

9.234 Model_CreteDelayDispose2.h

```

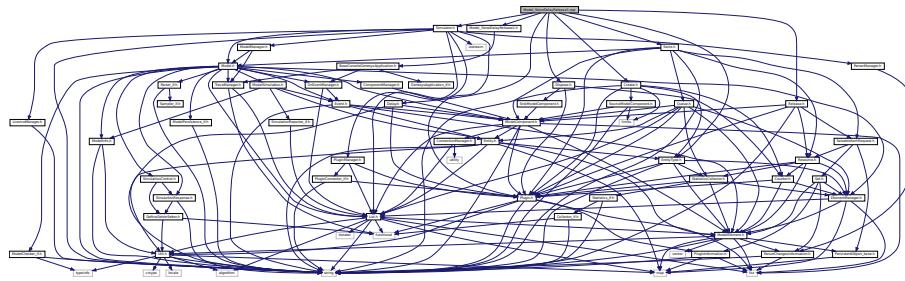
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SecondExampleOfSimulation.h
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 18:59
00012 */
00013
00014 #ifndef SECONDEXAMPLEOFSIMULATION_H
00015 #define SECONDEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_CreateDelayDispose2 : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_CreateDelayDispose2();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* SECONDEXAMPLEOFSIMULATION_H */
00027

```

9.235 Model_SeizeDelayRelease1.cpp File Reference

```
#include "Model_SeizeDelayRelease1.h"
#include "Simulator.h"
#include "Create.h"
#include "Seize.h"
#include "Delay.h"
#include "Release.h"
#include "Dispose.h"
#include "EntityType.h"
```

Include dependency graph for Model_SeizeDelayRelease1.cpp:



9.236 Model_SeizeDelayRelease1.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ThirdExampleOfSimultion.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 16:43
00012 */
00013
00014 #include "Model_SeizeDelayRelease1.h"
00015
00016 // you have to included need libs
00017
00018 // GEnSyS Simulator
00019 #include "Simulator.h"
00020
00021 // Model Components
00022 #include "Create.h"
00023 #include "Seize.h"
00024 #include "Delay.h"
00025 #include "Release.h"
00026 #include "Dispose.h"
00027
00028 // Model elements
00029 #include "EntityType.h"
00030
00031 Model_SeizeDelayRelease1::Model_SeizeDelayRelease1() {
00032 }
00033
00034 int Model_SeizeDelayRelease1::main(int argc, char** argv) {
00035     Simulator* genesys = new Simulator();
00036     // Handle traces and simulation events to output them
00037     this->setDefaultTraceHandlers(genesys->getTracer());
00038     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00039     // insert plugins
00040     this->insertFakePluginsByHand(genesys);
00041     Model* model = genesys->getModels()->newModel();
00042     // build the simulation model
00043     // set model infos
00044     ModelSimulation* sim = model->getSimulation();
00045     sim->setReplicationLength(3600);
00046     sim->setReplicationLengthTimeUnit(Util::TimeUnit::second);
00047     sim->setNumberOfReplications(30);
```

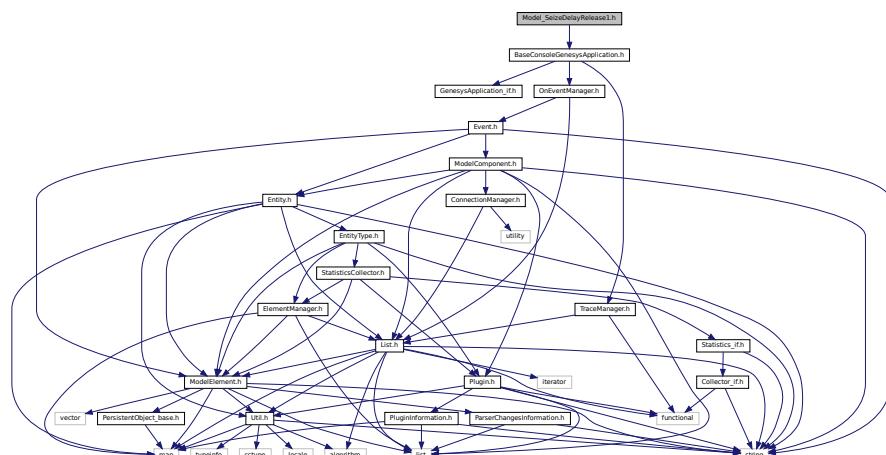
```

00052 // 
00053 EntityType* customer = new EntityType(model, "Customer");
00054 //
00055 Create* createl = new Create(model);
00056 createl->setEntityType(customer);
00057 createl-> setTimeBetweenCreationsExpression("expo(20)");
00058 createl-> setTimeUnit(Util::TimeUnit::second);
00059 createl-> setEntitiesPerCreation(1);
00060 createl-> setFirstCreation(0.0);
00061 //
00062 Resource* machine1 = new Resource(model, "Machine_1");
00063 machine1-> setCapacity(1);
00064 //
00065 Queue* queueSeize1 = new Queue(model, "Seize_1.Queue");
00066 queueSeize1-> setOrderRule(Queue::OrderRule::FIFO);
00067 //
00068 Seize* seize1 = new Seize(model);
00069 //seize1-> setResource(machine1);
00070 //seize1-> setQuantity("1");
00071 seize1-> getSeizeRequests()-> insert(new SeizableItemRequest(machine1, "1"));
00072 seize1-> setQueue(queueSeize1);
00073 //
00074 Delay* delay1 = new Delay(model);
00075 delay1-> setDelayExpression("unif(15,30)");
00076 delay1-> setDelayTimeUnit(Util::TimeUnit::second);
00077 //
00078 Release* release1 = new Release(model);
00079 //release1-> setResource(machine1);
00080 //release1-> setQuantity("1");
00081 release1-> getReleaseRequests()-> insert(new SeizableItemRequest(machine1, "1"));
00082 //
00083 Dispose* dispose1 = new Dispose(model);
00084 // connect model components to create a "workflow"
00085 createl-> getNextComponents()-> insert(seize1);
00086 seize1-> getNextComponents()-> insert(delay1);
00087 delay1-> getNextComponents()-> insert(release1);
00088 release1-> getNextComponents()-> insert(dispose1);
00089 // save the model into a text file
00090 model-> save("./models/Model_SeizeDelayRelease1.txt");
00091 // execute the simulation
00092 sim-> start();
00093
00094 return 0;
00095 };

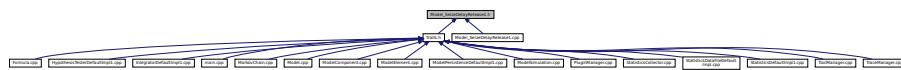
```

9.237 Model_SeizeDelayRelease1.h File Reference

#include "BaseConsoleGenesysApplication.h"
Include dependency graph for Model_SeizeDelayRelease1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Model_SeizeDelayRelease1](#)

9.238 Model_SeizeDelayRelease1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ThirdExampleOfSimultion.h
00009  * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 16:43
00012 */
00013
00014 #ifndef THIRDEXAMPLEOFSIMULTION_H
00015 #define THIRDEXAMPLEOFSIMULTION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_SeizeDelayRelease1 : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_SeizeDelayRelease1();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025 #endif /* THIRDEXAMPLEOFSIMULTION_H */
00026

```

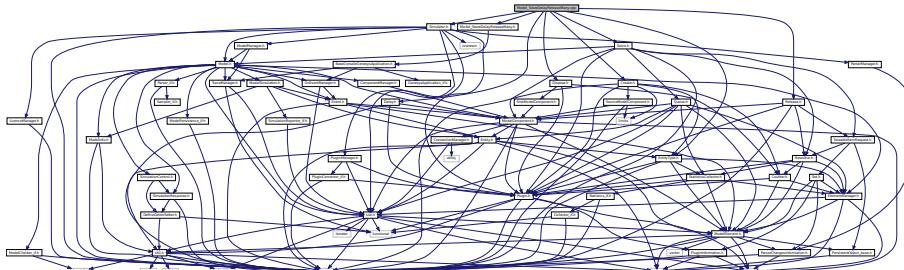
9.239 Model_SeizeDelayReleaseMany.cpp File Reference

```

#include "Model_SeizeDelayReleaseMany.h"
#include "Simulator.h"
#include "Create.h"
#include "Seize.h"
#include "Delay.h"
#include "Release.h"
#include "Dispose.h"
#include "EntityType.h"

```

Include dependency graph for Model_SeizeDelayReleaseMany.cpp:



9.240 Model_SeizeDelayReleaseMany.cpp

```

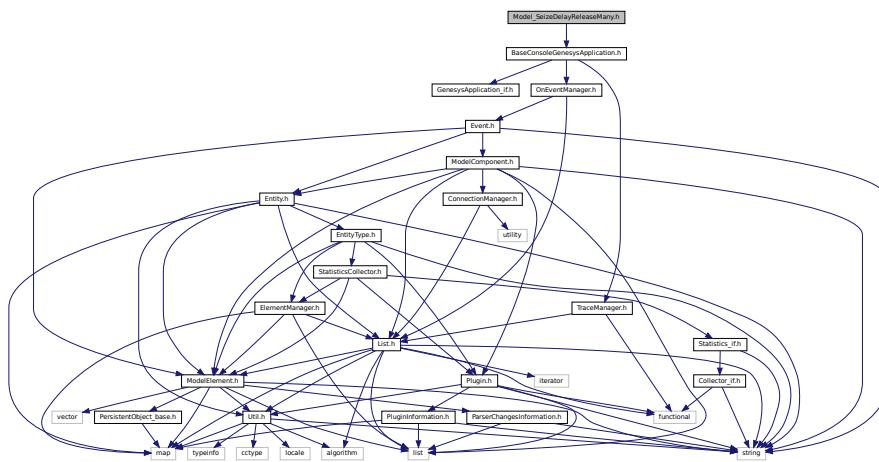
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Model_SeizeDelayReleaseMany.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 9 de abril de 2021, 18:09
00012 */
00013
00014 #include "Model_SeizeDelayReleaseMany.h"
00015
00016 // you have to included need libs
00017
00018 // GEnSys Simulator
00019 #include "Simulator.h"
00020
00021 // Model Components
00022 #include "Create.h"
00023 #include "Seize.h"
00024 #include "Delay.h"
00025 #include "Release.h"
00026 #include "Dispose.h"
00027
00028 // Model elements
00029 #include "EntityType.h"
00030
00031 Model_SeizeDelayReleaseMany::Model_SeizeDelayReleaseMany() {
00032 }
00033
00038 int Model_SeizeDelayReleaseMany::main(int argc, char** argv) {
00039     Simulator* genesys = new Simulator();
00040     this->setDefaultTraceHandlers(genesys->getTracer());
00041     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00042     this->insertFakePluginsByHand(genesys);
00043     Model* m = genesys->getModels()->newModel();
00044     m->getSimulation()->setReplicationLength(60);
00045     EntityType* customer = new EntityType(m);
00046     Create* createl = new Create(m);
00047     createl->setEntityType(customer);
00048     createl-> setTimeBetweenCreationsExpression("unif(1,2)");
00049     Resource* machine1 = new Resource(m);
00050     Resource* machine2 = new Resource(m);
00051     Resource* machine3 = new Resource(m);
00052     Resource* machine4 = new Resource(m);
00053     Queue* queueSeizel = new Queue(m);
00054     queueSeizel->setOrderRule(Queue::OrderRule::FIFO);
00055     Seize* seizel = new Seize(m);
00056     seizel->getSeizeRequests()->insert(new SeizableItemRequest(machine1));
00057     seizel->getSeizeRequests()->insert(new SeizableItemRequest(machine2));
00058     seizel->getSeizeRequests()->insert(new SeizableItemRequest(machine3));
00059     seizel->getSeizeRequests()->insert(new SeizableItemRequest(machine4));
00060     seizel->setQueue(queueSeizel);
00061     Delay* delay1 = new Delay(m);
00062     delay1->setDelayExpression("unif(1,2)");
00063     Release* release1 = new Release(m);
00064     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine1));
00065     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine2));
00066     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine3));
00067     release1->getReleaseRequests()->insert(new SeizableItemRequest(machine4));
00068     Dispose* dispose1 = new Dispose(m);
00069     // connect model components to create a "workflow"
00070     createl->getNextComponents()->insert(seizel);
00071     seizel->getNextComponents()->insert(delay1);
00072     delay1->getNextComponents()->insert(release1);
00073     release1->getNextComponents()->insert(dispose1);
00074     // save the model into a text file
00075     m->save("./models/Model_SeizeDelayReleaseMany.txt");
00076     genesys->getModels()->current()->getSimulation()->start();
00077     return 0;
00078 };

```

9.241 Model_SeizeDelayReleaseMany.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Include dependency graph for Model_SeizeDelayReleaseMany.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Model_SeizeDelayReleaseMany](#)

9.242 Model_SeizeDelayReleaseMany.h

```

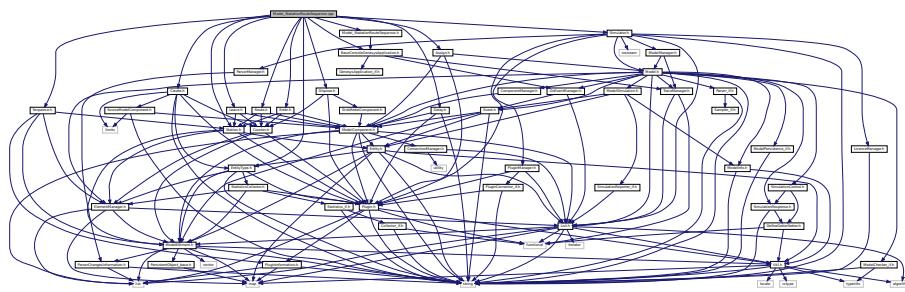
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Model_SeizeDelayReleaseMany.h
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de abril de 2021, 18:09
00012 */
00013
00014 #ifndef MODEL_SEIZEDELAYRELEASEMANY_H
00015 #define MODEL_SEIZEDELAYRELEASEMANY_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_SeizeDelayReleaseMany : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_SeizeDelayReleaseMany();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* MODEL_SEIZEDELAYRELEASEMANY_H */
00027

```

9.243 Model_StatationRouteSequence.cpp File Reference

```
#include "Model_StatationRouteSequence.h"
#include "BaseConsoleGenesysApplication.h"
#include "Simulator.h"
#include "Create.h"
#include "Route.h"
#include "Enter.h"
#include "Leave.h"
#include "Dispose.h"
#include "Station.h"
#include "EntityType.h"
#include "Delay.h"
#include "Sequence.h"
#include "Assign.h"
```

Include dependency graph for Model_StatationRouteSequence.cpp:



9.244 Model_StatationRouteSequence.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FifthExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 12 de março de 2021, 15:39
00012 */
00013
00014 #include "Model_StatationRouteSequence.h"
00015 #include "BaseConsoleGenesysApplication.h"
00016 #include "Simulator.h"
00017 #include "Create.h"
00018 #include "Route.h"
00019 #include "Enter.h"
00020 #include "Leave.h"
00021 #include "Dispose.h"
00022 #include "Station.h"
00023 #include "EntityType.h"
00024 #include "Delay.h"
00025 #include "Sequence.h"
00026 #include "Assign.h"
00027
00028 Model_StatationRouteSequence::Model_StatationRouteSequence() {
00029 }
00030
00031 int Model_StatationRouteSequence::main(int argc, char** argv) {
00032     Simulator* genesys = new Simulator();
00033     this->insertFakePluginsByHand(genesys);
00034     this->setDefaultTraceHandlers(genesys->getTracer());
00035     genesys->getTracer()->setTraceLevel(Util::TraceLevel::modelSimulationEvent);
00036
00037     Model* m = genesys->getModels()->newModel();
00038     m->getSimulation()->setReplicationLength(60);
00039 }
```

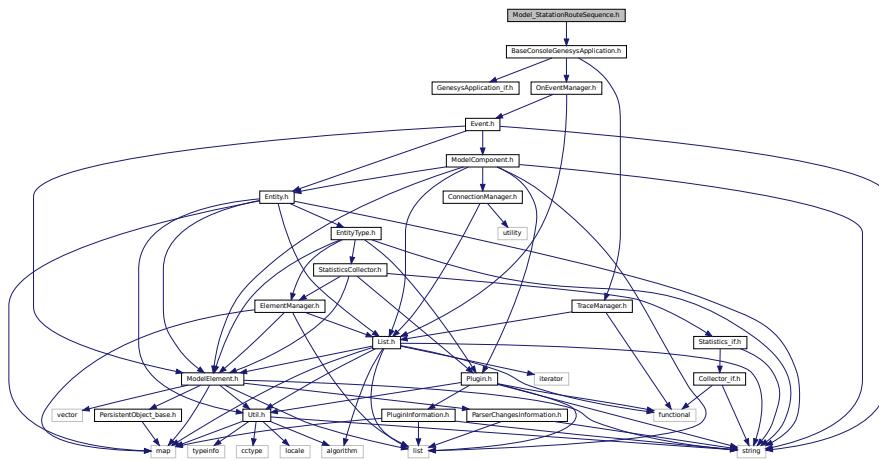
```

00040     Create* cl = new Create(m);
00041     cl->setEntityType(new EntityType(m));
00042     cl-> setTimeBetweenCreationsExpression("10");
00043     Station* s1 = new Station(m);
00044     Route* r0 = new Route(m);
00045     r0->setRouteDestinationType(Route::DestinationType::BySequence);
00046     // (Route::DestinationType::Station);
00047     // r0->setStation(s1);
00048     Enter* e1 = new Enter(m);
00049     e1->setStation(s1);
00050     Delay* d1 = new Delay(m);
00051     Station* s2 = new Station(m);
00052     Leave* l1 = new Leave(m);
00053     l1->setStation(s1);
00054     Route* r1 = new Route(m);
00055     r1->setRouteDestinationType(Route::DestinationType::BySequence);
00056     // (Route::DestinationType::Station);
00057     // r1->setStation(s2);
00058     r1-> setTimeExpression("0.1");
00059     Enter* e2 = new Enter(m);
00060     e2->setStation(s2);
00061     Delay* d2 = new Delay(m);
00062     d2-> setDelayExpression("2");
00063     Leave* l2 = new Leave(m);
00064     l2->setStation(s2);
00065     Station* s3 = new Station(m);
00066     Route* r2 = new Route(m);
00067     r2->setRouteDestinationType(Route::DestinationType::BySequence);
00068     // (Route::DestinationType::Station);
00069     // r2->setStation(s3);
00070     r2-> setTimeExpression("0.2");
00071     Enter* e3 = new Enter(m);
00072     e3->setStation(s3);
00073     Dispose* dpl = new Dispose(m);
00074
00075     Sequence* seq = new Sequence(m);
00076     seq->getSteps()->insert(new SequenceStep(s1));
00077     seq->getSteps()->insert(new SequenceStep(s2));
00078     seq->getSteps()->insert(new SequenceStep(s1));
00079     seq->getSteps()->insert(new SequenceStep(s1));
00080     seq->getSteps()->insert(new SequenceStep(s3));
00081
00082     Assign* a1 = new Assign(m);
00083     a1->getAssignments()->insert(new Assign::Assignment("Entity.Sequence",
00084         std::to_string(seq->getId())));
00085
00086     cl->getNextComponents()->insert(a1);
00087     a1->getNextComponents()->insert(r0);
00088     e1->getNextComponents()->insert(d1);
00089     d1->getNextComponents()->insert(l1);
00090     l1->getNextComponents()->insert(r1);
00091     e2->getNextComponents()->insert(d2);
00092     d2->getNextComponents()->insert(l2);
00093     l2->getNextComponents()->insert(r2);
00094     e3->getNextComponents()->insert(dpl);
00095
00096     ModelSimulation* sim = m->getSimulation();
00097     sim->getBreakpointsOnComponent()->insert(a1);
00098     sim->getBreakpointsOnComponent()->insert(l2);
00099     sim->getBreakpointsOnTime()->insert(40.0);
00100    sim->getBreakpointsOnTime()->insert(20.0);
00101
00102    m->save("./models/Model_StatationRouteSequence.txt");
00103
00104    do {
00105        sim->start();
00106    } while (sim->isPaused());
00107    return 0;
00108 }
```

9.245 Model_StatationRouteSequence.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Include dependency graph for Model_StationRouteSequence.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Model_StationRouteSequence](#)

9.246 Model_StationRouteSequence.h

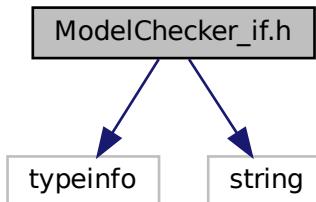
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FifthExampleOfSimulation.h
00009 * Author: rlcancian
00010 *
00011 * Created on 12 de março de 2021, 15:39
00012 */
00013
00014 #ifndef FIFTHEXAMPLEOFSIMULATION_H
00015 #define FIFTHEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Model_StationRouteSequence : public BaseConsoleGenesysApplication {
00020 public:
00021     Model_StationRouteSequence();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* FIFTHEXAMPLEOFSIMULATION_H */
00027

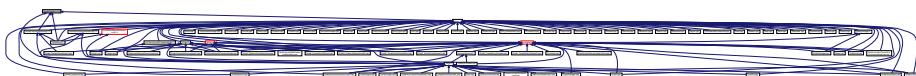
```

9.247 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](#)

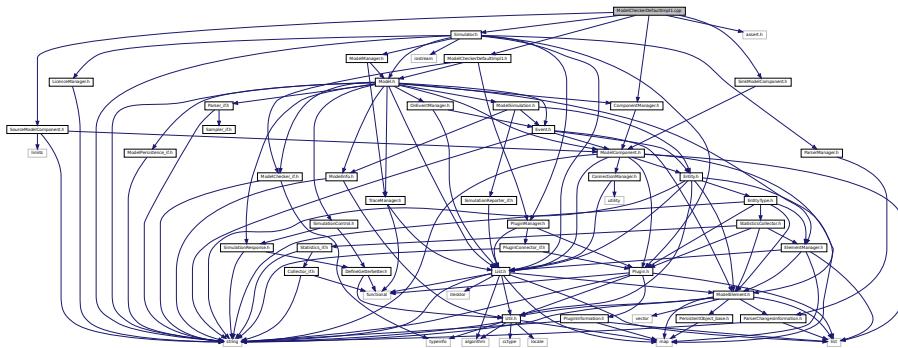
9.248 ModelChecker_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelChecker_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 15:48
00012 */
00013
00014 #ifndef MODELCHECKER_IF_H
00015 #define MODELCHECKER_IF_H
00016
00017 #include <typeinfo>
00018 #include <string>
00019
00020 //##include "Model.h"
00021 //class Model;
00022
00023 class ModelChecker_if {
00024 public:
00025     virtual bool checkAll() = 0;
00026     virtual bool checkConnected() = 0;
00027     virtual bool checkSymbols() = 0;
00028     virtual bool checkActivationCode() = 0;
00029     virtual bool checkLimits() = 0;
00030     //virtual bool verifySymbol(std::string componentName, std::string expressionName, std::string
00031     //expression, std::string expressionResult, bool mandatory) = 0;
00032
00033 };
00034
00035 #endif /* MODELCHECKER_IF_H */
00036
00037
00038
  
```

9.249 ModelCheckerDefaultImpl1.cpp File Reference

```
#include "ModelCheckerDefaultImpl1.h"
#include "SourceModelComponent.h"
#include "SinkModelComponent.h"
#include "ComponentManager.h"
#include "Simulator.h"
#include <assert.h>
Include dependency graph for ModelCheckerDefaultImpl1.cpp:
```



9.250 ModelCheckerDefaultImpl1.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelCheckerDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 18:44
00012 */
00013
00014 #include "ModelCheckerDefaultImpl1.h"
00015 #include "SourceModelComponent.h"
00016 #include "SinkModelComponent.h"
00017 #include "ComponentManager.h"
00018 #include "Simulator.h"
00019
00020 #include <assert.h>
00021
00022 //using namespace GenesysKernel;
00023
00024 ModelCheckerDefaultImpl1::ModelCheckerDefaultImpl1(Model* model) {
00025     _model = model;
00026 }
00027
00028 bool ModelCheckerDefaultImpl1::checkAll() {
00029     bool res = true;
00030     res &= checkSymbols();
00031     //res &= checkAndAddInternalLiterals();
00032     //res &= checkActivationCode();
00033     res &= checkLimits();
00034     res &= checkConnected();
00035     return res;
00036 }
00037
00038 //bool ModelCheckerDefaultImpl1::checkAndAddInternalLiterals() {
00039 //    /* \todo: +-: not implemented yet */
00040 //    return true;
00041 //}
00042
00043 void ModelCheckerDefaultImpl1::_recursiveConnectedTo(PluginManager* pluginManager, ModelComponent*
00044     comp, List<ModelComponent*>* visited, List<ModelComponent*>* unconnected, bool* drenoFound) {
00044     visited->insert(comp);
00045     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Connected to component \""
00045         comp->getName() + "\"");
```

```

00046     Plugin* plugin = pluginManager->find(comp->getClassName());
00047     assert(plugin != nullptr);
00048     if (plugin->getPluginInfo()->isSink() || (plugin->getPluginInfo()->isSendTransfer() &&
00049         comp->getNextComponents()->size() == 0)) { //dynamic_cast<SinkModelComponent*> (comp) != nullptr) {
00050         // it is a sink OR it can send entities through a transfer and has no nextConnections
00051         *drenoFound = true;
00052     } else { // it is not a sink
00053         if (comp->getNextComponents()->size() == 0) {
00054             unconnected->insert(comp);
00055             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Component \""
00056             + comp->getName() + "\" is unconnected (not a sink with no next components connected to)");
00057             *drenoFound = false;
00058         } else {
00059             ModelComponent* nextComp;
00060             for (std::list<Connection*>::iterator it = comp->getNextComponents()->list()->begin(); it !=
00061                 comp->getNextComponents()->list()->end(); it++) {
00062                 nextComp = (*it)->first;
00063                 if (visited->find(nextComp) == visited->list()->end()) { // not visited yet
00064                     *drenoFound = false;
00065                     Util::IncIndent();
00066                     this->recursiveConnectedTo(pluginManager, nextComp, visited, unconnected,
00067                     drenoFound);
00068                     Util::DecIndent();
00069                 } else {
00070                     Util::IncIndent();
00071                     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Connected to "
00072                     + nextComp->getName());
00073                     Util::DecIndent();
00074                     *drenoFound = true;
00075                 }
00076             }
00077             /* \todo: +-: not implemented yet */
00078             _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking connected");
00079             bool resultAll = true;
00080             PluginManager* pluginManager = this->_model->getParentSimulator()->getPlugins();
00081             Plugin* plugin;
00082             Util::IncIndent();
00083             {
00084                 List<ModelComponent*>* visited = new List<ModelComponent*>();
00085                 List<ModelComponent*>* unconnected = new List<ModelComponent*>();
00086                 ModelComponent* comp;
00087                 for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it !=
00088                     _model->getComponents()->end(); it++) {
00089                     comp = (*it);
00090                     plugin = pluginManager->find(comp->getClassName());
00091                     assert(plugin != nullptr);
00092                     if (plugin->getPluginInfo()->isSource() || plugin->getPluginInfo()->isReceiveTransfer()) {
00093                         //dynamic_cast<SourceModelComponent*> (comp) != nullptr) {
00094                             // it is a source component OR it can receive entities from transfer
00095                             bool drenoFound = false;
00096                             recursiveConnectedTo(pluginManager, comp, visited, unconnected, &drenoFound);
00097                         }
00098                         // check if any component remains unconnected
00099                         for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it !=
00100                             _model->getComponents()->end(); it++) {
00101                             comp = (*it);
00102                             if (visited->find(comp) == visited->list()->end()) { // not found
00103                                 resultAll = false;
00104                                 _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Component \""
00105                                 + comp->getName() + "\" is unconnected.");
00106                             }
00107                         Util::DecIndent();
00108                     }
00109             }
00110             Util::DecIndent();
00111             return resultAll;
00112         }
00113         bool ModelCheckerDefaultImpl1::checkSymbols() {
00114             bool res = true;
00115             _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking symbols");
00116             Util::IncIndent();
00117             {
00118                 // check components
00119                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Components:");
00120                 Util::IncIndent();
00121                 {
00122                     //List<ModelComponent*>* components = _model->getComponents();
00123                     for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it !=
00124                         _model->getComponents()->end(); it++) {
00125                         res &= (*it)->Check((*it));
00126                     }
00127                 }
00128             }
00129         }
00130     }
00131 
```

```

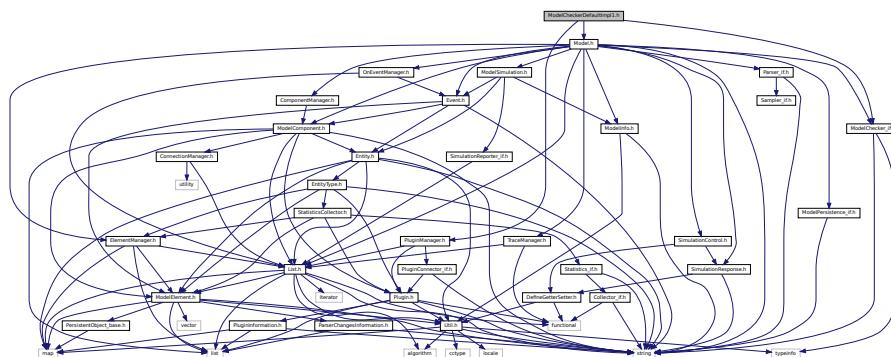
00123         }
00124     }
00125     Util::DecIndent();
00126
00127     // check elements
00128     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Elements:");
00129     Util::IncIndent();
00130     {
00131         std::string elementType;
00132         bool result;
00133         ModelElement* element;
00134         std::string* errorMessage = new std::string();
00135         std::list<std::string>* elementTypes = _model->getElements()->getElementClassnames();
00136         for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end(); typeIt++) {
00137             elementType = (*typeIt);
00138             List<ModelElement*>* elements = _model->getElements()->getElementList(elementType);
00139             for (std::list<ModelElement*>::iterator it = elements->list()->begin(); it != elements->list()->end(); it++) {
00140                 element = (*it);
00141                 // copied from modelComponent. It is not inside the ModelElement::Check because
00142                 ModelElement has no access to Model to call Tracer
00143                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Checking " +
00144                 element->getClassName() + ": \" " + element->getName() + "\\" (id " + std::to_string(element->getId())
00145                 + ")"); //std::to_string(component->_id));
00146                 Util::IncIndent();
00147                 {
00148                     try {
00149                         result = element->Check((*it), errorMessage);
00150                         res &= result;
00151                         if (!result) {
00152                             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error:
00153                             Checking has failed with message '" + *errorMessage + "'");
00154                         }
00155                     } catch (const std::exception& e) {
00156                         _model->getTracer()->traceError(e, "Error verifying component " +
00157                         element->show());
00158                     }
00159                 Util::DecIndent();
00160             }
00161             Util::DecIndent();
00162
00163             return res;
00164     }
00165
00166     bool ModelCheckerDefaultImpl1::checkActivationCode() {
00167     /* \todo: ++: not implemented yet */
00168     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking activation code");
00169     Util::IncIndent();
00170     {
00171     }
00172     Util::DecIndent();
00173     return true;
00174 }
00175 }
00176
00177 bool ModelCheckerDefaultImpl1::checkLimits() {
00178     bool res = true;
00179     std::string text;
00180     unsigned int value, limit;
00181     LicenceManager *licence = _model->getParentSimulator()->getLicenceManager();
00182     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Checking model limits");
00183     Util::IncIndent();
00184     {
00185         value = _model->getComponents()->getNumberOfComponents();
00186         limit = licence->getModelComponentsLimit();
00187         res &= value <= limit;
00188         _model->getTracer()->trace("Model has " + std::to_string(value) + "/" + std::to_string(limit)
00189         + " components");
00190         if (!res) {
00191             text = "Model has " + std::to_string(_model->getComponents()->getNumberOfComponents()) + "
00192             components, exceeding the limit of " + std::to_string(licence->getModelComponentsLimit()) + "
00193             components imposed by the current activation code";
00194             //_model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00195         } else {
00196             value = _model->getElements()->getNumberOfElements();
00197             limit = licence->getModelElementsLimit();
00198             res &= value <= limit;
00199             _model->getTracer()->trace("Model has " + std::to_string(value) + "/" +
00200             std::to_string(limit) + " elements");
00201             if (!res) {
00202                 text = "Model has " + std::to_string(_model->getElements()->getNumberOfElements()) + "

```

```
elements, exceeding the limit of " + std::to_string(licence->getModelElementsLimit()) + " elements
imposed by the current activation code";
00199         //_model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00200     }
00201 }
00202 if (!res) {
00203     _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error: Checking has failed with
message '" + text + "'");
00204 }
00205 }
00206 Util::DecIndent();
00207 return res;
00208 }
```

9.251 ModelCheckerDefaultImpl1.h File Reference

```
#include "ModelChecker_if.h"
#include "Model.h"
#include "PluginManager.h"
Include dependency graph for ModelCheckerDefaultImpl1.h:
```



This graph shows which files directly or indirectly include this file:



Glasses

- class ModelCheckerDefaultImpl

9.252 ModelCheckerDefaultImpl1.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelCheckerDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 18:44
00012 */
00013
00014 #ifndef MODELCHECKERDEFAULTIMPL1_H
```

```

00015 #define MODELCHECKERDEFAULTIMPL1_H
00016
00017 #include "ModelChecker_if.h"
00018 #include "Model.h"
00019 #include "PluginManager.h"
00020
00021 //namespace GenesysKernel {
00022
00023     class ModelCheckerDefaultImpl1 : public ModelChecker_if {
00024     public:
00025         ModelCheckerDefaultImpl1(Model* model);
00026         virtual ~ModelCheckerDefaultImpl1() = default;
00027     public:
00028         virtual bool checkAll();
00029         virtual bool checkConnected();
00030         virtual bool checkSymbols();
00031         virtual bool checkActivationCode();
00032         virtual bool checkLimits();
00033         //virtual bool verifySymbol(std::string componentName, std::string expressionName, std::string
00034         //expression, std::string expressionResult, bool mandatory);
00035     private:
00036         void _recursiveConnectedTo(PluginManager* pluginManager, ModelComponent* comp,
00037             List<ModelComponent*>* visited, List<ModelComponent*>* unconnected, bool* drenoFound);
00038     private:
00039     Model* _model;
00040 };
00041 //namespace\\}
00040 /* MODELCHECKERDEFAULTIMPL1_H */
00041

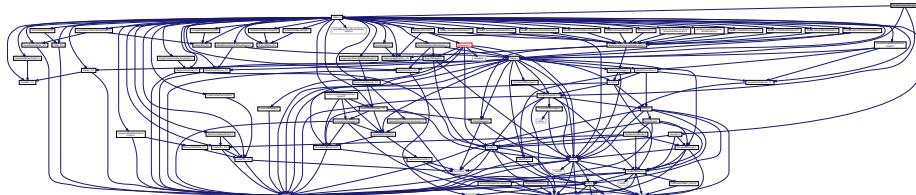
```

9.253 ModelComponent.cpp File Reference

```

#include "ModelComponent.h"
#include "Model.h"
#include "Traits.h"
Include dependency graph for ModelComponent.cpp:

```



9.254 ModelComponent.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Element.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 15:56
00012 */
00013
00014 #include "ModelComponent.h"
00015 #include "Model.h"
00016 #include "Traits.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 ModelComponent::ModelComponent(Model* model, std::string componentTypename, std::string name) :
00021     ModelElement(model, componentTypename, name, false) {
00022     model->getComponents()->insert(this);
00023 }
00024 ModelComponent::~ModelComponent() {
00025     _parentModel->getComponents()->remove(this);

```

```

00026 }
00027
00028 void ModelComponent::Execute(Entity* entity, ModelComponent* component, unsigned int inputNumber) {
00029     std::string msg = "Entity " + std::to_string(entity->entityNumber()) + " has arrived at component
00030     \\" + component->_name + "\";
00031     // \todo: How can I know the number of inputs?
00032     if (inputNumber > 0)
00033         msg += " by input " + std::to_string(inputNumber);
00034     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentArrival, msg);
00035     Util::IncIndent();
00036     try {
00037         component->_execute(entity);
00038     } catch (const std::exception& e) {
00039         component->_parentModel->getTracer()->traceError(e, "Error executing component " +
00040             component->show());
00041     }
00042
00043 void ModelComponent::CreateInternalElements(ModelComponent* component) {
00044     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component
00045     \\" + component->_name + "\"); //std::to_string(component->_id));
00046     try {
00047         component->_createInternalElements();
00048     } catch (const std::exception& e) {
00049         component->_parentModel->getTracer()->traceError(e, "Error creating elements of component " +
00050             component->show());
00051     };
00052 std::map<std::string, std::string>* ModelComponent::SaveInstance(ModelComponent* component) {
00053     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentDetailed, "Writing
00054     component \\" + component->_name + "\"); //std::to_string(component->_id));
00055     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00056     try {
00057         fields = component->_saveInstance();
00058     } catch (const std::exception& e) {
00059         component->_parentModel->getTracer()->traceError(e, "Error executing component " +
00060             component->show());
00061     }
00062
00063 bool ModelComponent::Check(ModelComponent* component) {
00064     component->_parentModel->getTracer()->trace(Util::TraceLevel::componentDetailed, "Checking " +
00065         component->_typename + " : \\" + component->_name + "\"); //std::to_string(component->_id));
00066     bool res = false;
00067     std::string* errorMessage = new std::string();
00068     Util::IncIndent();
00069     {
00070         try {
00071             res = component->_check(errorMessage);
00072             if (!res) {
00073                 component->_parentModel->getTracer()->trace(Util::TraceLevel::errorFatal, "Error:
00074                 Checking has failed with message '" + *errorMessage + "'");
00075             }
00076         } catch (const std::exception& e) {
00077             component->_parentModel->getTracer()->traceError(e, "Error verifying component " +
00078                 component->show());
00079         }
00080     }
00081     Util::DecIndent();
00082     return res;
00083 }
00084
00085 ConnectionManager* ModelComponent::getNextComponents() const {
00086     return _connections;
00087 }
00088
00089 std::string ModelComponent::show() {
00090     return ModelElement::show(); // "(id=" + std::to_string(this->_id) + ",name=\\""+this->_name +
00091     "\")"; // , nextComponents[]=( " + _nextComponents->show() + ")");
00092 }
00093
00094 bool ModelComponent::_loadInstance(std::map<std::string, std::string>* fields) {
00095     bool res = ModelElement::_loadInstance(fields);
00096     if (res) {
00097         // Now it should load nextComponents. The problem is that the nextComponent may not be loaded
00098         yet.
00099         // So, what can be done is to temporarily load the ID of the nextComponents, and to wait until
00100         all the components have been loaded to update nextComponents based on the temporarilyIDs now being
00101         loaded
00102         //unsigned short nextSize = std::stoi((*fields->find("nextSize")).second);
00103         //this->_tempLoadNextComponentsIDs = new List<Util::identification>();
00104         //for (unsigned short i = 0; i < nextSize; i++) {
00105         //    Util::identification nextId = std::stoi((*fields->find("nextId" +
00106             std::to_string(i))).second);

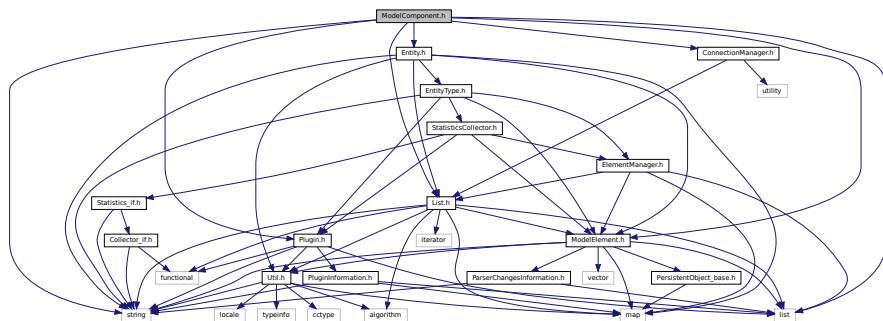
```

```

00099         //      this->_tempLoadNextComponentsIDs->insert(nextId);
00100     //}
00101 }
00102 return res;
00103 }
00104
00105 std::map<std::string, std::string>* ModelComponent::_saveInstance() {
00106     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00107     if (true) {//(_connections->size() != 1) { // save nextSize only if it is != 1
00108         fields->emplace("nextSize", std::to_string(_connections->size()));
00109     }
00110     unsigned short i = 0;
00111     for (std::list<Connection*>::iterator it = _connections->list()->begin(); it != _connections->list()->end(); it++) {
00112         fields->emplace("nextId" + std::to_string((*it)->first->_id));
00113         if (true) {//((*it)->second != 0) { // save nextInputNumber only if it is != 0
00114             fields->emplace("nextInputNumber" + std::to_string(i), std::to_string((*it)->second));
00115         }
00116         i++;
00117     }
00118     return fields;
00119 }
00120
00121 void ModelComponent::_createInternalElements() {
00122
00123 }
```

9.255 ModelComponent.h File Reference

```
#include <string>
#include <list>
#include "Plugin.h"
#include "List.h"
#include "Entity.h"
#include "ModelElement.h"
#include "ConnectionManager.h"
Include dependency graph for ModelComponent.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class ModelComponent

9.256 ModelComponent.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Element.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 15:56
00012 */
00013
00014 #ifndef MODELCOMPONENT_H
00015 #define MODELCOMPONENT_H
00016
00017 #include <string>
00018 #include <list>
00019
00020 #include "Plugin.h"
00021 #include "List.h"
00022 #include "Entity.h"
00023 #include "ModelElement.h"
00024 #include "ConnectionManager.h"
00025 //namespace GenesysKernel {
00026
00027     class Model;
00028
00029     class ModelComponent : public ModelElement {
00030     public:
00031         ModelComponent(Model* model, std::string componentTypename, std::string name = "");
00032         virtual ~ModelComponent();
00033     public:
00034         virtual std::string show();
00035     public:
00036         ConnectionManager* getNextComponents() const;
00037     public: // static
00038         static void Execute(Entity* entity, ModelComponent* component, unsigned int inputNumber);
00039         //static void InitBetweenReplications(ModelComponent* component);
00040         static void CreateInternalElements(ModelComponent* component);
00041         static bool Check(ModelComponent* component);
00042         static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00043         static std::map<std::string, std::string>* SaveInstance(ModelComponent* component);
00044     protected: // pure virtual methods
00045         virtual void _execute(Entity* entity) = 0;
00046     protected: // virtual methods
00047         virtual std::map<std::string, std::string>* _saveInstance();
00048         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00049         virtual void _createInternalElements();
00050     protected:
00051         ConnectionManager* _connections = new ConnectionManager();
00052     };
00053 //namespace\\}
00054 #endif /* MODELCOMPONENT_H */
00055

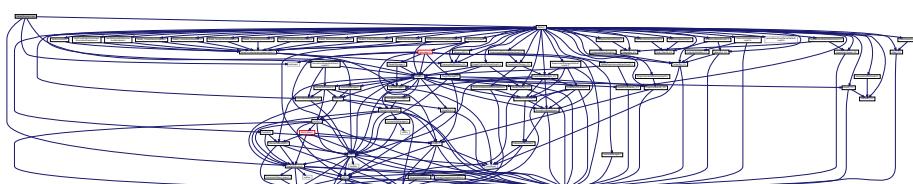
```

9.257 ModelElement.cpp File Reference

```

#include <iostream>
#include "ModelElement.h"
#include "Model.h"
#include "Traits.h"
Include dependency graph for ModelElement.cpp:

```



9.258 ModelElement.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelElement.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 19:40
00012 */
00013
00014 //#include <typeinfo>
00015 #include <iostream>
00016 #include "ModelElement.h"
00017 #include "Model.h"
00018 #include "Traits.h"
00019
00020 //using namespace GenesysKernel;
00021
00022 ModelElement::ModelElement(Model* model, std::string thistypename, std::string name, bool
00023     insertIntoModel) {
00024     _id = Util::GenerateNewId(); //GenerateNewIdOfType(thistypename);
00025     _typename = thistypename;
00026     _parentModel = model;
00027     _reportStatistics = Traits<ModelElement>::reportStatistics; // \todo: shouold be a parameter
00028     before insertIntoModel
00029         if (name == "")
00030             _name = thistypename + "_" + std::to_string(Util::GenerateNewIdOfType(thistypename));
00031         else
00032             _name = name;
00033         if (insertIntoModel)
00034             model->insert(this);
00035     }
00036 //ModelElement::ModelElement(const ModelElement &orig) {
00037 //this->_parentModel = orig->_parentModel;
00038 //this->_name = "copy_of_" + orig->_name;
00039 //this->_typename = orig->_typename;
00040 //}
00041
00042 ModelElement::~ModelElement() {
00043     _parentModel->getTracer()->trace(Util::TraceLevel::everythingMostDetailed, "Removing Element \""
00044         + this->_name + "\" from the model");
00045     _removeChildrenElements();
00046     _parentModel->getElements()->remove(this);
00047 }
00048 void ModelElement::_removeChildrenElements() {
00049     Util::IncIndent();
00050     {
00051         for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it !=
00052             _childrenElements->end(); it++) {
00053             this->_parentModel->getElements()->remove((*it).second);
00054             (*it).second->~ModelElement();
00055         }
00056         _childrenElements->clear();
00057     }
00058     Util::DecIndent();
00059 }
00060
00061 void ModelElement::_removeChildElement(std::string key) {
00062     Util::IncIndent();
00063     {
00064         //for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it !=
00065         _childrenElements->end(); it++) {
00066             std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin();
00067             while (it != _childrenElements->end()) {
00068                 if ((*it).first == key) {
00069                     this->_parentModel->getElements()->remove((*it).second);
00070                     (*it).second->~ModelElement();
00071                     _childrenElements->erase(it);
00072                 }
00073             }
00074         }
00075     Util::DecIndent();
00076 }
00077
00078 bool ModelElement::_loadInstance(std::map<std::string, std::string>* fields) {
00079     bool res = true;
00080     std::map<std::string, std::string>::iterator it;

```

```

00081     it = fields->find("id");
00082     if != fields->end() ? this->_id = std::stoi((*it).second) : res = false;
00083     it = fields->find("name");
00084     if != fields->end() ? this->_name = (*it).second : this->_name = "";
00085     //if != fields->end() ? this->_name = (*it).second : res = false;
00086     it = fields->find("reportStatistics");
00087     if != fields->end() ? this->_reportStatistics = std::stoi((*it).second) : this->_reportStatistics
00088     = Traits<ModelElement>::reportStatistics;
00089     return res;
00090 }
00091 std::map<std::string, std::string>* ModelElement::_saveInstance() {
00092     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00093     fields->emplace("typename", this->_typename);
00094     fields->emplace("id", std::to_string(this->_id));
00095     fields->emplace("name", "\" + this->_name + "\"");
00096     if (this->_reportStatistics != Traits<ModelElement>::reportStatistics)
00097         fields->emplace("reportStatistics", std::to_string(this->_reportStatistics));
00098     return fields;
00099 }
00100 bool ModelElement::_check(std::string* errorMessage) {
00101     return true; // if there is no override, return true
00102 }
00103
00104 ParserChangesInformation* ModelElement::_getParserChangesInformation() {
00105     return new ParserChangesInformation(); // if there is no override, return no changes
00106 }
00107
00108 void ModelElement::_initBetweenReplications() {
00109 }
00110 }
00111 */
00112 std::list<std::map<std::string, std::string>*>* ModelElement::_saveInstance(std::string type) {
00113     std::list<std::map<std::string, std::string>*>* fields = ModelElement::_saveInstance();
00114     fields->push_back(type);
00115     return fields;
00116 }
00117 */
00118 */
00119
00120 std::string ModelElement::show() {
00121     std::string children = "";
00122     if (_childrenElements->size() > 0) {
00123         children = ", children=[";
00124         for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it != _childrenElements->end(); it++) {
00125             children += (*it).second->getName() + ",";
00126         }
00127         children = children.substr(0, children.length() - 1) + "]";
00128     }
00129     return "id=" + std::to_string(_id) + ",name=\"" + _name + "\" + children;
00130 }
00131
00132 std::list<std::string>* ModelElement::getChildrenElementKeys() const {
00133     std::list<std::string>* result = new std::list<std::string>();
00134     return result;
00135 }
00136
00137 Util::identification ModelElement::getId() const {
00138     return _id;
00139 }
00140
00141 void ModelElement::setName(std::string _name) {
00142     this->_name = _name;
00143 }
00144
00145 std::string ModelElement::getName() const {
00146     return _name;
00147 }
00148
00149 std::string ModelElement::getClassname() const {
00150     return _typename;
00151 }
00152
00153 void ModelElement::InitBetweenReplications(ModelElement* element) {
00154     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component
00155     //\" + component->_name + "\", //std::to_string(component->_id));
00156     try {
00157         element->_initBetweenReplications();
00158     } catch (const std::exception& e) {
00159         element->_parentModel->getTracer()->traceError(e, "Error initing component " +
00160             element->show());
00161     };
00162 }
00162 ModelElement* ModelElement::LoadInstance(Model* model, std::map<std::string, std::string>* fields,

```

```

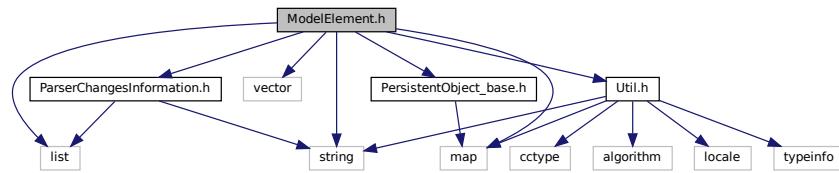
00163     bool insertIntoModel) {
00164         std::string name = "";
00165         if (insertIntoModel) {
00166             // extracts the name from the fields even before "_laodInstance" and even before construct a
00167             // new ModelElement in such way when constructing the ModelElement, it's done with the correct name and
00168             // that correct name is show in trace
00169             std::map<std::string, std::string>::iterator it = fields->find("name");
00170             if (it != fields->end())
00171                 name = (*it).second;
00172         }
00173         ModelElement* newElement = new ModelElement(model, "ModelElement", name, insertIntoModel);
00174         try {
00175             newElement->_loadInstance(fields);
00176         } catch (const std::exception& e) {
00177         }
00178         return newElement;
00179     std::map<std::string, std::string>* ModelElement::SaveInstance(ModelElement* element) {
00180         std::map<std::string, std::string>* fields; // = new std::list<std::string>();
00181         try {
00182             fields = element->_saveInstance();
00183         } catch (const std::exception& e) {
00184             //element->_model->getTrace()->traceError(e, "Error saving anElement " + element->show());
00185         }
00186         return fields;
00187     }
00188
00189     bool ModelElement::Check(ModelElement* element, std::string* errorMessage) {
00190         // element->_model->getTraceManager()->trace(Util::TraceLevel::mostDetailed, "Checking " +
00191         element->_typename + ": " + element->_name); //std::to_string(element->_id));
00192         bool res = false;
00193         Util::IncIndent();
00194         {
00195             try {
00196                 res = element->_check(errorMessage);
00197                 // element->_model->getTraceManager()->trace(Util::TraceLevel::errors,
00198                 "Error: Checking has failed with message '" + *errorMessage + "'");
00199             } catch (const std::exception& e) {
00200                 // element->_model->getTraceManager()->traceError(e, "Error verifying element "
00201                 + element->show());
00202             }
00203             Util::DecIndent();
00204         }
00205         return res;
00206     }
00207     void ModelElement::CreateInternalElements(ModelElement* element) {
00208         try {
00209             Util::IncIndent();
00210             element->_createInternalElements();
00211             Util::DecIndent();
00212         } catch (const std::exception& e) {
00213             //element->...->_model->getTraceManager()->traceError(e, "Error creating elements of element "
00214             + element->show());
00215         }
00216
00217     void ModelElement::_createInternalElements() {
00218
00219     }
00220
00221     void ModelElement::setReportStatistics(bool reportStatistics) {
00222         this->_reportStatistics = reportStatistics;
00223     }
00224
00225     bool ModelElement::isReportStatistics() const {
00226         return _reportStatistics;
00227     }

```

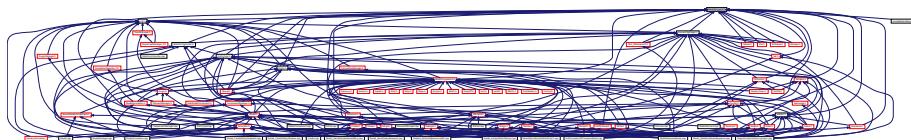
9.259 ModelElement.h File Reference

```
#include <string>
#include <list>
#include <vector>
#include <map>
#include "Util.h"
```

```
#include "ParserChangesInformation.h"
#include "PersistentObject_base.h"
Include dependency graph for ModelElement.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ModelElement](#)

9.260 ModelElement.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelElement.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:40
00012 */
00013
00014 #ifndef MODELEMENT_H
00015 #define MODELEMENT_H
00016
00017 #include <string>
00018 #include <list>
00019 #include <vector>
00020 #include <map>
00021 #include "Util.h"
00022
00023 #include "ParserChangesInformation.h"
00024 #include "PersistentObject_base.h"
00025
00026 //namespace GenesysKernel {
00027     class Model;
00028
00033     class ModelElement: PersistentObject_base {
00034     public:
00035         ModelElement(Model* model, std::string elementTypename, std::string name = "", bool
00036             insertIntoModel = true);
00037         //ModelElement(Model* model, std::string elementTypename, std::string name = "", bool
00038         //insertIntoModel = true);
00038         //ModelElement(const ModelElement &orig);
00039         virtual ~ModelElement();
00040
00041     public: // get & set
00041         Util::identification getId() const;
  
```

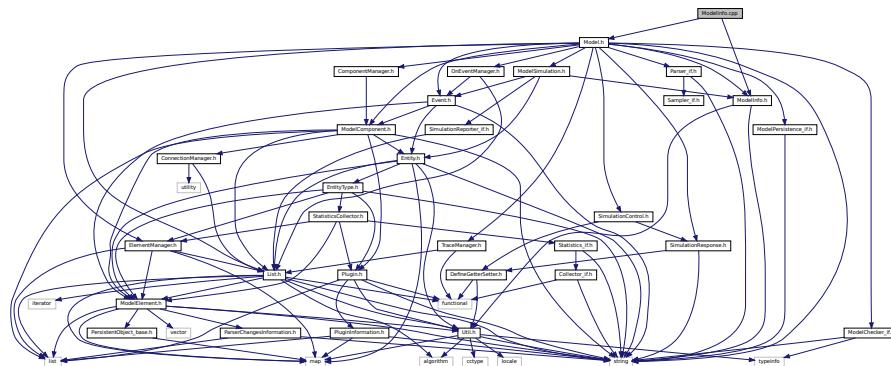
```

00042     void setName(std::string _name);
00043     std::string getName() const;
00044     std::string getclassname() const;
00045     bool isReportStatistics() const;
00046     void setReportStatistics(bool reportStatistics);
00047 public: // static
00048     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields,
00049                                         bool insertIntoModel); // \todo: return ModelComponent* ?
00050     static std::map<std::string, std::string>* SaveInstance(ModelElement* element);
00051     static bool Check(ModelElement* element, std::string* errorMessage);
00052     static void CreateInternalElements(ModelElement* element);
00053     static void InitBetweenReplications(ModelElement* element);
00054 public:
00055     virtual std::string show();
00056     std::list<std::string>* getChildElementKeys() const;
00057     ModelElement* getChildElement(std::string key) const;
00058 protected:
00059     void _setChildElement(std::string key, ModelElement* child);
00060     void _removeChildrenElements();
00061     void _removeChildElement(std::string key);
00062 protected: // must be overridden by derived classes
00063     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00064     virtual std::map<std::string, std::string>* _saveInstance();
00065 protected: // could be overridden by derived classes
00066     virtual bool _check(std::string* errorMessage);
00067     virtual ParserChangesInformation* _getParserChangesInformation();
00068     virtual void _initBetweenReplications();
00069     virtual void _createInternalElements();
00070 private:
00071     void _build(Model* model, std::string thistypename, bool insertIntoModel);
00072 protected:
00073     Util::identification _id;
00074     std::string _name;
00075     std::string _typename;
00076     bool _reportStatistics;
00077     Model* _parentModel;
00078 protected:
00079     std::map<std::string, ModelElement*>* _childrenElements = new std::map<std::string,
00080                                         ModelElement*>();
00081 };
00082 //namespace\\}
00083
00084 #endif /* MODELELEMENT_H */
00085

```

9.261 ModellInfo.cpp File Reference

```
#include "ModellInfo.h"
#include "Model.h"
Include dependency graph for ModellInfo.cpp:
```



9.262 ModellInfo.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates

```

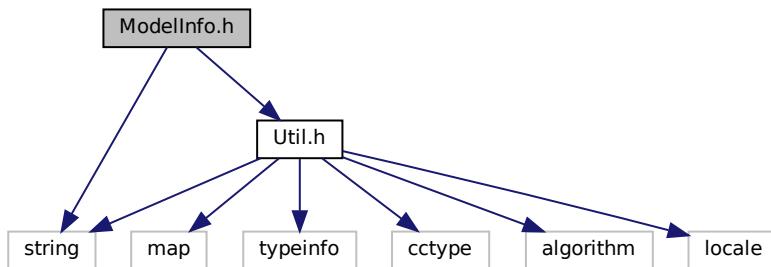
```
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelInfo.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 18:13
00012 */
00013
00014 #include "ModelInfo.h"
00015 #include "Model.h"
00016
00017 //using namespace GenesysKernel;
00018
00019 ModelInfo::ModelInfo() {
00020     _name = "Model " + std::to_string(Util::GenerateNewIdOfType<Model>());
00021 }
00022
00023 std::string ModelInfo::show() {
00024     return "analystName=\"" + this->_analystName + "\"" +
00025         ",description=\"" + this->_description + "\"" +
00026         ",name=\"" + this->_name + "\"" +
00027         ",version=" + this->_version;
00028 }
00029
00030 void ModelInfo::setName(std::string _name) {
00031     this->_name = _name;
00032     _hasChanged = true;
00033 }
00034
00035 std::string ModelInfo::getName() const {
00036     return _name;
00037 }
00038
00039 void ModelInfo::setAnalystName(std::string _analystName) {
00040     this->_analystName = _analystName;
00041     _hasChanged = true;
00042 }
00043
00044 std::string ModelInfo::getAnalystName() const {
00045     return _analystName;
00046 }
00047
00048 void ModelInfo::setDescription(std::string _description) {
00049     this->_description = _description;
00050     _hasChanged = true;
00051 }
00052
00053 std::string ModelInfo::getDescription() const {
00054     return _description;
00055 }
00056
00057 void ModelInfo::setTitle(std::string _projectTitle) {
00058     this->_projectTitle = _projectTitle;
00059     _hasChanged = true;
00060 }
00061
00062 std::string ModelInfo::getTitle() const {
00063     return _projectTitle;
00064 }
00065
00066 void ModelInfo::setVersion(std::string _version) {
00067     this->_version = _version;
00068     _hasChanged = true;
00069 }
00070
00071 std::string ModelInfo::getVersion() const {
00072     return _version;
00073 }
00074
00075 void ModelInfo::loadInstance(std::map<std::string, std::string>* fields) {
00076     this->_analystName = loadField(fields, "analystName", "");
00077     this->_description = loadField(fields, "description", "");
00078     this->_name = loadField(fields, "name", "");
00079     this->_projectTitle = loadField(fields, "projectTitle", "");
00080     this->_version = loadField(fields, "version", "1.0");
00081     _hasChanged = false;
00082 }
00083
00084 // \todo!: implement check method (to check things like terminating condition)
00085
00086 std::map<std::string, std::string>* ModelInfo::saveInstance() {
00087     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00088     fields->emplace("typename", "ModelInfo");
00089     if (_analystName != "") fields->emplace("analystName", "\\" + _analystName + "\\");
00090     if (_description != "") fields->emplace("description", "\\" + _description + "\\");
00091 }
```

```

00091     fields->emplace("name", "\\" + getName() + "\\");
00092     if (_projectTitle != "") fields->emplace("projectTitle", "\\" + _projectTitle + "\\");
00093     if (_version != "1.0") fields->emplace("version", "\\" + _version + "\\");
00094     _hasChanged = false;
00095     return fields;
00096 }
00097
00098 bool ModelInfo::hasChanged() const {
00099     return _hasChanged;
00100 }
```

9.263 ModelInfo.h File Reference

```
#include <string>
#include "Util.h"
Include dependency graph for ModelInfo.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ModelInfo](#)

9.264 ModelInfo.h

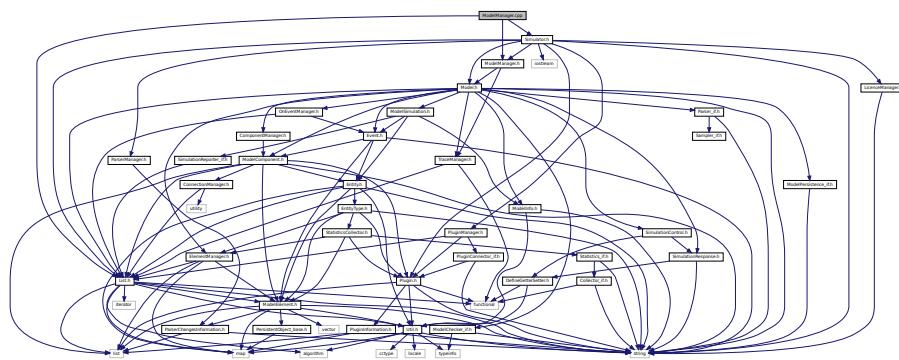
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelInfo.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 18:13
00012 */
00013
00014 #ifndef MODELINFO_H
00015 #define MODELINFO_H
00016
00017 #include <string>
00018 #include "Util.h"
```

```
00019
00020 //namespace GenesysKernel {
00021
00025     class ModelInfo {
00026     public:
00027         ModelInfo();
00028         virtual ~ModelInfo() = default;
00029     public:
00030         std::string show();
00031     public: // gets and sets
00032         void setName(std::string _name);
00033         std::string getName() const;
00034         void setAnalystName(std::string _analystName);
00035         std::string getAnalystName() const;
00036         void setDescription(std::string _description);
00037         std::string getDescription() const;
00038         void setProjectTitle(std::string _projectTitle);
00039         std::string getProjectTitle() const;
00040         void setVersion(std::string _version);
00041         std::string getVersion() const;
00042     public:
00043         void loadInstance(std::map<std::string, std::string>* fields);
00044         std::map<std::string, std::string>* saveInstance();
00045         bool hasChanged() const;
00046     private: // with public access (get & set)
00047         // model general information
00048         std::string _name;
00049         std::string _analystName = "";
00050         std::string _description = "";
00051         std::string _projectTitle = "";
00052         std::string _version = "1.0";
00053         bool _hasChanged = false;
00054     };
00055 //namespace\\\
00056 #endif /* MODELINFO_H */
```

9.265 ModelManager.cpp File Reference

```
#include "ModelManager.h"
#include "List.h"
#include "Simulator.h"
Include dependency graph for ModelManager.cpp:
```



9.266 ModelManager.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: ModelManager.cpp  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 31 de Maio de 2019, 08:37
```

```

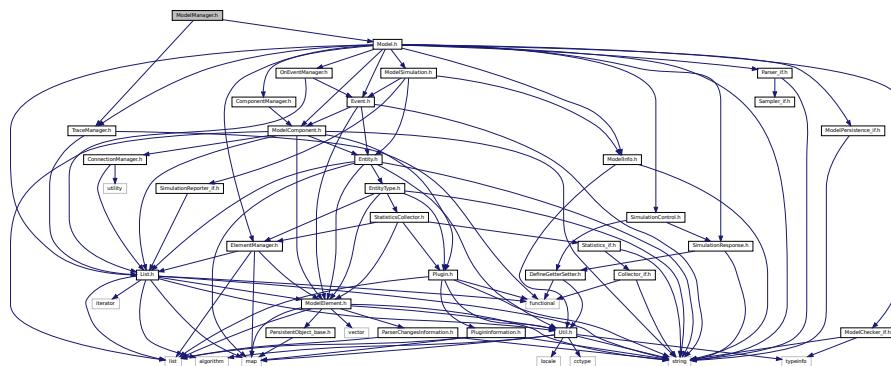
00012  */
00013
00014 #include "ModelManager.h"
00015 #include "List.h"
00016 #include "Simulator.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 ModelManager::ModelManager(Simulator* simulator) {
00021     _simulator = simulator;
00022     _currentModel = nullptr;
00023 }
00024
00025 Model* ModelManager::newModel() {
00026     _currentModel = new Model(_simulator);
00027     return _currentModel;
00028 }
00029
00030 void ModelManager::insert(Model* model) {
00031     _models->insert(model);
00032     this->_currentModel = model;
00033     _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model successfully inserted");
00034 }
00035
00036 void ModelManager::remove(Model* model) {
00037     _models->remove(model);
00038     if (_currentModel == model) {
00039         _currentModel = this->front();
00040     }
00041     model->~Model();
00042     _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model successfully removed");
00043 }
00044
00045 unsigned int ModelManager::size() {
00046     return _models->size();
00047 }
00048
00049 bool ModelManager::saveModel(std::string filename) {
00050     if (_currentModel != nullptr)
00051         return _currentModel->save(filename);
00052     return false;
00053 }
00054
00055 bool ModelManager::loadModel(std::string filename) {
00056     Model* model = new Model(_simulator);
00057     bool res = model->load(filename);
00058     if (res) {
00059         this->insert(model);
00060         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model successfully loaded");
00061     } else {
00062         model->~Model();
00063         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Model could not be loaded");
00064     }
00065     return res;
00066 }
00067
00068 void ModelManager::setCurrent(Model* model) {
00069     this->_currentModel = model;
00070 }
00071
00072 Model* ModelManager::current() {
00073     return _currentModel;
00074 }
00075
00076 Model* ModelManager::front() {
00077     return _models->front();
00078 }
00079
00080 Model* ModelManager::next() {
00081     return _models->next();
00082 }

```

9.267 ModelManager.h File Reference

```
#include "Model.h"
#include "TraceManager.h"
```

Include dependency graph for ModelManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ModelManager](#)

9.268 ModelManager.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 08:37
00012 */
00013
00014 #ifndef MODELMANAGER_H
00015 #define MODELMANAGER_H
00016
00017 #include "Model.h"
00018 #include "TraceManager.h"
00019
00020 //namespace GenesysKernel {
00021
00022     class ModelManager {
00023     public:
00024         ModelManager(Simulator* simulator);
00025         virtual ~ModelManager() = default;
00026     public:
00027         Model* newModel();
00028         void insert(Model* model);
00029         void remove(Model* model);
00030         void setCurrent(Model* model);
00031         bool saveModel(std::string filename);
00032         bool loadModel(std::string filename);
00033         unsigned int size();
00034     public:
00035         Model* front();
00036         Model* current();
00037         Model* next();
00038         //Model* end();
00039     private:
00040         List<Model*>* _models = new List<Model*>();

```

```

00041     Model* _currentModel;
00042     private:
00043     Simulator* _simulator;
00044 };
00045 //namespace\\}
00046 #endif /* MODELMANAGER_H */
00047

```

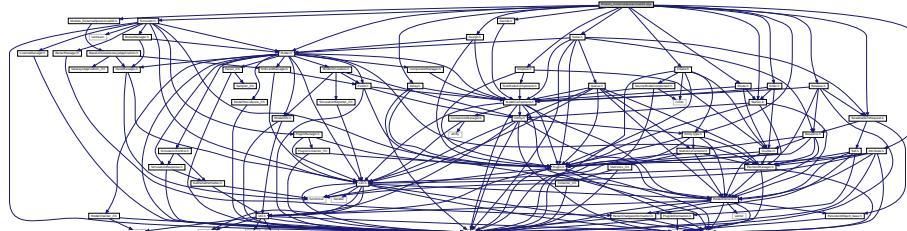
9.269 Modelo_SistemaOperacional02.cpp File Reference

```

#include "Modelo_SistemaOperacional02.h"
#include "Simulator.h"
#include "Model.h"
#include "Create.h"
#include "Assign.h"
#include "Seize.h"
#include "Route.h"
#include "Enter.h"
#include "Decide.h"
#include "Delay.h"
#include "Release.h"
#include "Dispose.h"
#include "EntityType.h"
#include "SeizableItemRequest.h"
#include "Station.h"
#include "Attribute.h"

```

Include dependency graph for Modelo_SistemaOperacional02.cpp:



9.270 Modelo_SistemaOperacional02.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Modelo_SistemasOperacionais02.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 12 de abril de 2021, 21:54
00012 */
00013
00014 #include "Modelo_SistemaOperacional02.h"
00015
00016 #include "Simulator.h"
00017 #include "Model.h"
00018 #include "Create.h"
00019 #include "Assign.h"
00020 #include "Seize.h"
00021 #include "Route.h"
00022 #include "Enter.h"
00023 #include "Decide.h"
00024 #include "Delay.h"
00025 #include "Release.h"
00026 #include "Dispose.h"

```

```

00027
00028 #include "EntityType.h"
00029 #include "SeizableItemRequest.h"
00030 #include "Station.h"
00031 #include "Attribute.h"
00032
00033 Modelo_SistemaOperacional02::Modelo_SistemaOperacional02() {
00034 }
00035
00036 int Modelo_SistemaOperacional02::main(int argc, char** argv) {
00037     Simulator* genesys = new Simulator();
00038     this->setDefaultTraceHandlers(genesys->getTracer());
00039     this->insertFakePluginsByHand(genesys);
00040     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00041
00042     Model* m = genesys->getModels()->newModel();
00043     m->load("./models/SistemaOperacional02.txt");
00044     m->getSimulation()->start();
00045     return;
00046     EntityType* et = new EntityType(m, "processo");
00047     Create* cl = new Create(m, "Processo é criado no computador");
00048     cl->setEntityType(et);
00049     cl-> setTimeBetweenCreationsExpression("expo(10)");
00050     cl-> setTimeUnit(Util::TimeUnit::millisecond);
00051     Assign* a1 = new Assign(m, "Define tempo de execução e memória ocupada");
00052     a1->getAssignments()->insert(new Assign::Assignment("memoriaOcupada", "TRUNC(UNIF(10,50))"));
00053     a1->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "NORM(3,1) * memoriaOcupada/10"));
00054     Attribute* att1 = new Attribute(m, "memoriaOcupada");
00055     Attribute* att2 = new Attribute(m, "tempoExecucao");
00056     cl->getNextComponents()->insert(a1);
00057     Resource* mem = new Resource(m, "memoria");
00058     mem->setCapacity(64);
00059     Queue* q1 = new Queue(m, "Fila_Alocacao_Memoria");
00060     Seize* sz1 = new Seize(m, "Processo aloca memória");
00061     a1->getNextComponents()->insert(sz1);
00062     sz1->setQueue(q1);
00063     sz1->getSeizeRequests()->insert(new SeizableItemRequest(mem, "memoriaOcupada"));
00064     Station* st1 = new Station(m, "Estação de Execução");
00065     Route* r1 = new Route(m, "Processo é enviado para execução na CPU");
00066     r1->setStation(st1);
00067     sz1->getNextComponents()->insert(r1);
00068
00069     Enter* el = new Enter(m, "Processo chega para ser executado");
00070     el->setStation(st1);
00071     Resource* cpu = new Resource(m, "CPU");
00072     Queue* q2 = new Queue(m, "Fila_CPU");
00073     Seize* sz2 = new Seize(m, "Processo aloca CPU");
00074     sz2->setQueue(q2);
00075     sz2->getSeizeRequests()->insert(new SeizableItemRequest(cpu, "1"));
00076     el->getNextComponents()->insert(sz2);
00077     Decide* decl = new Decide(m, "Define tempo de execução da fatia de tempo atual");
00078     decl->getConditions()->insert("tempoExecucao >= 1");
00079     sz2->getNextComponents()->insert(decl);
00080     Assign* a2 = new Assign(m, "Define execução por um quantum de tempo");
00081     a2->getAssignments()->insert(new Assign::Assignment("fatiaTempo", "1"));
00082     a2->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "tempoExecucao-fatiaTempo"));
00083     Attribute* att3 = new Attribute(m, "fatiaTempo");
00084     decl->getNextComponents()->insert(a2);
00085     Assign* a3 = new Assign(m, "Executa até o final");
00086     a3->getAssignments()->insert(new Assign::Assignment("fatiaTempo", "tempoExecucao"));
00087     a3->getAssignments()->insert(new Assign::Assignment("tempoExecucao", "tempoExecucao-fatiaTempo"));
00088     decl->getNextComponents()->insert(a3);
00089     Delay* dll = new Delay(m, "Processo executa na CPU");
00090     dll->setDelayExpression("fatiaTempo");
00091     dll->setDelayTimeUnit(Util::TimeUnit::millisecond);
00092     a2->getNextComponents()->insert(dll);
00093     a3->getNextComponents()->insert(dll);
00094     Release* r11 = new Release(m, "Processo libera CPU");
00095     r11->getReleaseRequests()->insert(new SeizableItemRequest(cpu, "1"));
00096     dll->getNextComponents()->insert(r11);
00097     Decide* dc2 = new Decide(m, "Se processo ainda precisa executar então vai para estação de execução");
00098     dc2->getConditions()->insert("tempoExecucao > 0");
00099     r11->getNextComponents()->insert(dc2);
00100     Route* r2 = new Route(m, "Processo é enviado de volta para execução");
00101     r2->setStation(st1);
00102     dc2->getNextComponents()->insert(r2);
00103     Station* st2 = new Station(m, "Estação de liberação de memória");
00104     Route* r3 = new Route(m, "Processo é enviado para liberar memória");
00105     r3->setStation(st2);
00106     dc2->getNextComponents()->insert(r3);
00107
00108     Enter* e2 = new Enter(m, "Processo chega para liberar memória");
00109     e2->setStation(st2);
00110     Release* r12 = new Release(m, "Processo libera memória");
00111     r12->getReleaseRequests()->insert(new SeizableItemRequest(mem, "memoriaOcupada"));

```

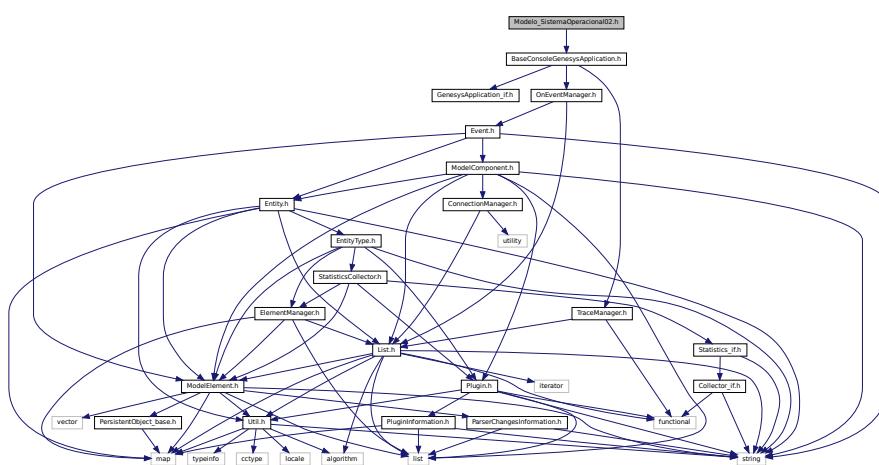
```

00112     e2->getNextComponents()->insert(r12);
00113     Dispose* disp1 = new Dispose(m, "Processo é encerrado");
00114     r12->getNextComponents()->insert(disp1);
00115
00116     ModelSimulation* sim = m->getSimulation();
00117     sim->setReplicationLength(1);
00118     sim->setReplicationLengthTimeUnit(Util::TimeUnit::second);
00119     genesys->getTracer()->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00120     m->save("./models/SistemaOperacional02.txt");
00121     sim->start();
00122 }

```

9.271 Modelo_SistemaOperacional02.h File Reference

#include "BaseConsoleGenesysApplication.h"
Include dependency graph for Modelo_SistemaOperacional02.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Modelo_SistemaOperacional02](#)

9.272 Modelo_SistemaOperacional02.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Modelo_SistemasOperacionais02.h
00009  * Author: rlcancian
00010  *
00011  * Created on 12 de abril de 2021, 21:54
00012 */
00013 #ifndef MODELO_SISTEMASOPERACIONAIS02_H

```

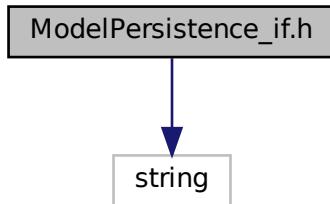
```

00015 #define MODELO_SISTEMASOPERACIONAIS02_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class Modelo_SistemaOperacional02 :public BaseConsoleGenesysApplication {
00020 public:
00021     Modelo_SistemaOperacional02();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* MODELO_SISTEMASOPERACIONAIS02_H */
00027

```

9.273 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](#)

9.274 ModelPersistence_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    ModelPersistence_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 24 de Agosto de 2018, 19:22
00012 */
00013
00014 #ifndef MODELERSISTENCE_IF_H
00015 #define MODELERSISTENCE_IF_H
00016
00017 #include <string>

```

```

00018
00019 class ModelPersistence_if {
00020 public:
00021     // \todo: not a good interface for sure. The Bridge pattern should be a lot better
00022     virtual bool save(std::string filename) = 0;
00023     virtual bool load(std::string filename) = 0;
00024     virtual bool hasChanged() = 0;
00025 private:
00026 };
00027
00028 #endif /* MODEL PERSISTENCE IF H */
00029
00030
00031
00032
00033

```

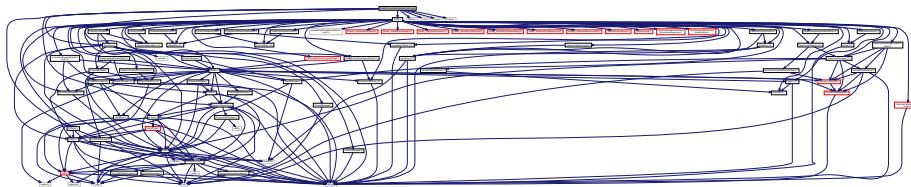
9.275 ModelPersistenceDefaultImpl1.cpp File Reference

```

#include "ModelPersistenceDefaultImpl1.h"
#include <fstream>
#include <ctime>
#include <regex>
#include <cassert>
#include "ModelComponent.h"
#include "Simulator.h"
#include "Traits.h"
#include "Counter.h"

```

Include dependency graph for ModelPersistenceDefaultImpl1.cpp:



9.276 ModelPersistenceDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelPersistenceDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 20 de Maio de 2019, 20:41
00012 */
00013
00014 #include "ModelPersistenceDefaultImpl1.h"
00015 #include <fstream>
00016 #include <ctime>
00017 #include <regex>
00018 #include <cassert>
00019 #include "ModelComponent.h"
00020 #include "Simulator.h"
00021 #include "Traits.h"
00022 #include "Counter.h"
00023
00024 //using namespace GenesysKernel;
00025
00026 ModelPersistenceDefaultImpl1::ModelPersistenceDefaultImpl1(Model* model) {
00027     _model = model;
00028 }
00029
00030 std::map<std::string, std::string>* ModelPersistenceDefaultImpl1::_getSimulatorInfoFieldsToSave() {
00031     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00032     fields->emplace("typename", "SimulatorInfo");
00033     fields->emplace("name", "\"" + _model->getParentSimulator()->getName() + "\"");

```

```

00034     fields->emplace("versionNumber",
00035         std::to_string(_model->getParentSimulator()->getVersionNumber()));
00036     fields->emplace("version", "\"" + _model->getParentSimulator()->getVersion() + "\"");
00037     return fields;
00038 }
00039 bool ModelPersistenceDefaultImpl1::save(std::string filename) {
00040     _model->getTracer()->trace(Util::TraceLevel::toolInternal, "Saving file \"\" + filename + \"\"");
00041     Util::IncIndent();
00042     std::list<std::string> *simulatorInfosToSave, *simulationInfosToSave, *modelInfosToSave,
00043     *modelElementsToSave, *modelComponentsToSave;
00044     {
00045         //bool res = true;
00046         std::map<std::string, std::string>* fields;
00047         fields = _getSimulatorInfoFieldsToSave();
00048         simulatorInfosToSave = _adjustFieldsToSave(fields);
00049         // save model own infos
00050         fields = _model->getInfos()->saveInstance();
00051         modelInfosToSave = _adjustFieldsToSave(fields);
00052         // save model own infos
00053         // \todo save modelSimulation fields (breakpoints)
00054         fields = _model->getSimulation()->saveInstance();
00055         simulationInfosToSave = _adjustFieldsToSave(fields);
00056         // save infras
00057         modelElementsToSave = new std::list<std::string>();
00058         std::list<std::string>* elementTypenames = _model->getElements()->getElementClassnames();
00059         const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00060         for (std::list<std::string>::iterator itTypenames = elementTypenames->begin(); itTypenames != elementTypenames->end(); itTypenames++) {
00061             if ((*itTypenames) != Util::TypeOf<StatisticsCollector>() && (*itTypenames) != UtilTypeOfCounter) { // STATISTICS COLLECTOR and COUNTERs do NOT need to be saved
00062                 List<ModelElement*>* infras = _model->getElements()->getElementList((*itTypenames));
00063                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing elements of type
00064                 \"\" + (*itTypenames) + "\":");
00065                 Util::IncIndent();
00066                 {
00067                     for (std::list<ModelElement*>::iterator it = infras->list()->begin(); it != infras->list()->end(); it++) {
00068                         _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing " +
00069                         (*itTypenames) + " \"\" + (*it)->getName() + "\"");
00070                         fields = (*it)->SaveInstance((*it));
00071                         Util::IncIndent();
00072                         modelElementsToSave->merge(*_adjustFieldsToSave(fields));
00073                         Util::DecIndent();
00074                     }
00075                 }
00076                 // save components
00077                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Writing components\"\"");
00078                 //List<ModelComponent*>* components = this->_model->getComponents();
00079                 modelComponentsToSave = new std::list<std::string>();
00080                 Util::IncIndent();
00081                 {
00082                     for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00083                         fields = (*it)->SaveInstance((*it));
00084                         Util::IncIndent();
00085                         modelComponentsToSave->merge(*_adjustFieldsToSave(fields));
00086                         Util::DecIndent();
00087                     }
00088                 }
00089                 Util::DecIndent();
00090                 // SAVE FILE
00091                 _model->getTracer()->trace(Util::TraceLevel::toolDetailed, "Saving file");
00092                 Util::IncIndent();
00093                 {
00094                     // open file
00095                     std::ofstream savefile;
00096                     savefile.open(filename, std::ofstream::out);
00097                     savefile << "# Genesys simulation model " << std::endl;
00098                     time_t now = time(0);
00099                     char* dt = ctime(&now);
00100                     savefile << "# Last saved on " << dt;
00101                     savefile << "# simulator infos" << std::endl;
00102                     _saveContent(simulatorInfosToSave, &savefile);
00103                     savefile << "# model infos" << std::endl;
00104                     _saveContent(modelInfosToSave, &savefile);
00105                     savefile << "# simulation infos / experimental design" << std::endl;
00106                     _saveContent(simulationInfosToSave, &savefile);
00107                     savefile << "#" << std::endl;
00108                     savefile << "# model elements" << std::endl;
00109                     _saveContent(modelComponentsToSave, &savefile);
00110                     savefile << "#" << std::endl;
00111                     savefile << "# model components" << std::endl;
00112                     _saveContent(modelComponentsToSave, &savefile);

```

```

00113         savefile.close();
00114     }
00115     Util::DecIndent();
00116 }
00117 Util::DecIndent();
00118 this->_hasChanged = false;
00119 return true; // \todo: check if save really saved successfully
00120 }
00121
00122 void ModelPersistenceDefaultImpl1::_saveContent(std::list<std::string>* content, std::ofstream* file)
{
00123     for (std::list<std::string>::iterator it = content->begin(); it != content->end(); it++) {
00124         *file << (*it) << std::endl << std::endl;
00125     }
00126 }
00127
00128 bool ModelPersistenceDefaultImpl1::_loadFields(std::string line) {
00129     //std::regex regex(R"([=]+)"); // split on space R"([\s]+)" \todo: HOW SEPARATOR WITH MORE THAN
00130     ONE CHAR
00131     _model->getTracer()->trace(Util::TraceLevel::everythingMostDetailed, line);
00132     bool res = true;
00133     std::regex regex(R"([\s]+)"); // split on " " \todo Should be _lineseparator
00134     std::sregex_token_iterator titline.begin(), line.end(), regex, -1;
00135     std::list<std::string> lstfields{tit, {}};
00136     // for each field, separate key and value and form a map
00137     try {
00138         std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00139         regex = {R"([=]+)"};
00140         std::vector<std::string> veckeyval; //{{it,{}}};
00141         unsigned int i = 0;
00142         for (std::list<std::string>::iterator it = lstfields.begin(); it != lstfields.end(); it++) {
00143             //std::cout << (*it) << std::endl;
00144             tit = {(*it).begin(), (*it).end(), regex, -1};
00145             veckeyval = {tit, {}};
00146             trim((veckeyval[0]));
00147             if (veckeyval[0] != "") { // it should always be, rigth? \todo case for assert?
00148                 if (veckeyval.size() > 1) {
00149                     trim((veckeyval[1]));
00150                     if (veckeyval[1].substr(0, 1) == "\\" && veckeyval[1].substr(veckeyval[1].length() - 1, 1) == "") { // remove ""
00151                         veckeyval[1] = veckeyval[1].substr(1, veckeyval[1].length() - 2);
00152                     }
00153                     veckeyval[1] =
00154                     this->_convertLineseparatorReplacementBacktoLineseparator(veckeyval[1]);
00155                     fields->emplace(veckeyval[0], veckeyval[1]);
00156                 } else {
00157                     if (i == 0) {
00158                         fields->emplace("id", veckeyval[0]);
00159                     } else if (i == 1) {
00160                         fields->emplace("typename", veckeyval[0]);
00161                     } else if (i == 2) {
00162                         fields->emplace("name", veckeyval[0]);
00163                     } else {
00164                         fields->emplace(veckeyval[0], "");
00165                     }
00166                 }
00167             }
00168             // now the map<str,str> is ready. Look for the right class to load it
00169             Util::IncIndent();
00170         {
00171             std::string thistypename = (*fields->find("typename")).second;
00172             _model->getTracer()->trace(Util::TraceLevel::toolInternal, "loading " + thistypename +
00173             "");
00174             if (thistypename == "SimulatorInfo") {
00175                 this->_loadSimulatorInfoFields(fields);
00176             } else if (thistypename == "ModelInfo") {
00177                 _model->getInfos()->loadInstance(fields);
00178             } else if (thistypename == "ModelSimulation") {
00179                 _model->getSimulation()->loadInstance(fields);
00180             } else {
00181                 // this should be a ModelComponent or ModelElement.
00182                 //std::string thistypename = (*fields->find("typename")).second;
00183                 ModelElement* newTemUselessElement = ModelElement::LoadInstance(_model, fields,
00184                 false);
00185                 if (newTemUselessElement != nullptr) {
00186                     newTemUselessElement->~ModelElement();
00187                     Plugin* plugin =
00188                     this->_model->getParentSimulator()->getPlugins()->find(thistypename);
00189                     if (plugin != nullptr) {
00190                         res = plugin->loadAndInsertNew(_model, fields);
00191                         // save fields for components, in order to allow to connect components after
00192                         all of them have been loaded
00193                         if (res && plugin->getPluginInfo()->isComponent()) {
00194                             _componentFields->insert(_componentFields->end(), fields);
00195                         // _model->getTraceManager()->trace(Util::TraceLevel::errors, "Inserindo

```

```

00192         fields do componente "+plugin->getPluginInfo()->getPluginTypename());
00193             }
00194         } else {
00195             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error loading file:
00196             Could not identity typename \\" + thistypename + "\\\"");
00197             res = false;
00198         } else {
00199             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error loading file:
00200             Could not identity typename \\" + thistypename + "\\\"");
00201             res = false;
00202         }
00203     }
00204     Util::DecIndent();
00205 } catch (...) {
00206 }
00207
00208 return res;
00209 }
00210
00211 void ModelPersistenceDefaultImpl::_loadSimulatorInfoFields(std::map<std::string, std::string>*
00212 fields) {
00213     unsigned int savedVersionNumber = std::stoi((*fields->find("analystName")).second);
00214     unsigned int simulatorVersionNumber = _model->getParentSimulator()->getVersionNumber();
00215     if (savedVersionNumber != simulatorVersionNumber) {
00216         _model->getTracer()->trace("The version of the saved model differs from the simulator. Loading
00217 may not be possible", Util::TraceLevel::errorRecover);
00218     }
00219 }
00220
00221 bool ModelPersistenceDefaultImpl::load(std::string filename) {
00222     //std::list<Plugin*> plugins = this->_model->getParent()->getPlugins();
00223     //plugins->front()-
00224     //return false;
00225     bool res = true;
00226     _componentFields->clear();
00227     {
00228         std::ifstream modelFile;
00229         std::string inputLine;
00230         try {
00231             modelFile.open(filename);
00232             while (getline(modelFile, inputLine) && res) {
00233                 //trim(&inputLine);
00234                 if (inputLine.substr(0, 1) != "#" && !inputLine.empty()) {
00235                     //Util::IncIndent();
00236                     res &= _loadFields(inputLine);
00237                     //Util::DecIndent();
00238                 }
00239             }
00240             modelFile.close();
00241         } catch (...) {
00242             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Error loading file \\" +
00243 filename + "\\\"");
00244         }
00245         // check if something was loaded
00246         res &= (_model->getComponents()->getNumberOfComponents() > 0) &
00247             (_model->getElements()->getNumberOfElements() > 0);
00248         if (res) {
00249             //
00250             // CONNECT LOADED COMPONENTS (must wait for all components to be loaded so they can be
00251             // connected)
00252             //
00253             ComponentManager* cm = _model->getComponents();
00254             _model->getTracer()->trace(Util::TraceLevel::simulatorDetailed, "Connecting loaded
00255             components");
00256             Util::IncIndent();
00257             {
00258                 for (std::list<std::map<std::string, std::string*>>::iterator it =
00259                     _componentFields->begin(); it != _componentFields->end(); it++) {
00260                     std::map<std::string, std::string*> fields = (*it);
00261                     // find the component
00262                     ModelComponent* thisComponent = nullptr;
00263                     Util::identification thisId = std::stoi((*fields->find("id")).second);
00264                     for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp != cm->end();
00265                         itcomp++) {
00266                         if ((*itcomp)->getId() == thisId) {
00267                             thisComponent = (*itcomp);
00268                             break; // end inner for loop
00269                         }
00270                     }
00271                     assert(thisComponent != nullptr);
00272                 }
00273             }
00274         }
00275     }
00276 }
```

```

00268         // find the next components connected with this one
00269         unsigned short nextSize = 1;
00270         if (*fields->find("nextSize") != *fields->end()) { // found nextSize
00271             nextSize = std::stoi((*fields->find("nextSize")).second);
00272         }
00273         for (unsigned short i = 0; i < nextSize; i++) {
00274             Util::identification nextId = std::stoi((*fields->find("nextId" +
00275                 std::to_string(i))).second);
00276             unsigned short nextInputNumber = 0; // default value if it is not found below
00277             if (fields->find("nextInputNumber" + std::to_string(i)) != fields->end())
00278                 nextInputNumber = std::stoi((*fields->find("nextInputNumber" +
00279                     std::to_string(i))).second);
00280             ModelComponent* nextComponent = nullptr;
00281             for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp !=
00282                 cm->end(); itcomp++) { // connect the components
00283                 if ((*itcomp)->getId() == nextId) { // connect the components
00284                     nextComponent = (*itcomp);
00285                     thisComponent->getNextComponents()->insert(nextComponent,
00286                         nextInputNumber);
00287                     _model->getTracer()->trace(Util::TraceLevel::toolDetailed,
00288                         thisComponent->getName() + "<" + std::to_string(i) + ">" + " -> " + nextComponent->getName() + "<" +
00289                         std::to_string(nextInputNumber) + ">");
00290                     break;
00291                 }
00292             Util::DecIndent();
00293             _model->getTracer()->trace(Util::TraceLevel::simulatorInternal, "File successfully loaded with
00294             " + std::to_string(_model->getComponents()->getNumberOfComponents()) + " components and " +
00295             std::to_string(_model->getElements()->getNumberOfElements()) + " elements");
00296         }
00297         Util::DecIndent();
00298         if (res) {
00299             _hasChanged = false;
00300         }
00301     }
00302     std::string
00303     ModelPersistenceDefaultImpl1::_convertLineseparatorReplacementBackToLineseparator(std::string str) {
00304         size_t index = str.find(_fieldseparatorReplacement, 0);
00305         while (index != std::string::npos) {
00306             str.replace(index, _fieldseparatorReplacement.length(), _fieldseparator);
00307             /* Advance index forward so the next iteration doesn't pick it up as well. */
00308             index = str.find(_fieldseparatorReplacement, index + _fieldseparator.length());
00309         }
00310     }
00311
00312     std::string ModelPersistenceDefaultImpl1::_convertLineseparatorToLineseparatorReplacement(std::string
00313         str) {
00314         size_t index = str.find(_fieldseparator, 0);
00315         while (index != std::string::npos) {
00316             str.replace(index, _fieldseparator.length(), _fieldseparatorReplacement);
00317             /* Advance index forward so the next iteration doesn't pick it up as well. */
00318             index = str.find(_fieldseparator, index + _fieldseparatorReplacement.length());
00319         }
00320     }
00321
00322     std::list<std::string>* ModelPersistenceDefaultImpl1::_adjustFieldsToSave(std::map<std::string,
00323         std::string>* fields) {
00324         std::list<std::string>* newList = new std::list<std::string>();
00325         std::string newStr, strV2003, idV2003, typenameV2003, nameV2003; //, strV210329;
00326         idV2003 = "0";
00327         std::string attrValue;
00328         for (std::map<std::string, std::string>::iterator it = fields->begin(); it != fields->end(); it++)
00329         {
00330             //newStr += (*it).first + "=" + (*it).second + this->_linefieldseparator;
00331             if ((*it).first == "id")
00332                 idV2003 = (*it).second;
00333             else if ((*it).first == "typename")
00334                 typenameV2003 = (*it).second;
00335             else if ((*it).first == "name")
00336                 nameV2003 = _convertLineseparatorToLineseparatorReplacement((*it).second); //(*it).second;
00337             else {
00338                 // version V210329: (*it).second should NEVER contain _linefieldseparator. So, replace it
00339                 by _linefieldseparatorReplacement
00340                 attrValue = _convertLineseparatorToLineseparatorReplacement((*it).second);
00341                 strV2003 += (*it).first + "=" + attrValue + _fieldseparator;
00342             }
00343         }
00344     while (idV2003.length() < 3)

```

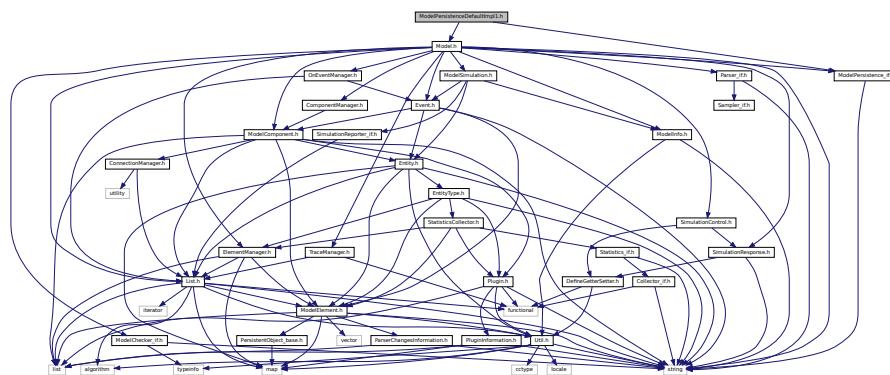
```

00342     idv2003 += _fieldseparator;
00343     //while (typenameV2003.length() < 15)
00344     // typenameV2003 += _fieldseparator;
00345     strv2003 = idv2003 + _fieldseparator + typenameV2003 + _fieldseparator + strv2003;
00346     _model->getTracer()->trace(Util::TraceLevel::toolDetailed, strv2003); //newStr
00347     newList->push_back(strv2003); //newStr
00348     return newList;
00349 }
00350
00351 bool ModelPersistenceDefaultImpl1::hasChanged() {
00352     return _hasChanged;
00353 }

```

9.277 ModelPersistenceDefaultImpl1.h File Reference

```
#include "ModelPersistence_if.h"
#include "Model.h"
Include dependency graph for ModelPersistenceDefaultImpl1.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ModelPersistenceDefaultImpl1](#)

9.278 ModelPersistenceDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelPersistenceDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 20 de Maio de 2019, 20:41
00012 */
00013
00014 #ifndef MODEL_PERSISTENCE_DEFAULT_IMPL1_H
00015 #define MODEL_PERSISTENCE_DEFAULT_IMPL1_H

```

```

00016
00017 #include "ModelPersistence_if.h"
00018 #include "Model.h"
00019
00020 //namespace GenesysKernel {
00021
00022 class ModelPersistenceDefaultImpl : public ModelPersistence_if {
00023 public:
00024     ModelPersistenceDefaultImpl(Model* model);
00025     virtual ~ModelPersistenceDefaultImpl() = default;
00026 public:
00027     virtual bool save(std::string filename);
00028     virtual bool load(std::string filename);
00029     virtual bool hasChanged();
00030 private:
00031     void _saveContent(std::list<std::string>* content, std::ofstream* file);
00032     bool _loadFields(std::string line);
00033     void _loadSimulatorInfoFields(std::map<std::string, std::string>* fields);
00034     std::list<std::string>* _adjustFieldsToSave(std::map<std::string, std::string>* fields);
00035     std::string _convertLineSeparatorToLineSeparatorReplacement(std::string str);
00036     std::string _convertLineSeparatorReplacementBackToLineSeparator(std::string str);
00037     std::map<std::string, std::string>* _getSimulatorInfoFieldsToSave();
00038 private:
00039     std::list<std::map<std::string, std::string>>* _componentFields = new
00040     std::list<std::map<std::string, std::string>>();
00041     Model* _model = nullptr;
00042     bool _hasChanged = false;
00043     std::string _fieldseparator = " "; // fields are separated by space
00044     std::string _fieldseparatorReplacement = "\\_"; // in cases where spaces are in data to be
00045     saved, they are replaced by this pattern, so there will not be separators inside data (replacement is
00046     \_);
00047 //namespace\\}
00048 #endif /* MODELPERSTENCEDEFAULTIMPL_H */
00049

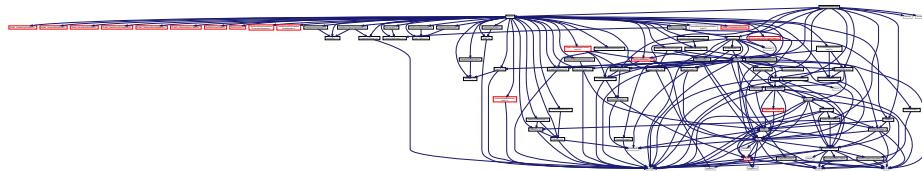
```

9.279 ModelSimulation.cpp File Reference

```

#include "ModelSimulation.h"
#include <iostream>
#include <cassert>
#include <chrono>
#include "Model.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "StatisticsCollector.h"
#include "Counter.h"
#include "Traits.h"
#include "SimulationControl.h"
#include "ComponentManager.h"
Include dependency graph for ModelSimulation.cpp:

```



9.280 ModelSimulation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 */

```

```

00008 * File: ModelSimulation.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 18:04
00012 */
00013
00014 #include "ModelSimulation.h"
00015 #include <iostream>
00016 #include <cassert>
00017 #include <chrono>
00018 #include "Model.h"
00019 #include "Simulator.h"
00020 #include "SourceModelComponent.h"
00021 #include "StatisticsCollector.h"
00022 #include "Counter.h"
00023 #include "Traits.h"
00024 #include "SimulationControl.h"
00025 #include "ComponentManager.h"
00026
00027 //using namespace GenesysKernel;
00028
00029 ModelSimulation::ModelSimulation(Model* model) {
00030     _model = model;
00031     _info = model->getInfos();
00032     _statsCountersSimulation->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00033         return a->getId() < b->getId();
00034     });
00035     _simulationReporter = new Traits<SimulationReporter_if>::Implementation(this, model,
00036     this->_statsCountersSimulation);
00037 }
00038 std::string ModelSimulation::show() {
00039     return "numberOfReplications=" + std::to_string(_numberOfReplications) +
00040             ",replicationLength=" + std::to_string(_replicationLength) + " " +
00041             Util::StrTimeUnit(this->_replicationLengthTimeUnit) +
00042             ",terminatingCondition=\"" + this->_terminatingCondition + "\" " +
00043             ",warmupTime=" + std::to_string(this->_warmUpPeriod) + " " +
00044             Util::StrTimeUnit(this->_warmUpPeriodTimeUnit);
00045 }
00046 bool ModelSimulation::_isReplicationEndCondition() {
00047     bool finish = _model->getFutureEvents()->size() == 0;
00048     finish |= _model->parseExpression(_terminatingCondition) != 0.0;
00049     if (_model->getFutureEvents()->size() > 0 && !finish) {
00050         // replication length has not been achieve (so far), but next event will happen after that,
00051         // so it's just fine to set now as the replicationLength
00052         finish |= _model->getFutureEvents()->front()->getTime() > _replicationLength;
00053     }
00054     return finish;
00055 }
00056 void ModelSimulation::_traceReplicationEnded() {
00057     std::string causeTerminated = "";
00058     if (_model->getFutureEvents()->empty()) {
00059         causeTerminated = "event queue is empty";
00060     } else if (_stopRequested) {
00061         causeTerminated = "user requested to stop";
00062     } else if (_model->getFutureEvents()->front()->getTime() > _replicationLength) {
00063         causeTerminated = "replication length " + std::to_string(_replicationLength) + " was
00064         achieved";
00065     } else if (_model->parseExpression(_terminatingCondition)) {
00066         causeTerminated = "termination condition was achieved";
00067     } else causeTerminated = "unknown";
00068     std::string message = "Replication " + std::to_string(_currentReplicationNumber) + " of " +
00069     std::to_string(_numberOfReplications) + " has finished with last event at time " +
00070     std::to_string(_simulatedTime) + " because " + causeTerminated;
00071     _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, message);
00072 }
00073 void ModelSimulation::start() {
00074     if (!_isPaused) { // begin of a new simulation
00075         Util::SetIndent(0); //force indentation
00076         if (!_model->check()) {
00077             _model->getTracer()->trace(Util::TraceLevel::errorFatal, "Model check failed. Cannot start
00078             simulation.");
00079             return;
00080         }
00081         _initSimulation();
00082         _model->getOnEvents()->NotifySimulationStartHandlers(new SimulationEvent(0, nullptr));
00083         _currentReplicationNumber = 1;
00084         Util::IncIndent();
00085         _initReplication();
00086     } else { // continue after a pause
00087         _model->getTracer()->trace("Replication resumed", Util::TraceLevel::modelSimulationEvent);
00088         _model->getOnEvents()->NotifySimulationPausedStartHandlers(new SimulationEvent(0, nullptr));
00089     }
00090     _isRunning = true;

```

```

00090     _isPaused = false;
00091     //std::chrono::time_point<std::chrono::_V2::system_clock, std::chrono::duration<long int,
00092     std::ratio < 1, 1000000000 > replicationStartTime = std::chrono::high_resolution_clock::now();
00093     do {
00094         Util::SetIndent(1);
00095         _model->getOnEvents()->NotifyReplicationStartHandlers(new
00096             SimulationEvent(_currentReplicationNumber, nullptr));
00097         // main simulation loop
00098         Util::IncIndent();
00099         while (!_isReplicationEndCondition() && !_pauseRequested) {
00100             _stepSimulation();
00101             if (!_pauseRequested) {
00102                 Util::SetIndent(1); // force
00103                 _replicationEnded();
00104                 _currentReplicationNumber++;
00105                 if (_currentReplicationNumber <= _numberOfReplications) {
00106                     _initReplication();
00107                 }
00108             } while (_currentReplicationNumber <= _numberOfReplications && !_pauseRequested);
00109             if (!_pauseRequested) {
00110                 if (this->_showReportsAfterSimulation)
00111                     _simulationReporter->showSimulationStatistics(); //_cStatsSimulation);
00112             Util::DecIndent();
00113
00114             _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, "Simulation of model \""
00115             _info->getName() + "\" has finished.\n");
00116             _model->getOnEvents()->NotifySimulationEndHandlers(new SimulationEvent(0, nullptr));
00117         } else {
00118             _pauseRequested = false;
00119             _isPaused = true;
00120         }
00121         _isRunning = false;
00122     }
00123 void ModelSimulation::_replicationEnded() {
00124     _traceReplicationEnded();
00125     _model->getOnEvents()->NotifyReplicationEndHandlers(new SimulationEvent(_currentReplicationNumber,
00126         nullptr));
00127     if (this->_showReportsAfterReplication)
00128         _simulationReporter->showReplicationStatistics();
00129 // _simulationReporter->showSimulationResponses();
00130     _actualizeSimulationStatistics();
00131 }
00132 void ModelSimulation::_actualizeSimulationStatistics() {
00133     //\todo: should not be only CSTAT and COUNTER, but any element that generateReportInformation
00134     const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00135     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00136
00137     StatisticsCollector *sc, *scSim;
00138     //ModelElement* me;
00139     List<ModelElement*>* cstats =
00140         _model->getElements()->elementList(Util::TypeOf<StatisticsCollector>());
00141     for (std::list<ModelElement*>::iterator itMod = cstats->list()->begin(); itMod != cstats->list()->end(); itMod++) {
00142         sc = dynamic_cast<StatisticsCollector*> ((*itMod));
00143         scSim = nullptr;
00144         for (std::list<ModelElement*>::iterator itSim = _statsCountersSimulation->list()->begin();
00145             itSim != _statsCountersSimulation->list()->end(); itSim++) {
00146             if ((*itSim)->getClassname() == UtilTypeOfStatisticsCollector) {
00147                 if ((*itSim)->getName() == _cte_stCountSimulNamePrefix + sc->getName() &&
00148                     dynamic_cast<StatisticsCollector*> ((*itSim)->getParent() == sc->getParent()) {
00149                     // found
00150                     scSim = dynamic_cast<StatisticsCollector*> ((*itSim));
00151                     break;
00152                 }
00153             }
00154             //scSim = dynamic_cast<StatisticsCollector*> (*(this->_statsCountersSimulation->find(*it)));
00155             if (scSim == nullptr) {
00156                 // this is a cstat created during the last replication
00157                 // scSim = new StatisticsCollector(_model->elements(), sc->name(), sc->getParent());
00158                 _statsCountersSimulation->insert(scSim);
00159             }
00160             assert(scSim != nullptr);
00161             scSim->getStatistics()->getCollector()->addValue(sc->getStatistics()->average());
00162             Counter *cnt; //, *cntSim;
00163             List<ModelElement*>* counters = _model->getElements()->elementList(Util::TypeOf<Counter>());
00164             for (std::list<ModelElement*>::iterator itMod = counters->list()->begin(); itMod != counters->list()->end(); itMod++) {
00165                 cnt = dynamic_cast<Counter*> ((*itMod));
00166                 //cntSim = nullptr;
00167                 scSim = nullptr;
00168                 for (std::list<ModelElement*>::iterator itSim = _statsCountersSimulation->list()->begin();

```

```

00168     itSim != _statsCountersSimulation->list()->end(); itSim++) {
00169         if ((*itSim)->getClassName() == UtilTypeOfStatisticsCollector) {
00170             // _model->getTraceManager()->trace(Util::TraceLevel::simulation, (*itSim)->getName() +
00171             " == "+_cte_stCountSimulNamePrefix + cnt->getName());
00172             if ((*itSim)->getName() == _cte_stCountSimulNamePrefix + cnt->getName() &&
00173                 dynamic_cast<StatisticsCollector*>(*itSim)->getParent() == cnt->getParent()) {
00174                 // found
00175                 scSim = dynamic_cast<StatisticsCollector*>(*itSim);
00176                 break;
00177             }
00178             if ((*itSim)->getTypename() == UtilTypeOfCounter) {
00179                 if ((*itSim)->getName() == _cte_stCountSimulNamePrefix + cnt->getName() &&
00180                     dynamic_cast<Counter*>(*itSim)->getParent() == cnt->getParent()) {
00181                     // found
00182                     cntSim = dynamic_cast<Counter*>(*itSim);
00183                     break;
00184                 }
00185             }
00186             /*
00187             assert(cntSim != nullptr);
00188             cntSim->incCountValue(cnt->getCountValue());
00189             */
00190             assert(scSim != nullptr);
00191             scSim->getStatistics()->getCollector()->addValue(cnt->getCountValue());
00192         }
00193     }
00194
00195 void ModelSimulation::_showSimulationHeader() {
00196     TraceManager* tm = _model->getTracer();
00197     tm->traceReport("\n-----");
00198     // simulator infos
00199     tm->traceReport(_model->getParentSimulator()->getName());
00200     tm->traceReport(_model->getParentSimulator()->getLicenceManager()->showLicence());
00201     tm->traceReport(_model->getParentSimulator()->getLicenceManager()->showLimits());
00202     // model infos
00203     tm->traceReport("Analyst Name: " + _info->getAnalystName());
00204     tm->traceReport("Project Title: " + _info->getProjectTitle());
00205     tm->traceReport("Number of Replications: " + std::to_string(_numberOfReplications));
00206     tm->traceReport("Replication Length: " + std::to_string(_replicationLength) + " " +
00207     Util::StrTimeUnit(_replicationLengthTimeUnit));
00208     //tm->traceReport(Util::TraceLevel::simulation, "");
00209     // model controls and responses
00210     std::string controls;
00211     for (std::list<SimulationControl*>::iterator it = _model->getControls()->list()->begin(); it !=
00212     _model->getControls()->list()->end(); it++) {
00213         controls += (*it)->getName() + "(" + (*it)->getType() + ")=" +
00214         std::to_string((*it)->getValue()) + ", ";
00215     }
00216     controls = controls.substr(0, controls.length() - 2);
00217     tm->traceReport("> Simulation controls: " + controls);
00218     std::string responses;
00219     for (std::list<SimulationResponse*>::iterator it = _model->getResponses()->list()->begin(); it !=
00220     _model->getResponses()->list()->end(); it++) {
00221         responses += (*it)->getName() + "(" + (*it)->getType() + "), ";
00222     }
00223     responses = responses.substr(0, responses.length() - 2);
00224     tm->traceReport("> Simulation responses: " + responses);
00225     tm->traceReport("");
00226 }
00227 void ModelSimulation::_initSimulation() {
00228     _showSimulationHeader();
00229     //model->tracer()->trace(Util::TraceLevel::modelSimulationEvent,
00230     "-----");
00231     _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, "");
00232     _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, "Simulation of model \" +
00233     _info->getName() + "\" is starting.");
00234     // copy all CStats and Counters (used in a replication) to CStats and counters for the whole
00235     // simulation
00236     // \todo: Should not be CStats and Counters, but any element that generates report importation
00237     this->_statsCountersSimulation->clear();
00238     StatisticsCollector* cstat;
00239     List<ModelElement*>* cstats =
00240     _model->getElements()->getElementList(Util::TypeOf<StatisticsCollector>());
00241     for (std::list<ModelElement*>::iterator it = cstats->list()->begin(); it != cstats->list()->end();
00242     it++) {
00243         cstat = dynamic_cast<StatisticsCollector*>((*it));
00244         // this new CSat should NOT be inserted into the model
00245         StatisticsCollector* newCStatSimul = new StatisticsCollector(_model,
00246         _cte_stCountSimulNamePrefix + cstat->getName(), cstat->getParent(), false);
00247         //_model->elements()->remove(Util::TypeOf<StatisticsCollector>(), newCStatSimul); // remove
00248         // from model, since it is automatically inserted by the constructor
00249         this->_statsCountersSimulation->insert(newCStatSimul);

```

```

00243     }
00244     // copy all Counters (used in a replication) to Counters for the whole simulation
00245     // \todo: Counters in replication should be converted into CStats in simulation. Each value
00246     // counted in a replication should be added in a CStat for Stats.
00247     Counter* counter;
00248     List<ModelElement*>* counters = _model->getElements()->getElementList(Util::TypeOf<Counter>());
00249     for (std::list<ModelElement*>::iterator it = counters->list()->begin(); it != counters->list()->end(); it++) {
00250         counter = dynamic_cast<Counter*> ((*it));
00251         /*
00252         Counter* newCountSimul = new Counter(_cte_stCountSimulNamePrefix + counter->getName(),
00253         counter->getParent());
00254         this->_statsCountersSimulation->insert(newCountSimul);
00255         */
00256         // addin a cstat (to stat the counts)
00257         StatisticsCollector* newCStatSimul = new StatisticsCollector(_model,
00258             _cte_stCountSimulNamePrefix + counter->getName(), counter->getParent(), false);
00259         this->_statsCountersSimulation->insert(newCStatSimul);
00260     }
00261     this->_simulationIsInitiated = true; // \todo Check the uses of _simulationIsInitiated and when it
00262     // should be set to false
00263
00264 void ModelSimulation::_initReplication() {
00265     TraceManager* tm = _model->getTracer();
00266     tm->trace(Util::TraceLevel::modelSimulationEvent, "");
00267     tm->trace(Util::TraceLevel::modelSimulationEvent, "Replication " +
00268     std::to_string(_currentReplicationNumber) + " of " + std::to_string(_numberOfReplications) + " is
00269     starting.");
00270
00271     _model->getFutureEvents()->clear();
00272     _simulatedTime = 0.0;
00273     _pauseRequested = false;
00274
00275     //if (_currentReplicationNumber > 1) {
00276     // init all components between replications
00277     for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00278         ModelElement::InitBetweenReplications((*it));
00279     }
00280     // init all elements between replications
00281     ModelElement* element;
00282     std::string elementType;
00283     //std::string* errorMessage = new std::string();
00284     std::list<std::string>* elementTypes = _model->getElements()->getElementClassnames();
00285     for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end(); typeIt++) {
00286         elementType = (*typeIt);
00287         List<ModelElement*>* elements = _model->getElements()->getElementList(elementType);
00288         for (std::list<ModelElement*>::iterator it = elements->list()->begin(); it != elements->list()->end(); it++) {
00289             element = (*it);
00290             ModelElement::InitBetweenReplications(element);
00291         }
00292     //}
00293     Util::ResetIdOfType(Util::TypeOf<Entity>());
00294     Util::ResetIdOfType(Util::TypeOf<Event>());
00295
00296     // insert first creation events
00297     SourceModelComponent *source;
00298     Entity *newEntity;
00299     Event *newEvent;
00300     double creationTime;
00301     unsigned int numToCreate;
00302     //std::list<ModelComponent*>* list = _model->getComponents()->list();
00303     for (std::list<ModelComponent*>::iterator it = _model->getComponents()->begin(); it != _model->getComponents()->end(); it++) {
00304         source = dynamic_cast<SourceModelComponent*> ((*it));
00305         if (source != nullptr) {
00306             creationTime = source->getFirstCreation();
00307             numToCreate = source->getEntitiesPerCreation();
00308             for (unsigned int i = 1; i <= numToCreate; i++) {
00309                 newEntity = new Entity(_model);
00310                 newEntity->setEntityType(source->getEntityType());
00311                 newEvent = new Event(creationTime, newEntity, (*it));
00312                 _model->getFutureEvents()->insert(newEvent);
00313             }
00314             source->setEntitiesCreated(numToCreate);
00315         }
00316     }
00317     if (this->_initializeStatisticsBetweenReplications) {
00318         _initStatistics();
00319     }
00320     this->_replicationIsInitiated = true; // \todo Check the uses of _replicationIsInitiated and when

```

```

        it should be set to false
00319 }
00320
00321 void ModelSimulation::_initStatistics() {
00322     StatisticsCollector* cstat;
00323     List<ModelElement*>* list =
00324         _model->getElements()->getElementList(Util::TypeOf<StatisticsCollector>());
00325     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end(); it++) {
00326         cstat = (StatisticsCollector*) (*it);
00327         cstat->getStatistics()->getCollector()->clear();
00328     }
00329     Counter* counter;
00330     list = _model->getElements()->getElementList(Util::TypeOf<Counter>());
00331     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->list()->end(); it++) {
00332         counter = (Counter*) (*it);
00333         counter->clear();
00334     }
00335
00336 void ModelSimulation::_checkWarmUpTime(Event* nextEvent) {
00337     double warmupTime = Util::TimeUnitConvert(this->_warmUpPeriodTimeUnit,
00338         this->replicationLengthTimeUnit);
00339     warmupTime *= _warmUpPeriod;
00340     if (warmupTime > 0.0 && _model->getSimulation()->getSimulatedTime() <= warmupTime &&
00341         nextEvent->getTime() > warmupTime) {// warmupTime. Time to initStats
00342         _model->getTracer()->trace(Util::TraceLevel::modelInternal, "Warmup time reached. Statistics
00343             are being reseted.");
00344         _initStatistics();
00345     }
00346 }
00347
00348 void ModelSimulation::_stepSimulation() {
00349     // process one single event
00350     Event* nextEvent;
00351     nextEvent = _model->getFutureEvents()->front();
00352     if (_warmUpPeriod > 0.0)
00353         _checkWarmUpTime(nextEvent);
00354     if (nextEvent->getTime() <= _replicationLength) {
00355         if (_checkBreakpointAt(nextEvent)) {
00356             this->_pauseRequested = true;
00357         } else {
00358             _model->getFutureEvents()->pop_front();
00359             _model->getOnEvents()->NotifyReplicationStepHandlers(new
00360                 SimulationEvent(_currentReplicationNumber, nullptr));
00361             _processEvent(nextEvent);
00362         }
00363     } else {
00364         this->_simulatedTime = _replicationLength;
00365     }
00366 }
00367
00368 bool ModelSimulation::_checkBreakpointAt(Event* event) {
00369     bool res = false;
00370     SimulationEvent* se = new SimulationEvent(_currentReplicationNumber, event);
00371     if (_breakpointsOnComponent->find(event->getComponent()) !=
00372         _breakpointsOnComponent->list()->end()) {
00373         if (_justTriggeredBreakpointsOnComponent == event->getComponent()) {
00374             _justTriggeredBreakpointsOnComponent = nullptr;
00375         } else {
00376             _justTriggeredBreakpointsOnComponent = event->getComponent();
00377             _model->getOnEvents()->NotifyBreakpointHandlers(se);
00378             _model->getTracer()->trace("Breakpoint found at component '" +
00379                 event->getComponent()->getName() + "'. Replication is paused.",
00380                 Util::TraceLevel::modelSimulationEvent);
00381             res = true;
00382         }
00383     } else if (_breakpointsOnEntity->find(event->getEntity()) != _breakpointsOnEntity->list()->end()) {
00384         if (_justTriggeredBreakpointsOnEntity == event->getEntity()) {
00385             _justTriggeredBreakpointsOnEntity = nullptr;
00386         } else {
00387             _justTriggeredBreakpointsOnEntity = event->getEntity();
00388             _model->getOnEvents()->NotifyBreakpointHandlers(se);
00389             _model->getTracer()->trace("Breakpoint found at entity '" + event->getEntity()->getName()
00390                 + "'. Replication is paused.", Util::TraceLevel::modelSimulationEvent);
00391             res = true;
00392         }
00393     }
00394     double time;
00395     for (std::list<double>::iterator it = _breakpointsOnTime->list()->begin(); it != _breakpointsOnTime->list()->end(); it++) {
00396         time = (*it);
00397         if (_simulatedTime < time && event->getTime() >= time) {
00398             if (_justTriggeredBreakpointsOnTime == time) { // just triggered this breakpoint

```

```

00393             _justTriggeredBreakpointsOnTime = 0.0;
00394         } else {
00395             _justTriggeredBreakpointsOnTime = time;
00396             _model->getOnEvents()->NotifyBreakpointHandlers(se);
00397             _model->getTracer()->trace("Breakpoint found at time '" +
00398                 std::to_string(event->getTime()) + "'. Replication is paused.");
00399             Util::TraceLevel::modelSimulationEvent);
00400         }
00401     }
00402     return res;
00403 }
00404
00405 void ModelSimulation::_processEvent(Event* event) {
00406     // _model->tracer()->traceSimulation(Util::TraceLevel::modelSimulationEvent, event->time(),
00407     event->entity(), event->component(), "");
00408     _model->getTracer()->trace(Util::TraceLevel::modelSimulationEvent, "Event: " + event->show() +
00409     "");
00410     Util::IncIndent();
00411     _model->getTracer()->trace(Util::TraceLevel::modelSimulationInternal, "Entity: " +
00412     event->getEntity()->show());
00413     this->_currentEntity = event->getEntity();
00414     this->_currentComponent = event->getComponent();
00415     this->_currentInputNumber = event->getComponentInputNumber();
00416     assert(_simulatedTime <= event->getTime());
00417     _simulatedTime = event->getTime();
00418     _model->getOnEvents()->NotifyProcessEventHandlers(new SimulationEvent(_currentReplicationNumber,
00419     event));
00420     try {
00421         //event->getComponent()->Execute(event->getEntity(), event->getComponent()); // Execute is
00422         static
00423             ModelComponent::Execute(event->getEntity(), event->getComponent(),
00424             event->getComponentInputNumber());
00425     } catch (std::exception *e) {
00426         _model->getTracer()->traceError(*e, "Error on processing event (" + event->show() + ")");
00427     }
00428     Util::DecIndent();
00429 }
00430
00431 void ModelSimulation::step() {
00432     if (_simulationIsInitiated && _replicationIsInitiated) {
00433         if (!_isReplicationEndCondition()) {
00434             try {
00435                 this->_stepSimulation();
00436             } catch (std::exception *e) {
00437                 _model->getTracer()->traceError(*e, "Error on simulation step");
00438             }
00439     }
00440     void ModelSimulation::stop() {
00441         this->_isPaused = false;
00442         this->_isRunning = false;
00443         this->_replicationIsInitiated = false;
00444         this->_simulationIsInitiated = false;
00445     }
00446
00447 void ModelSimulation::setPauseOnEvent(bool _pauseOnEvent) {
00448     this->_pauseOnEvent = _pauseOnEvent;
00449 }
00450
00451 bool ModelSimulation::isPauseOnEvent() const {
00452     return _pauseOnEvent;
00453 }
00454
00455 void ModelSimulation::setInitializeStatistics(bool _initializeStatistics) {
00456     this->_initializeStatisticsBetweenReplications = _initializeStatistics;
00457 }
00458
00459 bool ModelSimulation::isInitializeStatistics() const {
00460     return _initializeStatisticsBetweenReplications;
00461 }
00462
00463 void ModelSimulation::setInitializeSystem(bool _initializeSystem) {
00464     this->_initializeSystem = _initializeSystem;
00465 }
00466
00467 bool ModelSimulation::isInitializeSystem() const {
00468     return _initializeSystem;
00469 }
00470
00471 void ModelSimulation::setStepByStep(bool _stepByStep) {

```

```
00472     this->_stepByStep = _stepByStep;
00473 }
00474
00475 bool ModelSimulation::isStepByStep() const {
00476     return _stepByStep;
00477 }
00478
00479 void ModelSimulation::setPauseOnReplication(bool _pauseOnReplication) {
00480     this->_pauseOnReplication = _pauseOnReplication;
00481 }
00482
00483 bool ModelSimulation::isPauseOnReplication() const {
00484     return _pauseOnReplication;
00485 }
00486
00487 double ModelSimulation::getSimulatedTime() const {
00488     return _simulatedTime;
00489 }
00490
00491 bool ModelSimulation::isRunning() const {
00492     return _isRunning;
00493 }
00494
00495 unsigned int ModelSimulation::getCurrentReplicationNumber() const {
00496     return _currentReplicationNumber;
00497 }
00498
00499 ModelComponent* ModelSimulation::getCurrentComponent() const {
00500     return _currentComponent;
00501 }
00502
00503 Entity* ModelSimulation::getCurrentEntity() const {
00504     return _currentEntity;
00505 }
00506
00507 void ModelSimulation::setReporter(SimulationReporter_if* _simulationReporter) {
00508     this->_simulationReporter = _simulationReporter;
00509 }
00510
00511 SimulationReporter_if* ModelSimulation::getReporter() const {
00512     return _simulationReporter;
00513 }
00514
00515 unsigned int ModelSimulation::getCurrentInputNumber() const {
00516     return _currentInputNumber;
00517 }
00518
00519 void ModelSimulation:: setShowReportsAfterReplication(bool showReportsAfterReplication) {
00520     this->_showReportsAfterReplication = showReportsAfterReplication;
00521 }
00522
00523 bool ModelSimulation::isShowReportsAfterReplication() const {
00524     return _showReportsAfterReplication;
00525 }
00526
00527 void ModelSimulation:: setShowReportsAfterSimulation(bool showReportsAfterSimulation) {
00528     this->_showReportsAfterSimulation = showReportsAfterSimulation;
00529 }
00530
00531 bool ModelSimulation::isShowReportsAfterSimulation() const {
00532     return _showReportsAfterSimulation;
00533 }
00534
00535 List<double>* ModelSimulation::getBreakpointsOnTime() const {
00536     return _breakpointsOnTime;
00537 }
00538
00539 List<Entity*>* ModelSimulation::getBreakpointsOnEntity() const {
00540     return _breakpointsOnEntity;
00541 }
00542
00543 List<ModelComponent*>* ModelSimulation::getBreakpointsOnComponent() const {
00544     return _breakpointsOnComponent;
00545 }
00546
00547 bool ModelSimulation::isPaused() const {
00548     return _isPaused;
00549 }
00550
00551 void ModelSimulation::setNumberOfReplications(unsigned int _numberOfReplications) {
00552     this->_numberOfReplications = _numberOfReplications;
00553     _hasChanged = true;
00554 }
00555
00556 unsigned int ModelSimulation::getNumberOfReplications() const {
00557     return _numberOfReplications;
00558 }
```

```

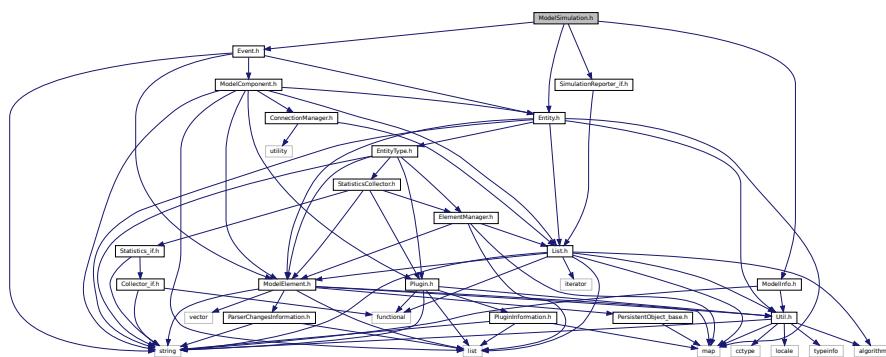
00559
00560 void ModelSimulation::setReplicationLength(double _replicationLength) {
00561     this->_replicationLength = _replicationLength;
00562     _hasChanged = true;
00563 }
00564
00565 double ModelSimulation::getReplicationLength() const {
00566     return _replicationLength;
00567 }
00568
00569 void ModelSimulation::setReplicationLengthTimeUnit(Util::TimeUnit _replicationLengthTimeUnit) {
00570     this->_replicationLengthTimeUnit = _replicationLengthTimeUnit;
00571     _hasChanged = true;
00572 }
00573
00574 Util::TimeUnit ModelSimulation::getReplicationLengthTimeUnit() const {
00575     return _replicationLengthTimeUnit;
00576 }
00577
00578 void ModelSimulation::setWarmUpPeriod(double _warmUpPeriod) {
00579     this->_warmUpPeriod = _warmUpPeriod;
00580     _hasChanged = true;
00581 }
00582
00583 double ModelSimulation::getWarmUpPeriod() const {
00584     return _warmUpPeriod;
00585 }
00586
00587 void ModelSimulation::setWarmUpPeriodTimeUnit(Util::TimeUnit _warmUpPeriodTimeUnit) {
00588     this->_warmUpPeriodTimeUnit = _warmUpPeriodTimeUnit;
00589     _hasChanged = true;
00590 }
00591
00592 Util::TimeUnit ModelSimulation::getWarmUpPeriodTimeUnit() const {
00593     return _warmUpPeriodTimeUnit;
00594 }
00595
00596 void ModelSimulation::setTerminatingCondition(std::string _terminatingCondition) {
00597     this->_terminatingCondition = _terminatingCondition;
00598     _hasChanged = true;
00599 }
00600
00601 std::string ModelSimulation::getTerminatingCondition() const {
00602     return _terminatingCondition;
00603 }
00604
00605 void ModelSimulation::loadInstance(std::map<std::string, std::string>* fields) {
00606     this->_numberOfReplications = std::stoi(loadField(fields, "numberOfReplications", "1"));
00607     this->_replicationLength = std::stod(loadField(fields, "replicationLength", "3600.0"));
00608     this->_replicationLengthTimeUnit = static_cast<Util::TimeUnit>
00609         (std::stoi((*fields->find("replicationLengthTimeUnit")).second));
00610     this->_terminatingCondition = loadField(fields, "terminatingCondition", "");
00611     this->_warmUpPeriod = std::stod(loadField(fields, "warmUpTime", "0.0"));
00612     this->_warmUpPeriodTimeUnit = static_cast<Util::TimeUnit>
00613         (std::stoi((*fields->find("warmUpTimeTimeUnit")).second));
00614     _hasChanged = false;
00615 }
00616
00617 // \todo:: implement check method (to check things like terminating condition)
00618 std::map<std::string, std::string>* ModelSimulation::saveInstance() {
00619     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00620     fields->emplace("typename", "ModelSimulation");
00621     if (_numberOfReplications != 1) fields->emplace("numberOfReplications",
00622         std::to_string(_numberOfReplications));
00623     if (this->_replicationLength != 3600) fields->emplace("replicationLength",
00624         std::to_string(_replicationLength));
00625     fields->emplace("replicationLengthTimeUnit", std::to_string(static_cast<int>
00626         (getReplicationLengthTimeUnit())));
00627     if (this->_terminatingCondition != "") fields->emplace("terminatingCondition", "\"" +
00628         _terminatingCondition + "\"");
00629     if (this->_warmUpPeriod) fields->emplace("warmUpTime", std::to_string(_warmUpPeriod));
00630     fields->emplace("warmUpTimeTimeUnit", std::to_string(static_cast<int>
00631         (getWarmUpPeriodTimeUnit())));
00632     _hasChanged = false;
00633     return fields;
00634 }

```

9.281 ModelSimulation.h File Reference

```
#include "Event.h"
#include "Entity.h"
```

```
#include "ModelInfo.h"
#include "SimulationReporter_if.h"
Include dependency graph for ModelSimulation.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ModelSimulation](#)

9.282 ModelSimulation.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 /*
00008  * File: ModelSimulation.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 18:04
00012 */
00013
00014 #ifndef MODELSIMULATION_H
00015 #define MODELSIMULATION_H
00016
00017 //##include <chrono>
00018 #include "Event.h"
00019 #include "Entity.h"
00020 #include "ModelInfo.h"
00021 #include "SimulationReporter_if.h"
00022 //##include "Counter.h"
00023 //namespace GenesysKernel {
00024 class Model;
00025
00030 class ModelSimulation {
00031 public:
00032     ModelSimulation(Model* model);
00033     virtual ~ModelSimulation() = default;
00034 public:
00035     std::string show();
00036 public: // simulation control
00037     void start();
00038     void pause();
00039     void step();
00040     void stop();
00041 public: // old modelInfos
```

```

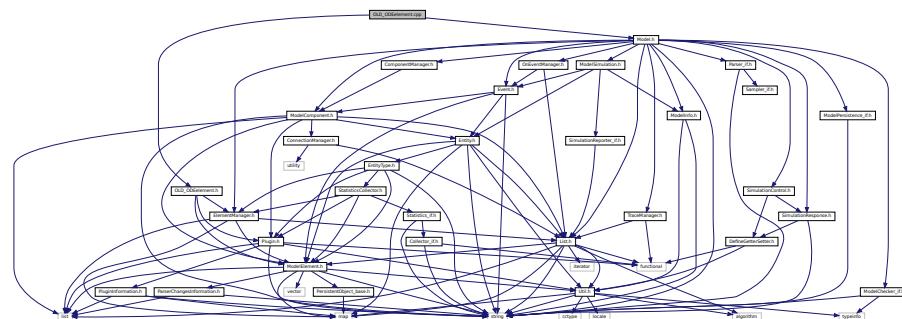
00042     void setNumberOfReplications(unsigned int _numberOfReplications);
00043     unsigned int getNumberOfReplications() const;
00044     void setReplicationLength(double _replicationLength);
00045     double getReplicationLength() const;
00046     void setReplicationLengthTimeUnit(Util::TimeUnit _replicationLengthTimeUnit);
00047     Util::TimeUnit getReplicationLengthTimeUnit() const;
00048     void setWarmUpPeriod(double _warmUpPeriod);
00049     double getWarmUpPeriod() const;
00050     void setWarmUpPeriodTimeUnit(Util::TimeUnit _warmUpPeriodTimeUnit);
00051     Util::TimeUnit getWarmUpPeriodTimeUnit() const;
00052     void setTerminatingCondition(std::string _terminatingCondition);
00053     std::string getTerminatingCondition() const;
00054 public: // gets and sets
00055     void setPauseOnEvent(bool _pauseOnEvent);
00056     bool isPauseOnEvent() const;
00057     void setStepByStep(bool _stepByStep);
00058     bool isStepByStep() const;
00059     void setInitializeStatistics(bool _initializeStatistics);
00060     bool isInitializeStatistics() const;
00061     void setInitializeSystem(bool _initializeSystem);
00062     bool isInitializeSystem() const;
00063     void setPauseOnReplication(bool _pauseBetweenReplications);
00064     bool isPauseOnReplication() const;
00065 public: // gets and sets ModelSubParts
00066     void setReporter(SimulationReporter_if* _simulationReporter);
00067     SimulationReporter_if* getReporter() const;
00068 public: // only gets
00069     double getSimulatedTime() const;
00070     bool isRunning() const;
00071     bool isPaused() const;
00072     unsigned int getCurrentReplicationNumber() const;
00073     ModelComponent* getCurrentComponent() const;
00074     Entity* getCurrentEntity() const;
00075     unsigned int getCurrentInputNumber() const;
00076     void setShowReportsAfterReplication(bool showReportsAfterReplication);
00077     bool isShowReportsAfterReplication() const;
00078     void setShowReportsAfterSimulation(bool showReportsAfterSimulation);
00079     bool isShowReportsAfterSimulation() const;
00080     List<double>* getBreakpointsOnTime() const;
00081     List<Entity*>* getBreakpointsOnEntity() const;
00082     List<ModelComponent*>* getBreakpointsOnComponent() const;
00083 public:
00084     void loadInstance(std::map<std::string, std::string>* fields);
00085     std::map<std::string, std::string>* saveInstance();
00086     /*
00087      * PRIVATE
00088     */
00089 private: // simulation control
00090     void _initSimulation();
00091     void _initReplication();
00092     void _initStatistics();
00093     void _checkWarmUpTime(Event* nextEvent);
00094     void _stepSimulation();
00095     void _replicationEnded();
00096     void _processEvent(Event* event);
00097 private:
00098     bool _checkBreakpointAt(Event* event);
00099     bool _isReplicationEndCondition();
00100    void _actualizeSimulationStatistics();
00101    void _showSimulationHeader();
00102    void _traceReplicationEnded();
00103 private:
00104    double _simulatedTime = 0.0;
00105    // \todo: list of double double _breakOnTimes;
00106    // \todo: list of modules _breakOnModules;
00107    bool _stepByStep = false;
00108    bool _pauseOnReplication = false;
00109    bool _pauseOnEvent = false;
00110    bool _initializeStatisticsBetweenReplications = true;
00111    bool _initializeSystem = true;
00112    bool _isRunning = false;
00113    bool _isPaused = false;
00114    bool _pauseRequested = false;
00115    bool _stopRequested = false;
00116    bool _simulationIsInitiated = false;
00117    bool _replicationIsInitiated = false;
00118    bool _showReportsAfterSimulation = true;
00119    bool _showReportsAfterReplication = true;
00120 private:
00121     // replication and warmup duration
00122     unsigned int _numberOfReplications = 1;
00123     double _replicationLength = 3600.0; // by default, 3600 s = 1.0 h
00124     Util::TimeUnit _replicationLengthTimeUnit = Util::TimeUnit::second;
00125     double _warmUpPeriod = 0.0;
00126     Util::TimeUnit _warmUpPeriodTimeUnit = Util::TimeUnit::second;
00127     std::string _terminatingCondition = "";
00128     bool _hasChanged = false;

```

```
00129 private:
00130     Entity* _currentEntity;
00131     ModelComponent* _currentComponent;
00132     unsigned int _currentInputNumber;
00133     unsigned int _currentReplicationNumber;
00134     //std::chrono::steady_clock::time_point _replicationStartTime;
00135     //std::chrono::steady_clock::time_point _replicationEndTime;
00136 private:
00137     const std::string _cte_stCountSimulNamePrefix = ""; //Simul.>;
00138     //std::list<ModelElement*>* _countersSimulation = new std::list<ModelElement*>();
00139 private:
00140     Model* _model;
00141     ModelInfo* _info;
00142     SimulationReporter_if* _simulationReporter;
00143     List<ModelElement*>* _statsCountersSimulation = new List<ModelElement*>();
00144     List<double>* _breakpointsOnTime = new List<double>();
00145     List<ModelComponent*>* _breakpointsOnComponent = new List<ModelComponent*>();
00146     List<Entity*>* _breakpointsOnEntity = new List<Entity*>();
00147     double _justTriggeredBreakpointsOnTime = 0.0;
00148     ModelComponent* _justTriggeredBreakpointsOnComponent = nullptr;
00149     Entity* _justTriggeredBreakpointsOnEntity = nullptr;
00150 };
00151 //namespace\\}
00152 #endif /* MODELSIMULATION_H */
```

9.283 OLD_ODElement.cpp File Reference

```
#include "OLD_ODEelement.h"
#include "Model.h"
```



9.284 OLD_ODEelement.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 13 de Junho de 2019, 19:12
00012 */
00013
00014 #include "OLD_ODEElement.h"
00015 #include "Model.h"
00016
00017 OLD_ODEElement::OLD_ODEElement(Model* model, std::string name) : ModelElement(model,
00018     Util::TypeOf<OLD_ODEElement>(), name) {
00019     //_elems = elems;
00020 }
00021 std::string OLD_ODEElement::show() {
00022     std::string txt = ModelElement::show();
00023     unsigned short i = 0;
```

```

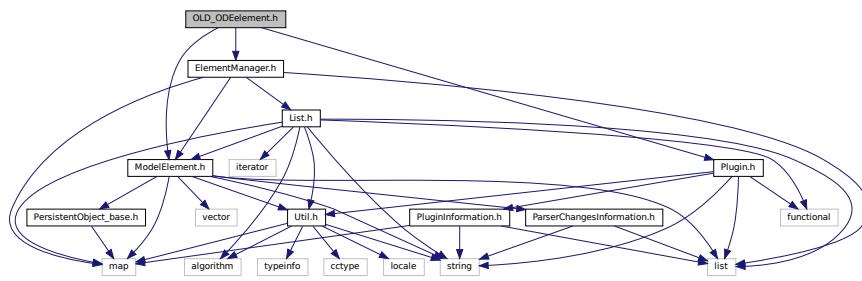
00024     for (std::list<ODEfunction*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->list()->end(); it++) {
00025         txt += ", func[" + std::to_string(i) + "]=" + (*it)->expression + "\", (func[" + std::to_string(i) + "][" + std::to_string((*it)->initialPoint) + "]=" + std::to_string((*it)->initialValue) + ")";
00026     }
00027     return txt;
00028 }
00029
00030 double OLD_ODElement::solve() {
00031     return 0.0; // dummy
00032 }
00033
00034
00035 void OLD_ODElement::setStepH(double _h) {
00036     this->_stepH = _h;
00037 }
00038
00039 double OLD_ODElement::getStepH() const {
00040     return _stepH;
00041 }
00042
00043 void OLD_ODElement::setEndTime(double _endTime) {
00044     this->_endTime = _endTime;
00045 }
00046
00047 double OLD_ODElement::getEndTime() const {
00048     return _endTime;
00049 }
00050
00051 List<ODEfunction*>* OLD_ODElement::getODEfunctions() const {
00052     return _ODEfunctions;
00053 }
00054
00055 PluginInformation* OLD_ODElement::GetPluginInformation() {
00056     PluginInformation* info = new PluginInformation(Util::TypeOf<OLD_ODElement>(),
00057     &OLD_ODElement::LoadInstance);
00058     return info;
00059 }
00060
00061 ModelElement* OLD_ODElement::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00062     OLD_ODElement* newElement = new OLD_ODElement(model);
00063     try {
00064         newElement->_loadInstance(fields);
00065     } catch (const std::exception& e) {
00066     }
00067     return newElement;
00068 }
00069
00070 bool OLD_ODElement::_loadInstance(std::map<std::string, std::string>* fields) {
00071     return ModelElement::_loadInstance(fields);
00072 }
00073
00074 std::map<std::string, std::string>* OLD_ODElement::_saveInstance() {
00075     return ModelElement::_saveInstance();
00076 }
00077
00078 bool OLD_ODElement::_check(std::string* errorMessage) {
00079     bool result = true;
00080     unsigned short i = 0;
00081     ODEFuction* func;
00082     for (std::list<ODEfunction*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->list()->end(); it++) {
00083         func = (*it);
00084         result &= _parentModel->checkExpression(func->expression, "expression[" + std::to_string(i++) + "]", errorMessage);
00085     }
00086     return result;
00087 }

```

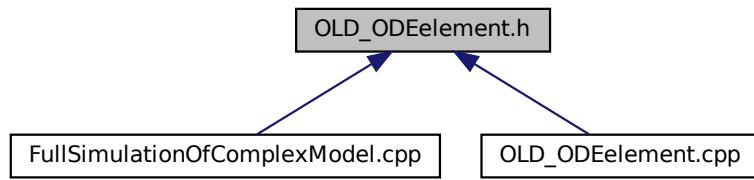
9.285 OLD_ODElement.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
```

Include dependency graph for OLD_ODEelement.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `ODEfunction`
 - class `OLD_ODEelement`

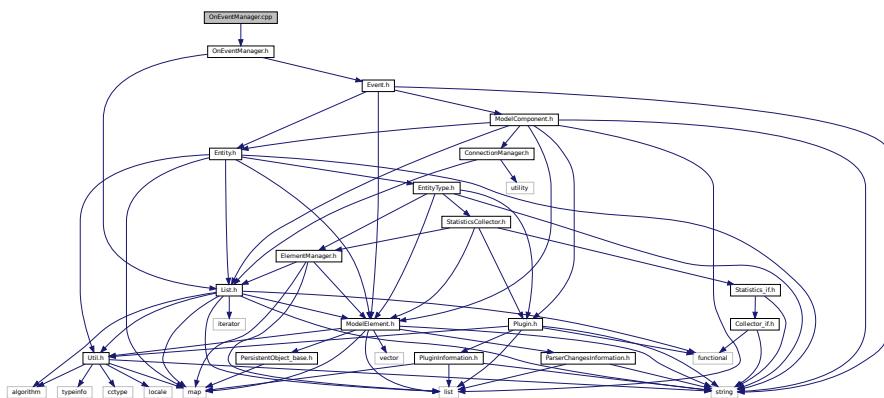
9.286 OLD ODEelement.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005  */
00006
00007 /*
00008  * File: ODE.h
00009  * Author: rlcancian
00010  *
00011  * Created on 13 de Junho de 2019, 19:12
00012  */
00013
00014 #ifndef OLD_ODEELEMENT_H
00015 #define OLD_ODEELEMENT_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020
00021 class ODEfunction {
00022 public:
00023
00024     ODEfunction(std::string expression, double initialPoint, double initialValue) {
00025         this->expression = expression;
00026         this->initialPoint = initialPoint;
00027         this->initialValue = initialValue;
00028     }
00029 }
```

```
00029     std::string expression;
00030     double initialPoint;
00031     double initialValue;
00032 };
00033
00034 class OLD_ODEelement : public ModelElement {
00035 public:
00036     OLD_ODEelement(Model* model, std::string name = "");
00037     virtual ~OLD_ODElement() = default;
00038 public:
00039     virtual std::string show();
00040 public:
00041     static PluginInformation* GetPluginInformation();
00042     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00043 public:
00044     double solve();
00045     void setStepH(double _h);
00046     double getStepH() const;
00047     void setEndTime(double _endTime);
00048     double getEndTime() const;
00049     List<ODEfunction*>* getODEfunctions() const;
00050 protected: // must be overridden by derived classes
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual std::map<std::string, std::string>* _saveInstance();
00053     virtual bool _check(std::string* errorMessage);
00054 private:
00055
00056 private:
00057     List<ODEfunction*>* _ODEfunctions = new List<ODEfunction*>();
00058     double _stepH = 0.1;
00059     double _endTime;
00060 };
00061
00062 #endif /* OLD_ODEELEMENT_H */
00063
```

9.287 OnEventManager.cpp File Reference

```
#include "OnEventManager.h"  
Include dependency graph for OnEventManager.cpp:
```



9.288 OnEventManager.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: OnEventManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 12:28
00012 */

```

```
00013
00014 #include "OnEventManager.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 OnEventManager::OnEventManager() {
00019 }
00020
00021 void OnEventManager::_addOnHandler(List<simulationEventHandler>*& list, simulationEventHandler
00022     EventHandler) {
00023     if (list->find(EventHandler) == list->list()->end())
00024         list->insert(EventHandler);
00025
00026 void OnEventManager::addOnReplicationStartHandler(simulationEventHandler EventHandler) {
00027     _addOnHandler(_onReplicationStartHandlers, EventHandler);
00028 }
00029
00030 void OnEventManager::addOnReplicationStepHandler(simulationEventHandler EventHandler) {
00031     _addOnHandler(_onReplicationStepHandlers, EventHandler);
00032 }
00033
00034 void OnEventManager::addOnProcessEventHandler(simulationEventHandler EventHandler) {
00035     _addOnHandler(_onEntityMoveHandlers, EventHandler);
00036 }
00037
00038 void OnEventManager::addOnEntityMoveHandler(simulationEventHandler EventHandler) {
00039     _addOnHandler(_onEntityMoveHandlers, EventHandler);
00040 }
00041
00042 void OnEventManager::addOnReplicationEndHandler(simulationEventHandler EventHandler) {
00043     _addOnHandler(_onReplicationEndHandlers, EventHandler);
00044 }
00045
00046 void OnEventManager::addOnSimulationStartHandler(simulationEventHandler EventHandler) {
00047     _addOnHandler(_onSimulationStartHandlers, EventHandler);
00048 }
00049
00050 void OnEventManager::addOnSimulationPausedStartHandler(simulationEventHandler EventHandler) {
00051     _addOnHandler(_onSimulationPausedStartHandlers, EventHandler);
00052 }
00053
00054 void OnEventManager::addOnSimulationEndHandler(simulationEventHandler EventHandler) {
00055     _addOnHandler(_onSimulationEndHandlers, EventHandler);
00056 }
00057
00058 void OnEventManager::addOnBreakpointHandler(simulationEventHandler EventHandler) {
00059     _addOnHandler(_onBreakpointHandlers, EventHandler);
00060 }
00061
00062 void OnEventManager::_NotifyHandlers(List<simulationEventHandler>*& list, SimulationEvent* se) {
00063     for (std::list<simulationEventHandler>::iterator it = list->list()->begin(); it != list->list()->end(); it++) {
00064         (*it)(se);
00065     }
00066 }
00067
00068 void OnEventManager::_NotifyHandlerMethods(List<simulationEventHandlerMethod>*& list, SimulationEvent*
00069     se) {
00070     for (std::list<simulationEventHandlerMethod>::iterator it = list->list()->begin(); it != list->list()->end(); it++) {
00071         (*it)(se);
00072     }
00073 }
00074 void OnEventManager::NotifyReplicationStartHandlers(SimulationEvent* se) {
00075     this->_NotifyHandlers(this->_onReplicationStartHandlers, se);
00076
00077 }
00078
00079 void OnEventManager::NotifyReplicationStepHandlers(SimulationEvent* se) {
00080     this->_NotifyHandlers(this->_onReplicationStepHandlers, se);
00081 }
00082
00083 void OnEventManager::NotifyReplicationEndHandlers(SimulationEvent* se) {
00084     this->_NotifyHandlers(this->_onReplicationEndHandlers, se);
00085 }
00086
00087 void OnEventManager::NotifyEntityMoveHandlers(SimulationEvent* se) {
00088     this->_NotifyHandlers(this->_onEntityMoveHandlers, se);
00089 }
00090
00091 void OnEventManager::NotifyProcessEventHandlers(SimulationEvent* se) {
00092     this->_NotifyHandlers(this->_onProcessEventHandlers, se);
00093     this->_NotifyHandlerMethods(this->_onProcessEventHandlerMethods, se);
00094 }
00095
```

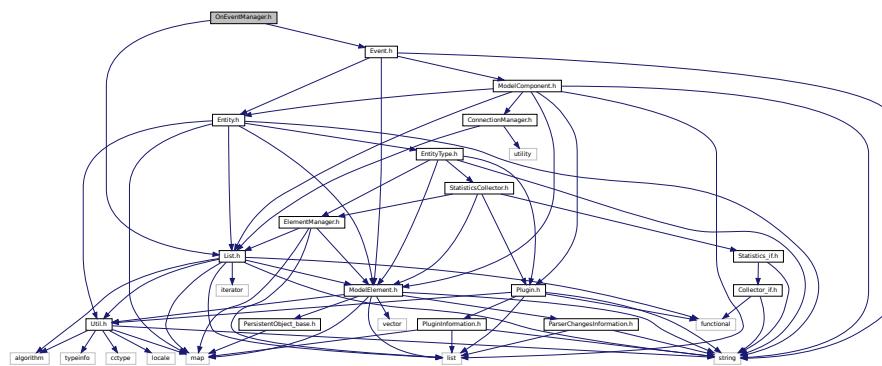
```

00096 void OnEventManager::NotifySimulationStartHandlers(SimulationEvent* se) {
00097     this->_NotifyHandlers(this->_onSimulationStartHandlers, se);
00098 }
00099
00100 void OnEventManager::NotifySimulationPausedStartHandlers(SimulationEvent* se) {
00101     this->_NotifyHandlers(this->_onSimulationPausedStartHandlers, se);
00102 }
00103
00104 void OnEventManager::NotifySimulationEndHandlers(SimulationEvent* se) {
00105     this->_NotifyHandlers(this->_onSimulationEndHandlers, se);
00106 }
00107
00108 void OnEventManager::NotifyBreakpointHandlers(SimulationEvent* se) {
00109     this->_NotifyHandlers(this->_onBreakpointHandlers, se);
00110 }

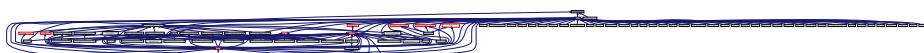
```

9.289 OnEventManager.h File Reference

```
#include "List.h"
#include "Event.h"
Include dependency graph for OnEventManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [SimulationEvent](#)
- class [OnEventManager](#)

Typedefs

- typedef void(* [simulationEventHandler](#)) ([SimulationEvent](#) *)
- typedef std::function< void([SimulationEvent](#) *) > [simulationEventHandlerMethod](#)

9.289.1 Typedef Documentation

9.289.1.1 simulationEventHandler `typedef void(* simulationEventHandler) (SimulationEvent *)`

Definition at line 49 of file [OnEventManager.h](#).

9.289.1.2 simulationEventHandlerMethod `typedef std::function<void(SimulationEvent*)> simulationEventHandlerMethod;`

Definition at line 51 of file [OnEventManager.h](#).

9.290 OnEventManager.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: OnEventManager.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 7 de Novembro de 2018, 12:28  
00012 */  
00013  
00014 #ifndef ONEVENTMANAGER_H  
00015 #define ONEVENTMANAGER_H  
00016  
00017 #include "List.h"  
00018 #include "Event.h"  
00019  
00020 //namespace GenesysKernel {  
00021     /* \todo: To implement as item (1) for DS3  
00022     * used to get and set values no matter the class (for process analyser)  
00023     * should be a wait to invoke a getter or setter no matter the class (a pointer to a member  
function without specifying the class  
00024     */  
00025     //typedef double (*memberFunctionGetDoubleVarHandler)(); //template<> ... typedef double  
(T::*getDoubleVarHandler)() or something like that  
00026     //typedef void (*memberFunctionSetDoubleVarHandler)(double);  
00027  
00028     class SimulationEvent {  
00029     public:  
00030  
00031         SimulationEvent(unsigned int replicationNumber, Event* event) {  
00032             _replicationNumber = replicationNumber;  
00033             _event = event;  
00034         }  
00035     public:  
00036  
00037         unsigned int getReplicationNumber() const {  
00038             return _replicationNumber;  
00039         }  
00040  
00041         Event* getEventProcessed() const {  
00042             return _event;  
00043         }  
00044     private:  
00045         unsigned int _replicationNumber;  
00046         Event* _event;  
00047     };  
00048  
00049     typedef void (*simulationEventHandler)(SimulationEvent*);  
00050     // for handlers that are class members (methods)  
00051     typedef std::function<void(SimulationEvent*)> simulationEventHandlerMethod;  
00052  
00053     class OnEventManager {  
00054     public:  
00055         OnEventManager();  
00060         virtual ~OnEventManager() = default;  
00061     public: // event listeners (handlers)  
00062         void addOnReplicationStartHandler(simulationEventHandler EventHandler);  
00063         void addOnReplicationStepHandler(simulationEventHandler EventHandler);  
00064         void addOnReplicationEndHandler(simulationEventHandler EventHandler);  
00065         void addOnProcessEventHandler(simulationEventHandler EventHandler);  
00066         void addOnEntityMoveHandler(simulationEventHandler EventHandler);  
00067         void addOnSimulationStartHandler(simulationEventHandler EventHandler);  
00068         void addOnSimulationPausedStartHandler(simulationEventHandler EventHandler);  
00069         void addOnSimulationEndHandler(simulationEventHandler EventHandler);
```

```

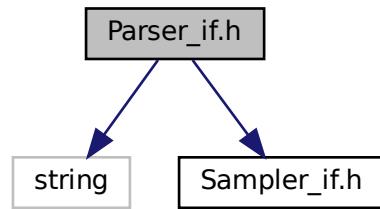
00070     void addOnEntityRemoveHandler(simulationEventHandler EventHandler);
00071     void addOnBreakpointHandler(simulationEventHandler EventHandler);
00072     // for handlers that are class members (methods)
00073     template<typename Class> void addOnProcessEventHandler(Class * object, void
00074     (Class::*function) (SimulationEvent*));
00075     // \todo: ...
00076     public:
00077     void NotifyReplicationStartHandlers(SimulationEvent* se);
00078     void NotifyReplicationStepHandlers(SimulationEvent* se);
00079     void NotifyReplicationEndHandlers(SimulationEvent* se);
00080     void NotifyProcessEventHandlers(SimulationEvent* se);
00081     void NotifyEntityMoveHandlers(SimulationEvent* se);
00082     void NotifySimulationStartHandlers(SimulationEvent* se);
00083     void NotifySimulationPausedStartHandlers(SimulationEvent* se);
00084     void NotifySimulationEndHandlers(SimulationEvent* se);
00085     void NotifyBreakpointHandlers(SimulationEvent* se);
00086     private:
00087     void _NotifyHandlers(List<simulationEventHandler>* list, SimulationEvent* se);
00088     void _NotifyHandlerMethods(List<simulationEventHandlerMethod>* list, SimulationEvent* se);
00089     void _addOnHandler(List<simulationEventHandler>* list, simulationEventHandler EventHandler);
00090     private: // events listener
00091     List<simulationEventHandler>* _onReplicationStartHandlers = new
00092     List<simulationEventHandler>();
00093     List<simulationEventHandler>* _onReplicationStepHandlers = new List<simulationEventHandler>();
00094     List<simulationEventHandler>* _onReplicationEndHandlers = new List<simulationEventHandler>();
00095     List<simulationEventHandler>* _onProcessEventHandlers = new List<simulationEventHandler>();
00096     List<simulationEventHandler>* _onEntityMoveHandlers = new List<simulationEventHandler>();
00097     List<simulationEventHandler>* _onSimulationStartHandlers = new List<simulationEventHandler>();
00098     List<simulationEventHandler>* _onSimulationPausedStartHandlers = new
00099     List<simulationEventHandler>();
00100     List<simulationEventHandler>* _onSimulationEndHandlers = new List<simulationEventHandler>();
00101     List<simulationEventHandler>* _onBreakpointHandlers = new List<simulationEventHandler>();
00102     // for handlers that are class members (methods)
00103     List<simulationEventHandlerMethod>* _onProcessEventHandlerMethods = new
00104     List<simulationEventHandlerMethod>();
00105     // \todo: ...
00106     template<typename Class> void OnEventManager::addOnProcessEventHandler(Class * object, void
00107     (Class::*function) (SimulationEvent*)) {
00108         simulationEventHandlerMethod handlerMethod = std::bind(function, object,
00109         std::placeholders::_1);
00110         // \todo: if handlerMethod already insert, should not insert it again. Problem to solve <...>
00111         for function
00112             //if (_onProcessEventHandlerMethods->find(handlerMethod) ==
00113             _onProcessEventHandlerMethods->list()->end())
00114             this->_onProcessEventHandlerMethods->insert(handlerMethod);
00115             // trying unique to solve the issue
00116             //this->_onProcessEventHandlerMethods->list()->unique(); // does not work
00117             // \todo: probably to override == operator for type simulationEventHandlerMethod
00118     // ...
00119     }
00120 //namespace\\}
00121 #endif /* ONEVENTMANAGER_H */
00122

```

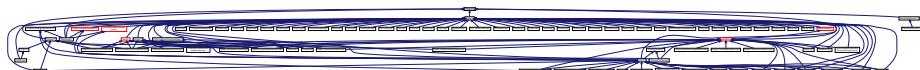
9.291 Parser_if.h File Reference

```
#include <string>
#include "Sampler_if.h"
```

Include dependency graph for Parser_if.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Parser_if](#)

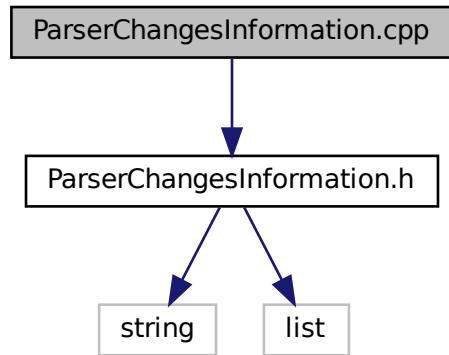
9.292 Parser_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Parser_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 23 de Agosto de 2018, 15:57
00012 */
00013
00014 #ifndef PARSER_IF_H
00015 #define PARSER_IF_H
00016
00017 #include <string>
00018 #include "Sampler_if.h"
00019
00020 class Parser_if {
00021 public:
00022     virtual double parse(const std::string expression) = 0; // may throw exception
00023     virtual double parse(const std::string expression, bool* success, std::string* errorMessage) = 0;
00024     // does not throw exceptions
00025     virtual std::string* getErrorMessage() = 0; // to get error message in the case of thrown
00026     exception
00027     virtual void setSampler(Sampler_if* sampler) = 0;
00028     virtual Sampler_if* sampler() const = 0;
00029     // ...
00030 };
00031 #endif /* PARSER_IF_H */
  
```

9.293 ParserChangesInformation.cpp File Reference

```
#include "ParserChangesInformation.h"
Include dependency graph for ParserChangesInformation.cpp:
```



9.294 ParserChangesInformation.cpp

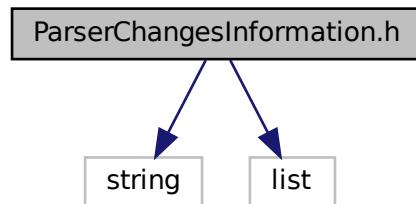
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ParserChangesInformation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:42
00012 */
00013
00014 #include "ParserChangesInformation.h"
00015
00016 ParserChangesInformation::ParserChangesInformation() {
00017 }
00018
00019 std::list<ParserChangesInformation::ParserChangesInformation::lexicalelement*>*
00020     ParserChangesInformation::getLexicalElementToRemove() const {
00021     return _lexicalElementToRemove;
00022 }
00023 std::list<ParserChangesInformation::lexicalelement*>*
00024     ParserChangesInformation::getLexicalElementToAdd() const {
00025     return _lexicalElementToAdd;
00026 }
00027 std::list<ParserChangesInformation::lexinclude*>* ParserChangesInformation::getLexIncludeToRemove()
00028 const {
00029     return _lexIncludeToRemove;
00030 }
00031 std::list<ParserChangesInformation::lexinclude*>* ParserChangesInformation::getLexIncludeToAdd() const
00032 {
00033     return _lexIncludeToAdd;
00034 }
00035 std::list<ParserChangesInformation::production*>* ParserChangesInformation::getProductionToRemove()
00036 const {
00037     return _productionToRemove;
00038 }
00039 std::list<ParserChangesInformation::production*>* ParserChangesInformation::getProductionToAdd() const
  
```

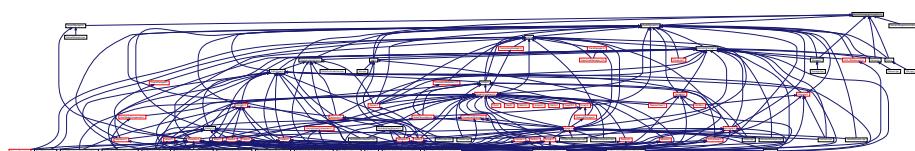
```
00040     return _productionToAdd;
00041 }
00042
00043 std::list<ParserChangesInformation::typeobj*>* ParserChangesInformation::getTypeobjToRemove() const {
00044     return _typeobjToRemove;
00045 }
00046
00047 std::list<ParserChangesInformation::typeobj*>* ParserChangesInformation::getTypeobjToAdd() const {
00048     return _typeobjToAdd;
00049 }
00050
00051 std::list<ParserChangesInformation::token*>* ParserChangesInformation::getTokensToRemove() const {
00052     return _tokensToRemove;
00053 }
00054
00055 std::list<ParserChangesInformation::token*>* ParserChangesInformation::getTokensToAdd() const {
00056     return _tokensToAdd;
00057 }
```

9.295 ParserChangesInformation.h File Reference

```
#include <string>
#include <list>
Include dependency graph for ParserChangesInformation.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ParserChangesInformation](#)

9.296 ParserChangesInformation.h

```

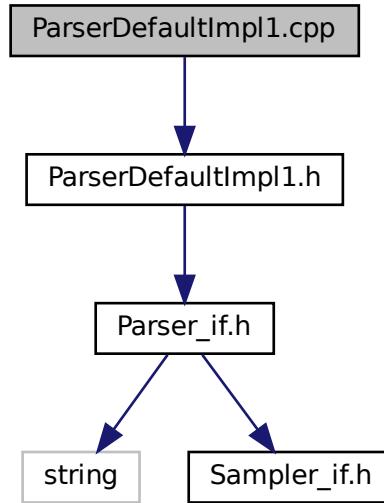
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserChangesInformation.h
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 20:42
00012 */
00013
00014 #ifndef PARSERCHANGESINFORMATION_H
00015 #define PARSERCHANGESINFORMATION_H
00016
00017 #include <string>
00018 #include <list>
00019
00020 class ParserChangesInformation {
00021 public:
00022     typedef std::string token;
00023     typedef std::string typeobj;
00024     typedef std::string lexinclude;
00025     typedef std::pair<std::string, std::string> production;
00026     typedef std::pair<std::string, std::string> lexicalement;
00027 public:
00028     ParserChangesInformation();
00029     virtual ~ParserChangesInformation() = default;
00030
00031 public: // gets and sets
00032     std::list<ParserChangesInformation::lexicalement*>* getLexicalElementToRemove() const;
00033     std::list<ParserChangesInformation::lexicalement*>* getLexicalElementToAdd() const;
00034     std::list<ParserChangesInformation::lexinclude*>* getLexIncludeToRemove() const;
00035     std::list<ParserChangesInformation::lexinclude*>* getLexIncludeToAdd() const;
00036     std::list<ParserChangesInformation::production*>* getProductionToRemove() const;
00037     std::list<ParserChangesInformation::production*>* getProductionToAdd() const;
00038     std::list<ParserChangesInformation::typeobj*>* getTypeobjToRemove() const;
00039     std::list<ParserChangesInformation::typeobj*>* getTypeobjToAdd() const;
00040     std::list<ParserChangesInformation::token*>* getTokensToRemove() const;
00041     std::list<ParserChangesInformation::token*>* getTokensToAdd() const;
00042 private:
00043     std::list<token*>* _tokensToAdd = new std::list<token*>();
00044     std::list<token*>* _tokensToRemove = new std::list<token*>();
00045     std::list<typeobj*>* _typeobjToAdd = new std::list<typeobj*>();
00046     std::list<typeobj*>* _typeobjToRemove = new std::list<typeobj*>();
00047     std::list<production*>* _productionToAdd = new std::list<production*>();
00048     std::list<production*>* _productionToRemove = new std::list<production*>();
00049     std::list<lexinclude*>* _lexIncludeToAdd = new std::list<lexinclude*>();
00050     std::list<lexinclude*>* _lexIncludeToRemove = new std::list<lexinclude*>();
00051     std::list<lexicalement*>* _lexicalElementToAdd = new std::list<lexicalement*>();
00052     std::list<lexicalement*>* _lexicalElementToRemove = new std::list<lexicalement*>();
00053
00054 };
00055
00056 #endif /* PARSERCHANGESINFORMATION_H */
00057

```

9.297 ParserDefaultImpl1.cpp File Reference

```
#include "ParserDefaultImpl1.h"
```

Include dependency graph for ParserDefaultImpl1.cpp:



9.298 ParserDefaultImpl1.cpp

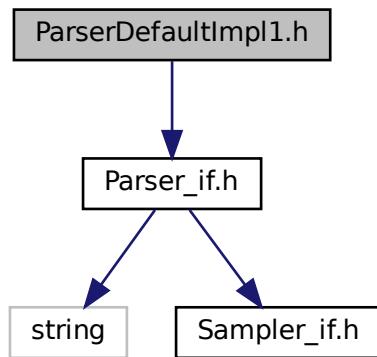
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ParserDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 01:25
00012 */
00013
00014 #include "ParserDefaultImpl1.h"
00015
00016 //namespace GenesysKernel {
00017
00018     ParserDefaultImpl1::ParserDefaultImpl1(Model* model) {
00019         _model = model;
00020     }
00021
00022     double ParserDefaultImpl1::parse(const std::string expression) { // may throw exception
00023         double result = std::atof(expression.c_str()); // change by a real parser
00024         return result;
00025     }
00026
00027     std::string* ParserDefaultImpl1::getErrorMessage() {
00028         std::string* errorMsg = new std::string();
00029         return errorMsg; /* @ \todo: */
00030     }
00031
00032     double ParserDefaultImpl1::parse(const std::string expression, bool* success, std::string*
00033     errorMessage) {
00034         try {
00035             double result = this->parse(expression);
00036             std::string temp(""); /* \todo: CHECK SCOPE OF VARIABLE */
00037             errorMessage = &temp;
00038             *success = true;
00039             return result;
00040         } catch (...) {
00041             std::string temp("Error parsing...");
00042             errorMessage = &temp;
00043             *success = false;
00044             return 0.0;
00045         }
00046     }
  
```

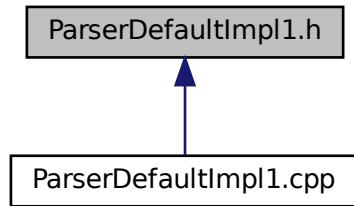
```
00044         }
00045     }
00046 //namespace\\}
00047
```

9.299 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

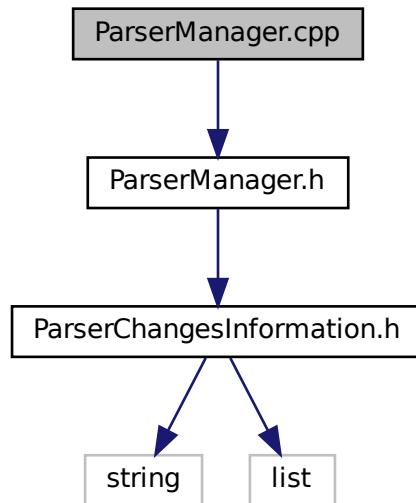
- class [ParserDefaultImpl1](#)

9.300 ParserDefaultImpl1.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 01:25
00012 */
00013
00014 #ifndef PARSERDEFAULTIMPL1_H
00015 #define PARSERDEFAULTIMPL1_H
00016
00017 #include "Parser_if.h"
00018 //namespace GenesysKernel {
00019     class Model;
00020
00021     class ParserDefaultImpl1 : public Parser_if {
00022     public:
00023         ParserDefaultImpl1(Model* model);
00024         virtual ~ParserDefaultImpl1() = default;
00025     public:
00026         double parse(const std::string expression); // may throw exception
00027         double parse(const std::string expression, bool* success, std::string* errorMessage);
00028         std::string* getErrorMessage();
00029     private:
00030         Model* _model;
00031     };
00032 //namespace\\}
00033 #endif /* PARSERDEFAULTIMPL1_H */
```

9.301 ParserManager.cpp File Reference

```
#include "ParserManager.h"  
Include dependency graph for ParserManager.cpp:
```



9.302 ParserManager.cpp

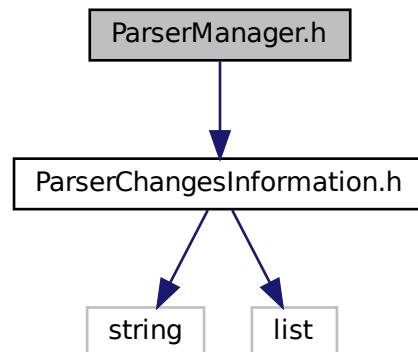
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserManager.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:34
00012 */
00013
00014 #include "ParserManager.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 ParserManager::ParserManager() {
00019 }
00020
00021

```

9.303 ParserManager.h File Reference

#include "ParserChangesInformation.h"
 Include dependency graph for ParserManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ParserManager](#)
- struct [ParserManager::NewParser](#)
- struct [ParserManager::GenerateNewParserResult](#)

9.304 ParserManager.h

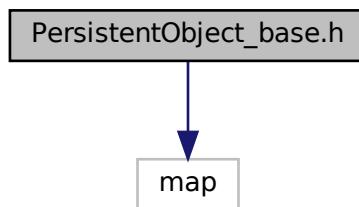
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserManager.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:34
00012 */
00013
00014 #ifndef PARSEMANAGER_H
00015 #define PARSEMANAGER_H
00016
00017 #include "ParserChangesInformation.h"
00018
00019 //namespace GenesysKernel {
00020
00021     class ParserManager {
00022     public:
00023
00024         struct NewParser {
00025             std::string bisonFilename;
00026             std::string flexFilename;
00027             std::string compiledParserFilename;
00028         };
00029
00030         struct GenerateNewParserResult {
00031             bool result;
00032             std::string bisonMessages;
00033             std::string lexMessages;
00034             std::string compilationMessages;
00035             NewParser newParser;
00036         };
00037     public:
00038         ParserManager();
00039         virtual ~ParserManager() = default;
00040     public:
00041         ParserManager::GenerateNewParserResult generateNewParser(ParserChangesInformation* changes);
00042         bool connectNewParser(ParserManager::NewParser newParser);
00043     private:
00044     };
00045 //namespace\\}
00046 #endif /* PARSEMANAGER_H */
00047

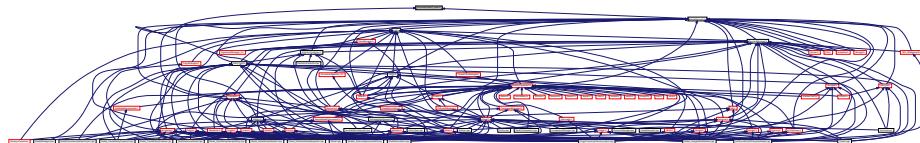
```

9.305 PersistentObject_base.h File Reference

#include <map>
Include dependency graph for PersistentObject_base.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PersistentObject_base

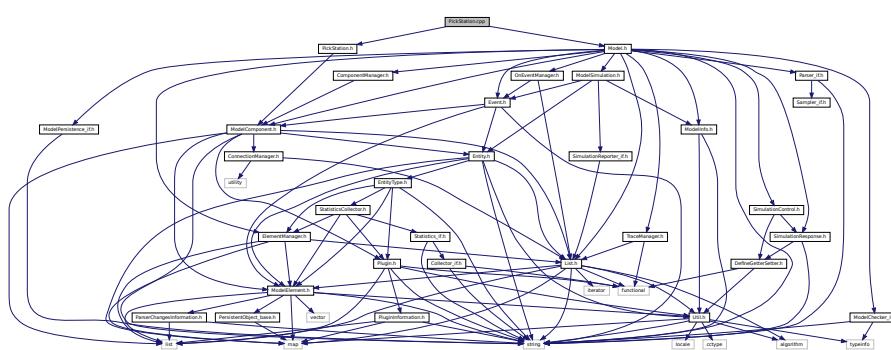
9.306 PersistentObject_base.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: persistentObject_base.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 9 de abril de 2021, 15:26  
00012 */  
00013  
00014 #ifndef PERSISTENTOBJECT_BASE_H  
00015 #define PERSISTENTOBJECT_BASE_H  
00016  
00017 #include <map>  
00018  
00019 class PersistentObject_base {  
00020 public:  
00021  
00022     PersistentObject_base() {  
00023     }  
00024     virtual ~PersistentObject_base() = default;  
00025 protected: // must be overridden by derived classes  
00026     virtual bool _loadInstance(std::map<std::string, std::string>* fields) = 0;  
00027     virtual std::map<std::string, std::string>* _saveInstance() = 0;  
00028 };  
00029  
00030 #endif /* PERSISTENTOBJECT_BASE_H */  
00031
```

9.307 PickStation.cpp File Reference

```
#include "PickStation.h"  
#include "Model.h"
```

Include dependency graph for PickStation.cpp:



9.308 PickStation.cpp

```

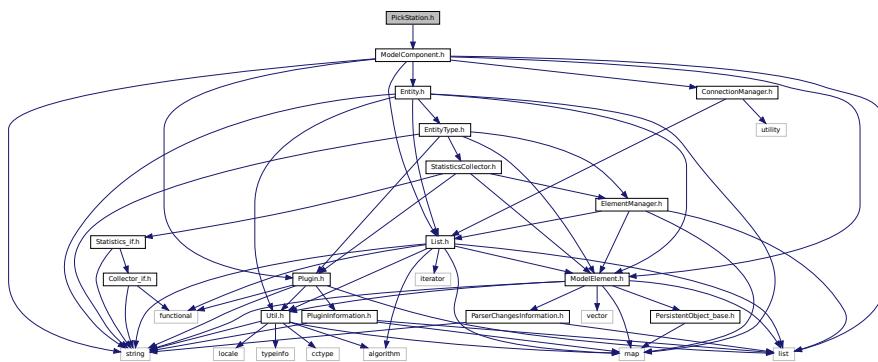
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PickStation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #include "PickStation.h"
00015
00016 #include "Model.h"
00017
00018 PickStation::PickStation(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<PickStation>(), name) {
00020
00021     std::string PickStation::show() {
00022         return ModelComponent::show() + "";
00023     }
00024
00025     ModelComponent* PickStation::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026         PickStation* newComponent = new PickStation(model);
00027         try {
00028             newComponent->_loadInstance(fields);
00029         } catch (const std::exception& e) {
00030             }
00031         return newComponent;
00032     }
00033 }
00034
00035 void PickStation::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool PickStation::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void PickStation::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* PickStation::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool PickStation::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* PickStation::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<PickStation>(),
00065         &PickStation::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

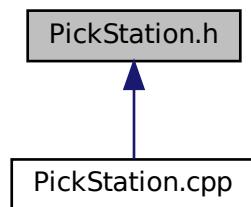
9.309 PickStation.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for PickStation.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [PickStation](#)

9.310 PickStation.h

```

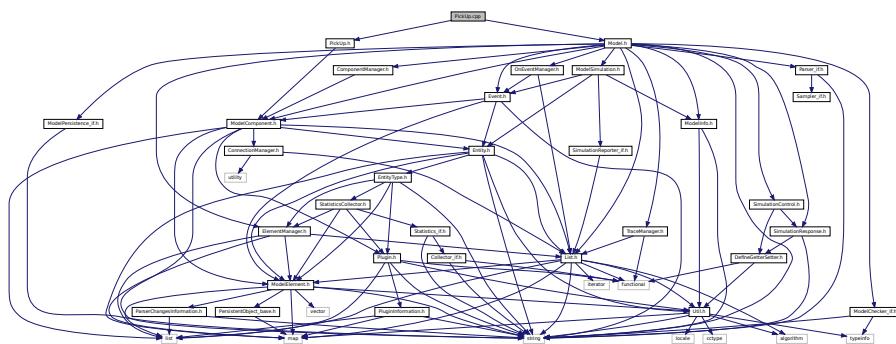
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PickStation.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #ifndef PICKSTATION_H
00015 #define PICKSTATION_H
00016
00017 #include "ModelComponent.h"
00018
00062 class PickStation : public ModelComponent {
00063 public: // constructors
00064     PickStation(Model* model, std::string name = "");
00065     virtual ~PickStation() = default;
00066 public: // virtual

```

```
00067     virtual std::string show();
00068 public: // static
00069     static PluginInformation* GetPluginInformation();
00070     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00071 protected: // virtual
00072     virtual void _execute(Entity* entity);
00073     virtual void _initBetweenReplications();
00074     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00075     virtual std::map<std::string, std::string>* _saveInstance();
00076     virtual bool _check(std::string* errorMessage);
00077 private: // methods
00078 private: // attributes 1:1
00079 private: // attributes 1:n
00080 };
00081
00082
00083 #endif /* PICKSTATION_H */
00084
```

9.311 PickUp.cpp File Reference

```
#include "PickUp.h"  
#include "Model.h"  
Include dependency graph for PickUp.cpp:
```



9.312 PickUp.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PickUp.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #include "PickUp.h"
00015
00016 #include "Model.h"
00017
00018 PickUp::PickUp(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<PickUp>(), name) {
00019 }
00020
00021 std::string PickUp::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* PickUp::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     PickUp* newComponent = new PickUp(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
```

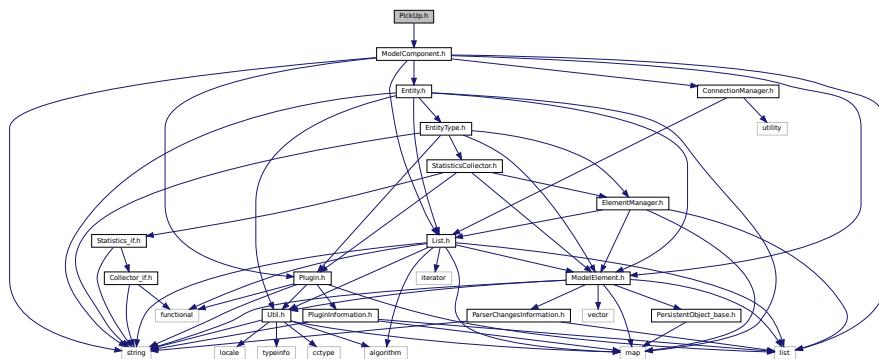
```

00032     return newComponent;
00033 }
00034
00035 void PickUp::_execute(Entity* entity) {
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00038     0.0);
00039 }
00040 bool PickUp::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void PickUp::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* PickUp::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool PickUp::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* PickUp::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<PickUp>(), &PickUp::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

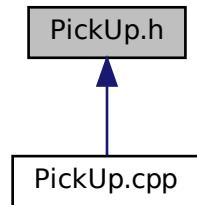
```

9.313 PickUp.h File Reference

```
#include "ModelComponent.h"
Include dependency graph for PickUp.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PickUp](#)

9.314 PickUp.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:      PickUp.h
00009 * Author:   rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #ifndef PICKUP_H
00015 #define PICKUP_H
00016
00017 #include "ModelComponent.h"
00018
00038 class PickUp : public ModelComponent {
00039 public: // constructors
00040     PickUp(Model* model, std::string name = "");
00041     virtual ~PickUp() = default;
00042 public: // virtual
00043     virtual std::string show();
00044 public: // static
00045     static PluginInformation* GetPluginInformation();
00046     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00047 protected: // virtual
00048     virtual void _execute(Entity* entity);
00049     virtual void _initBetweenReplications();
00050     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00051     virtual std::map<std::string, std::string>* _saveInstance();
00052     virtual bool _check(std::string* errorMessage);
00053 private: // methods
00054 private: // attributes 1:1
00055 private: // attributes 1:n
00056 };
00057
00058
00059 #endif /* PICKUP_H */
00060

```

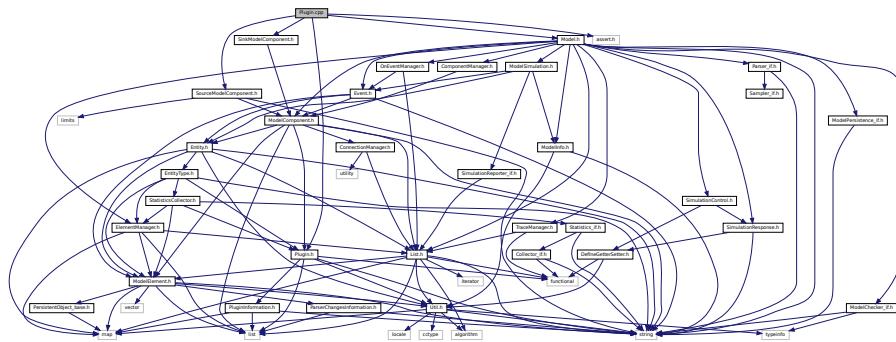
9.315 Plugin.cpp File Reference

```

#include "Plugin.h"
#include "Model.h"

```

```
#include "SourceModelComponent.h"
#include "SinkModelComponent.h"
#include <assert.h>
Include dependency graph for Plugin.cpp:
```



9.316 Plugin.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Plugin.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:58
00012 */
00013
00014 #include "Plugin.h"
00015 #include "Model.h"
00016 #include "SourceModelComponent.h"
00017 #include "SinkModelComponent.h"
00018 // #include "Assign.h"
00019 #include <assert.h>
00020
00021 //using namespace GenesysKernel;
00022
00023 Plugin::Plugin(StaticGetPluginInformation getInformation) {
00024     this->_StatMethodGetInformation = getInformation;
00025     try {
00026         PluginInformation* infos = _StatMethodGetInformation();
00027         this->_pluginInfo = infos;
00028         this->_isValidPlugin = true;
00029     } catch (...) {
00030         this->_isValidPlugin = false;
00031     }
00032 }
00033
00034 PluginInformation* Plugin::getPluginInfo() const {
00035     return _pluginInfo;
00036 }
00037
00038 bool Plugin::isValidPlugin() const {
00039     return _isValidPlugin;
00040 }
00041
00042 //Plugin::Plugin(std::string pluginTypename, bool source, bool drain) {
00043 //    //_fullfilename = fullfilename;
00044 //    _source = source;
00045 //    _drain = drain;
00046 //}
00047
00048 ModelElement* Plugin::loadNew(Model* model, std::map<std::string, std::string>* fields) {
00049     if (this->_pluginInfo->isComponent()) {
00050         return _loadNewComponent(model, fields);
00051     } else {
00052         return _loadNewElement(model, fields);
00053     }
00054 }
00055 }
```

```

00056 bool Plugin::loadAndInsertNew(Model* model, std::map<std::string, std::string>* fields) {
00057     if (this->_pluginInfo->isComponent()) {
00058         ModelComponent* newComp = _loadNewComponent(model, fields);
00059         if (newComp != nullptr) {
00060             //model->getTraceManager()->trace(newComp->show());
00061             return true; //model->components()->insert(newComp);
00062         }
00063     } else {
00064         ModelElement* newElem = _loadNewElement(model, fields);
00065         if (newElem != nullptr) {
00066             //model->getTraceManager()->trace(newElem->show());
00067             return true; //model->elements()->insert(this->_pluginInfo->pluginTypename(), newElem);
00068         }
00069     }
00070     return false;
00071 }
00072
00073 ModelComponent* Plugin::_loadNewComponent(Model* model, std::map<std::string, std::string>* fields) {
00074     //return this->_pluginInfo->loader(model, fields);
00075     StaticLoaderComponentInstance loader = this->_pluginInfo->GetComponentLoader();
00076     ModelComponent* newElementOrComponent = loader(model, fields);
00077     return newElementOrComponent;
00078 }
00079
00080 ModelElement* Plugin::_loadNewElement(Model* model, std::map<std::string, std::string>* fields) {
00081     StaticLoaderElementInstance loader = this->_pluginInfo->getElementLoader();
00082     ModelElement* newElementOrComponent = loader(model, fields);
00083     return newElementOrComponent;
00084 }

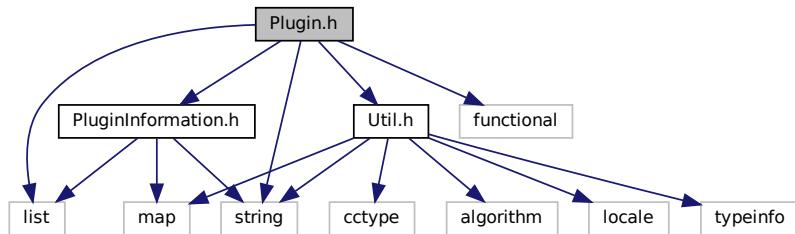
```

9.317 Plugin.h File Reference

```

#include "Util.h"
#include <string>
#include <functional>
#include <list>
#include "PluginInformation.h"
Include dependency graph for Plugin.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [Plugin](#)

9.318 Plugin.h

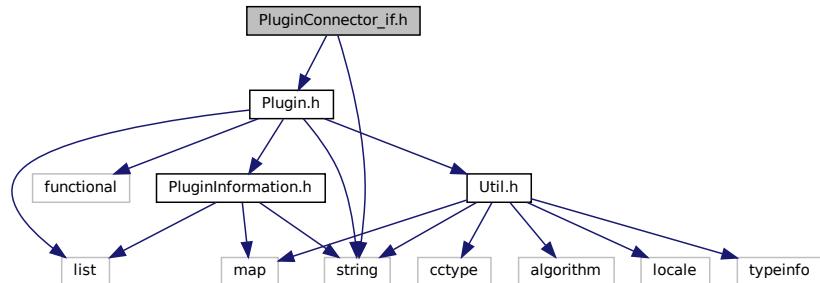
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Plugin.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:58
00012 */
00013
00014 #ifndef PLUGIN_H
00015 #define PLUGIN_H
00016
00017 #include "Util.h"
00018 #include <string>
00019 #include <functional>
00020 #include <list>
00021
00022 #include "PluginInformation.h"
00023
00024 //namespace GenesysKernel {
00025
00030     class Plugin {
00031     public:
00032         Plugin(std::string filename_so_dll);
00033         Plugin(StaticGetPluginInformation getInformation); // temporary. Just while compiled together
00034         virtual ~Plugin() = default;
00035     public:
00036         bool isValidPlugin() const;
00037         PluginInformation* getPluginInfo() const;
00038     public:
00039         ModelElement* loadNew(Model* model, std::map<std::string, std::string>* fields);
00040         bool loadAndInsertNew(Model* model, std::map<std::string, std::string>* fields);
00041     private:
00042         ModelComponent* _loadNewComponent(Model* model, std::map<std::string, std::string>* fields);
00043         ModelElement* _loadNewElement(Model* model, std::map<std::string, std::string>* fields);
00044     private: // read only
00045         bool _isValidPlugin;
00046         PluginInformation* _pluginInfo;
00047     private:
00048         StaticGetPluginInformation _StatMethodGetInformation;
00049     };
00050 //namespace\\}
00051 #endif /* PLUGIN_H */
00052

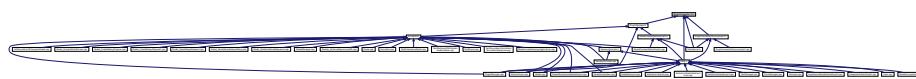
```

9.319 PluginConnector_if.h File Reference

```
#include <string>
#include "Plugin.h"
Include dependency graph for PluginConnector_if.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PluginConnector_if](#)

9.320 PluginConnector_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginConnector_if.h
00009  * Author: rlcancian
00010 *
00011  * Created on 9 de Setembro de 2019, 19:17
00012 */
00013
00014 #ifndef PLUGINCONNECTOR_IF_H
00015 #define PLUGINCONNECTOR_IF_H
00016
00017 #include<string>
00018 #include "Plugin.h"
00019
00020 //namespace GenesysKernel {
00021
00022     class PluginConnector_if {
00023     public:
00024         virtual Plugin* check(const std::string dynamicLibraryFilename) = 0;
00025         virtual Plugin* connect(const std::string dynamicLibraryFilename) = 0;
00026         virtual bool disconnect(const std::string dynamicLibraryFilename) = 0;
00027         virtual bool disconnect(Plugin* plugin) = 0;
00028     };
00029 //namespace\\}
00030 #endif /* PLUGINCONNECTOR_IF_H */
00031

```

9.321 PluginConnectorDummyImpl1.cpp File Reference

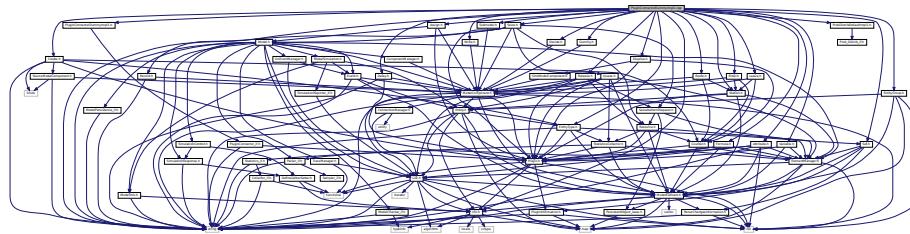
```

#include "PluginConnectorDummyImpl1.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Dummy.h"
#include "Route.h"
#include "Enter.h"
#include "Leave.h"
#include "Write.h"
#include "Submodel.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"

```

```
#include "ProbDistribDefaultImpl1.h"
#include "EntityGroup.h"
#include "Station.h"
#include "Formula.h"
#include "Set.h"
#include "Util.h"

Include dependency graph for PluginConnectorDummyImpl1.cpp:
```



9.322 PluginConnectorDummyImpl1.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PluginConnectorDummyImpl1.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 19:24
00012 */
00013
00014 #include "PluginConnectorDummyImpl1.h"
00015
00016 // Model Components
00017 #include "Create.h"
00018 #include "Delay.h"
00019 #include "Dispose.h"
00020 #include "Seize.h"
00021 #include "Release.h"
00022 #include "Assign.h"
00023 #include "Record.h"
00024 #include "Decide.h"
00025 #include "Dummy.h"
00026 #include "Route.h"
00027 #include "Enter.h"
00028 #include "Leave.h"
00029 #include "Write.h"
00030 // #include "LSODE.h"
00031 // #include "MarkovChain.h"
00032
00033 /**
00034 // #include "DropOff.h"
00035 // #include "Hold.h"
00036 // #include "Schedule.h"
00037 // #include "Batch.h"
00038 // #include "Separate.h"
00039 #include "Submodel.h"
00040 // #include "Match.h"
00041 // #include "PickUp.h"
00042 // #include "Remove.h"
00043 // #include "Search.h"
00044 // #include "Signal.h"
00045 // #include "Store.h"
00046 // #include "PickStation.h"
00047 // #include "Sequence.h"
00048 // #include "Start.h"
00049 // #include "Stop.h"
00050 // #include "Unstore.h"
00051 // #include "Failure.h"
00052 // #include "File.h"
00053 // #include "Storage.h"
00054 // #include "Access.h"
00055 // #include "Exit.h"
00056
```

```
00057 // Model elements
00058 #include "EntityType.h"
00059 #include "Attribute.h"
00060 #include "Variable.h"
00061 #include "ProbDistribDefaultImpl1.h"
00062 #include "EntityGroup.h"
00063 #include "Station.h"
00064 #include "Formula.h"
00065 #include "Set.h"
00066 // include "OLD_ODElement.h"
00067
00068 #include "Util.h"
00069
00070 //namespace GenesysKernel {
00071
00072     PluginConnectorDummyImpl1::PluginConnectorDummyImpl1() {
00073 }
00074
00075     Plugin* PluginConnectorDummyImpl1::check(const std::string dynamicLibraryFilename) {
00076         return nullptr;
00077     }
00078
00079     Plugin* PluginConnectorDummyImpl1::connect(const std::string dynamicLibraryFilename) {
00080         std::string fn = getFileName(dynamicLibraryFilename);
00081         StaticGetPluginInformation GetInfo = nullptr;
00082         Plugin* pluginResult = nullptr;
00083
00084         // model elements
00085         if (fn == "attribute.so")
00086             GetInfo = &Attribute::GetPluginInformation;
00087         else if (fn == "assign.so")
00088             GetInfo = &Assign::GetPluginInformation;
00089         else if (fn == "counter.so")
00090             GetInfo = &Counter::GetPluginInformation;
00091         else if (fn == "entitygroup.so")
00092             GetInfo = &EntityGroup::GetPluginInformation;
00093         else if (fn == "entitytype.so")
00094             GetInfo = &EntityType::GetPluginInformation;
00095         else if (fn == "formula.so")
00096             GetInfo = &Formula::GetPluginInformation;
00097         // else if (fn == "ode.so")
00098         //    GetInfo = &OLD_ODElement::GetPluginInformation;
00099         else if (fn == "queue.so")
00100             GetInfo = &Queue::GetPluginInformation;
00101         else if (fn == "resource.so")
00102             GetInfo = &Resource::GetPluginInformation;
00103         else if (fn == "set.so")
00104             GetInfo = &Set::GetPluginInformation;
00105         else if (fn == "statisticscollector.so")
00106             GetInfo = &StatisticsCollector::GetPluginInformation;
00107         else if (fn == "station.so")
00108             GetInfo = &Station::GetPluginInformation;
00109         else if (fn == "variable.so")
00110             GetInfo = &Variable::GetPluginInformation;
00111
00112         // model components
00113         else if (fn == "create.so")
00114             GetInfo = &Create::GetPluginInformation;
00115         else if (fn == "write.so")
00116             GetInfo = &Write::GetPluginInformation;
00117         else if (fn == "decide.so")
00118             GetInfo = &Decide::GetPluginInformation;
00119         else if (fn == "delay.so")
00120             GetInfo = &Delay::GetPluginInformation;
00121         else if (fn == "dispose.so")
00122             GetInfo = &Dispose::GetPluginInformation;
00123         else if (fn == "dummy.so")
00124             GetInfo = &Dummy::GetPluginInformation;
00125         else if (fn == "record.so")
00126             GetInfo = &Record::GetPluginInformation;
00127         else if (fn == "release.so")
00128             GetInfo = &Release::GetPluginInformation;
00129         else if (fn == "seize.so")
00130             GetInfo = &Seize::GetPluginInformation;
00131         else if (fn == "route.so")
00132             GetInfo = &Route::GetPluginInformation;
00133         else if (fn == "enter.so")
00134             GetInfo = &Enter::GetPluginInformation;
00135         else if (fn == "leave.so")
00136             GetInfo = &Leave::GetPluginInformation;
00137         // else if (fn == "lsode.so")
00138         //    GetInfo = &LSODE::GetPluginInformation;
00139         // else if (fn == "markovchain.so")
00140         //    GetInfo = &MarkovChain::GetPluginInformation;
00141         // else if (fn == "hold.so")
00142         //    GetInfo = &Hold::GetPluginInformation;
00143         // else if (fn == "schedule.so")
00144         //    GetInfo = &Schedule::GetPluginInformation;
00145         // else if (fn == "dropoff.so")
```

```

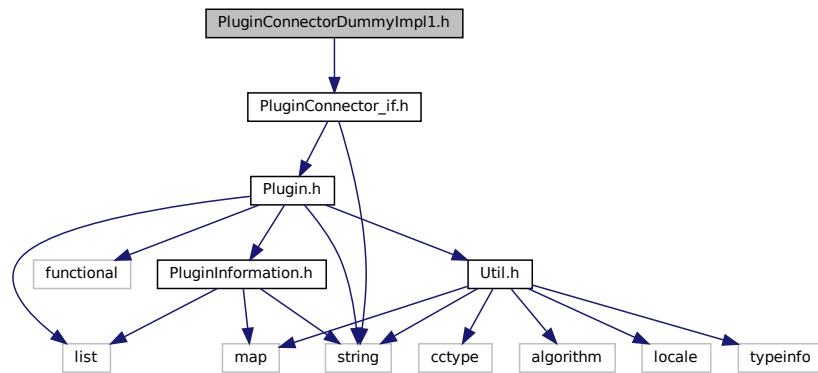
00144     //      GetInfo = &DropOff::GetPluginInformation;
00145     //  else if (fn == "batch.so")
00146     //      GetInfo = &Batch::GetPluginInformation;
00147     //  else if (fn == "separate.so")
00148     //      GetInfo = &Separate::GetPluginInformation;
00149 else if (fn == "submodel.so")
00150     GetInfo = &Submodel::GetPluginInformation;
00151 //  else if (fn == "match.so")
00152 //      GetInfo = &Match::GetPluginInformation;
00153 //  else if (fn == "pickup.so")
00154 //      GetInfo = &PickUp::GetPluginInformation;
00155 //else if (fn == "read.so")
00156 //  GetInfo = &Read::GetPluginInformation;
00157 //  else if (fn == "remove.so")
00158 //      GetInfo = &Remove::GetPluginInformation;
00159 //  else if (fn == "search.so")
00160 //      GetInfo = &Search::GetPluginInformation;
00161 //  else if (fn == "signal.so")
00162 //      GetInfo = &Signal::GetPluginInformation;
00163 //  else if (fn == "store.so")
00164 //      GetInfo = &Store::GetPluginInformation;
00165 //  else if (fn == "unstore.so")
00166 //      GetInfo = &Unstore::GetPluginInformation;
00167 //else if (fn == "expression.so")
00168 //  GetInfo = &Expression::GetPluginInformation;
00169 //  else if (fn == "failure.so")
00170 //      GetInfo = &Failure::GetPluginInformation;
00171 //  else if (fn == "file.so")
00172 //      GetInfo = &File::GetPluginInformation;
00173 //  else if (fn == "storage.so")
00174 //      GetInfo = &Storage::GetPluginInformation;
00175 //  else if (fn == "pickstation.so")
00176 //      GetInfo = &PickStation::GetPluginInformation;
00177 //  else if (fn == "sequence.so")
00178 //      GetInfo = &Sequence::GetPluginInformation;
00179 //  else if (fn == "access.so")
00180 //      GetInfo = &Access::GetPluginInformation;
00181 //  else if (fn == "exit.so")
00182 //      GetInfo = &Exit::GetPluginInformation;
00183 //  else if (fn == "start.so")
00184 //      GetInfo = &Start::GetPluginInformation;
00185 //  else if (fn == "stop.so")
00186 //      GetInfo = &Stop::GetPluginInformation;
00187 //else if (fn == "conveyour.so")
00188 //  GetInfo = &Conveyour::GetPluginInformation;
00189 //else if (fn == "segment.so")
00190 //  GetInfo = &Segment::GetPluginInformation;
00191
00192 //else if (fn=="")
00193
00194     if (GetInfo != nullptr) {
00195         pluginResult = new Plugin(GetInfo);
00196     }
00197     return pluginResult;
00198 }
00199
00200 bool PluginConnectorDummyImpl1::disconnect(const std::string dynamicLibraryFilename) {
00201     return true;
00202 }
00203
00204 bool PluginConnectorDummyImpl1::disconnect(Plugin* plugin) {
00205     return true;
00206 }
00207
00208 //namespace\\

```

9.323 PluginConnectorDummyImpl1.h File Reference

```
#include "PluginConnector_if.h"
```

Include dependency graph for PluginConnectorDummyImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [PluginConnectorDummyImpl1](#)

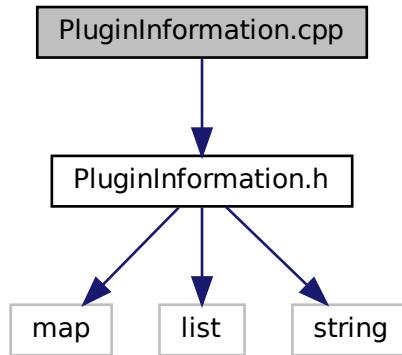
9.324 PluginConnectorDummyImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginConnectorDummyImpl1.h
00009  * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 19:24
00012 */
00013
00014 #ifndef PLUGINCONNECTORDUMMYIMPL1_H
00015 #define PLUGINCONNECTORDUMMYIMPL1_H
00016
00017 #include "PluginConnector_if.h"
00018 //namespace GenesysKernel {
00019
00020     class PluginConnectorDummyImpl1 : public PluginConnector_if {
00021     public:
00022         PluginConnectorDummyImpl1();
00023         virtual ~PluginConnectorDummyImpl1() = default;
00024     public:
00025         virtual Plugin* check(const std::string dynamicLibraryFilename);
00026         virtual Plugin* connect(const std::string dynamicLibraryFilename);
00027         virtual bool disconnect(const std::string dynamicLibraryFilename);
00028         virtual bool disconnect(Plugin* plugin);
00029     private:
00030     };
00031 }
00032 //namespace\\}
00033 #endif /* PLUGINCONNECTORDUMMYIMPL1_H */
00034
  
```

9.325 PluginInformation.cpp File Reference

```
#include "PluginInformation.h"
Include dependency graph for PluginInformation.cpp:
```



9.326 PluginInformation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PluginInformation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 20:02
00012 */
00013
00014 #include "PluginInformation.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 PluginInformation::PluginInformation(std::string pluginTypename, StaticLoaderComponentInstance
00019     componentloader) {
00020     this->_componentloader = componentloader;
00021     this->elementloader = nullptr;
00022     this->_isComponent = true;
00023     this->_pluginTypename = pluginTypename;
00024 }
00025 PluginInformation::PluginInformation(std::string pluginTypename, StaticLoaderElementInstance
00026     elementloader) {
00027     this->_componentloader = nullptr;
00028     this->elementloader = elementloader;
00029     this->_isComponent = false;
00030     this->_pluginTypename = pluginTypename;
00031 }
00032 StaticLoaderElementInstance PluginInformation::getElementLoader() const {
00033     return _elementloader;
00034 }
00035
00036 StaticLoaderComponentInstance PluginInformation::GetComponentLoader() const {
00037     return _componentloader;
00038 }
00039
00040 bool PluginInformation::isGenerateReport() const {
00041     return _generateReport;
00042 }
00043
  
```

```
00044 bool PluginInformation::isComponent() const {
00045     return _isComponent;
00046 }
00047
00048 bool PluginInformation::isSendTransfer() const {
00049     return _sendTransfer;
00050 }
00051
00052 bool PluginInformation::isReceiveTransfer() const {
00053     return _receiveTransfer;
00054 }
00055
00056 bool PluginInformation::isSink() const {
00057     return _isSink;
00058 }
00059
00060 bool PluginInformation::isSource() const {
00061     return _isSource;
00062 }
00063
00064 std::string PluginInformation::getObservation() const {
00065     return _observation;
00066 }
00067
00068 std::string PluginInformation::getVersion() const {
00069     return _version;
00070 }
00071
00072 std::string PluginInformation::getDate() const {
00073     return _date;
00074 }
00075
00076 std::string PluginInformation::getAuthor() const {
00077     return _author;
00078 }
00079
00080 std::string PluginInformation::getPluginTypename() const {
00081     return _pluginTypename;
00082 }
00083
00084 void PluginInformation::insertDynamicLibFileDependence(std::string filename) {
00085     _dynamicLibFilenameDependencies->insert(_dynamicLibFilenameDependencies->end(), filename);
00086 }
00087
00088 void PluginInformation::setDynamicLibFilenameDependencies(std::list<std::string>*
00089     dynamicLibFilenameDependencies) {
00090     this->_dynamicLibFilenameDependencies = dynamicLibFilenameDependencies;
00091 }
00092 std::list<std::string>* PluginInformation::getDynamicLibFilenameDependencies() const {
00093     return _dynamicLibFilenameDependencies;
00094 }
00095
00096 void PluginInformation::setGenerateReport(bool generateReport) {
00097     this->_generateReport = generateReport;
00098 }
00099
00100 void PluginInformation::setSendTransfer(bool sendTransfer) {
00101     this->_sendTransfer = sendTransfer;
00102 }
00103
00104 void PluginInformation::setReceiveTransfer(bool receiveTransfer) {
00105     this->_receiveTransfer = receiveTransfer;
00106 }
00107
00108 void PluginInformation::setSink(bool Sink) {
00109     _isSink = Sink;
00110 }
00111
00112 void PluginInformation::setSource(bool Source) {
00113     _isSource = Source;
00114 }
00115
00116 void PluginInformation::setObservation(std::string observation) {
00117     this->_observation = observation;
00118 }
00119
00120 void PluginInformation::setVersion(std::string version) {
00121     this->_version = version;
00122 }
00123
00124 void PluginInformation:: setDate(std::string date) {
00125     this->_date = date;
00126 }
00127
00128 void PluginInformation::setAuthor(std::string author) {
00129     this->_author = author;
```

```

00130 }
00131
00132 void PluginInformation::setMaximumOutputs(unsigned short _maximumOutputs) {
00133     this->_maximumOutputs = _maximumOutputs;
00134 }
00135
00136 unsigned short PluginInformation::getMaximumOutputs() const {
00137     return _maximumOutputs;
00138 }
00139
00140 void PluginInformation::setMinimumOutputs(unsigned short _minimumOutputs) {
00141     this->_minimumOutputs = _minimumOutputs;
00142 }
00143
00144 unsigned short PluginInformation::getMinimumOutputs() const {
00145     return _minimumOutputs;
00146 }
00147
00148 void PluginInformation::setMaximumInputs(unsigned short _maximumInputs) {
00149     this->_maximumInputs = _maximumInputs;
00150 }
00151
00152 unsigned short PluginInformation::getMaximumInputs() const {
00153     return _maximumInputs;
00154 }
00155
00156 void PluginInformation::setMinimumInputs(unsigned short _minimumInputs) {
00157     this->_minimumInputs = _minimumInputs;
00158 }
00159
00160 unsigned short PluginInformation::getMinimumInputs() const {
00161     return _minimumInputs;
00162 }
00163
00164

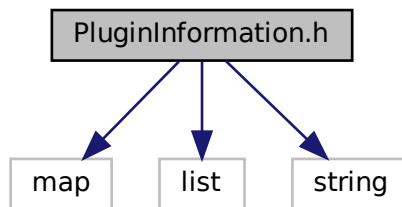
```

9.327 PluginInformation.h File Reference

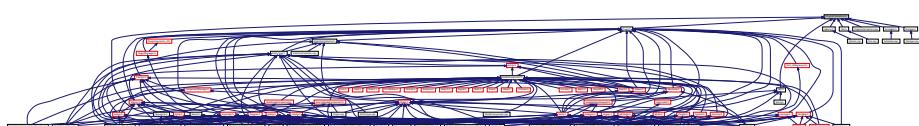
```

#include <map>
#include <list>
#include <string>
Include dependency graph for PluginInformation.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [PluginInformation](#)

Typedefs

- [typedef ModelComponent *\(* StaticLoaderComponentInstance\) \(Model *, std::map< std::string, std::string > *\)](#)
- [typedef ModelElement *\(* StaticLoaderElementInstance\) \(Model *, std::map< std::string, std::string > *\)](#)
- [typedef PluginInformation *\(* StaticGetPluginInformation\) \(\)](#)

9.327.1 Typedef Documentation

9.327.1.1 [StaticGetPluginInformation](#) `typedef PluginInformation*(* StaticGetPluginInformation) ()`

Definition at line 31 of file [PluginInformation.h](#).

9.327.1.2 [StaticLoaderComponentInstance](#) `typedef ModelComponent*(* StaticLoaderComponentInstance) (Model *, std::map< std::string, std::string > *)`

Definition at line 28 of file [PluginInformation.h](#).

9.327.1.3 [StaticLoaderElementInstance](#) `typedef ModelElement*(* StaticLoaderElementInstance) (Model *, std::map< std::string, std::string > *)`

Definition at line 29 of file [PluginInformation.h](#).

9.328 PluginInformation.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  PluginInformation.h
00009  * Author: rlcancian
00010 *
00011  * Created on 9 de Setembro de 2019, 20:02
00012 */
00013
00014 #ifndef PLUGININFORMATION_H
00015 #define PLUGININFORMATION_H
00016
00017 #include <map>
00018 #include <list>
00019 #include <string>
00020
00021 //namespace GenesysKernel {
00022     class ModelElement;
00023     class ModelComponent;
00024     class Model;
```

```

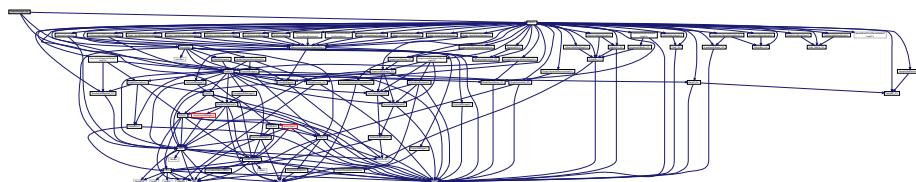
00025     class ElementManager;
00026
00027     // \todo: the following 2 types were different but now on they are the same and should be merged
00028     typedef ModelComponent* (*StaticLoaderComponentInstance)(Model*, std::map<std::string,
00029         std::string>*);
00030     typedef ModelElement* (*StaticLoaderElementInstance)(Model*, std::map<std::string, std::string>*);
00031     class PluginInformation;
00032     typedef PluginInformation* (*StaticGetPluginInformation)();
00033
00034     class PluginInformation {
00035     public:
00036         PluginInformation(std::string pluginTypename, StaticLoaderComponentInstance componentloader);
00037         PluginInformation(std::string pluginTypename, StaticLoaderElementInstance elementloader);
00038     public:
00039         // gets
00040         StaticLoaderElementInstance getElementLoader() const;
00041         StaticLoaderComponentInstance GetComponentLoader() const;
00042         bool isGenerateReport() const;
00043         bool isComponent() const;
00044         bool isSendTransfer() const;
00045         bool isReceiveTransfer() const;
00046         bool isSink() const;
00047         bool isSource() const;
00048         std::string getObservation() const;
00049         std::string getVersion() const;
00050         std::string getDate() const;
00051         std::string getAuthor() const;
00052         std::string getPluginTypename() const;
00053         // sets
00054         void insertDynamicLibFileDependence(std::string filename);
00055         void setDynamicLibFilenameDependencies(std::list<std::string>*
00056             dynamicLibFilenameDependencies);
00057             std::list<std::string>* getDynamicLibFilenameDependencies() const;
00058             void setGenerateReport(bool generateReport);
00059             void setSendTransfer(bool sendTransfer);
00060             void setReceiveTransfer(bool receiveTransfer);
00061             void setSink(bool Sink);
00062             void setSource(bool Source);
00063             void setObservation(std::string observation);
00064             void setVersion(std::string version);
00065             void setDate(std::string date);
00066             void setAuthor(std::string author);
00067             void setMaximumOutputs(unsigned short _maximumOutputs);
00068             unsigned short getMaximumOutputs() const;
00069             void setMinimumOutputs(unsigned short _minimumOutputs);
00070             unsigned short getMinimumOutputs() const;
00071             void setMaximumInputs(unsigned short _maximumInputs);
00072             unsigned short getMaximumInputs() const;
00073             void setMinimumInputs(unsigned short _minimumInputs);
00074             unsigned short getMinimumInputs() const;
00075     public:
00076     private:
00077         std::string _author = "prof. Dr. Ing. Rafael Luiz Cancian";
00078         std::string _date = "01/07/2018";
00079         std::string _version = "0.9.1";
00080         std::string _observation = "First implementation not fully completed nor tested. Use with
00081         caution.";
00082         bool _isSource = false;
00083         bool _isSink = false;
00084         bool _receiveTransfer = false;
00085         bool _sendTransfer = false;
00086         bool _generateReport = false;
00087         std::list<std::string>* _dynamicLibFilenameDependencies = new std::list<std::string>();
00088         // set from constructor
00089         std::string _pluginTypename;
00090         bool _isComponent = false;
00091         unsigned short _minimumInputs = 1;
00092         unsigned short _maximumInputs = 1;
00093         unsigned short _minimumOutputs = 1;
00094         unsigned short _maximumOutputs = 1;
00095         StaticLoaderComponentInstance _componentloader;
00096         StaticLoaderElementInstance _elementloader;
00097     };
00098 //namespace\\}
00099
00100 #endif /* PLUGININFORMATION_H */
00101

```

9.329 PluginManager.cpp File Reference

```
#include "PluginManager.h"
#include "Simulator.h"
```

```
#include "Traits.h"
Include dependency graph for PluginManager.cpp:
```



9.330 PluginManager.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginManager.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Maio de 2019, 17:49
00012 */
00013
00014 #include "PluginManager.h"
00015 #include "Simulator.h"
00016 #include "Traits.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 PluginManager::PluginManager(Simulator* simulator) {
00021     _simulator = simulator;
00022     this->_pluginConnector = new Traits<PluginConnector_if>::Implementation();
00023 }
00024
00025
00026 //bool PluginManager::check(Plugin* plugin){
00027 //    return true;
00028 //}
00029
00030 bool PluginManager::_insert(Plugin* plugin) {
00031     PluginInformation *plugInfo = plugin->getPluginInfo();
00032     if (plugin->isIsValidPlugin() && plugInfo != nullptr) {
00033         std::string msg = "Inserting ";
00034         if (plugInfo->isComponent())
00035             msg += "component";
00036         else
00037             msg += "element";
00038         msg += " plugin \"" + plugin->getPluginInfo()->getPluginTypename() + "\"";
00039         _simulator->getTracer()->trace(Util::TraceLevel::simulatorDetailed, msg);
00040         // insert all dependencies before to insert this plugin
00041         bool allDependenciesInserted = true;
00042         if (plugInfo->getDynamicLibFilenameDependencies()->size() > 0) {
00043             Util::IncIndent();
00044             {
00045                 _simulator->getTracer()->trace(Util::TraceLevel::simulatorDetailed, "Inserting
dependencies...");
00046                 Util::IncIndent();
00047                 {
00048                     for (std::list<std::string>::iterator it =
00049                         plugInfo->getDynamicLibFilenameDependencies()->begin(); it !=
00050                         plugInfo->getDynamicLibFilenameDependencies()->end(); it++) {
00051                         allDependenciesInserted &= (this->insert((*it)) != nullptr);
00052                     }
00053                     Util::DecIndent();
00054                 }
00055             }
00056             if (!allDependenciesInserted) {
00057                 _simulator->getTracer()->trace(Util::TraceLevel::errorRecover, "Plugin dependencies could
not be inserted; therefore, plugin will not be inserted");
00058                 return false;
00059             }
00060             if (this->find(plugInfo->getPluginTypename()) != nullptr) { // plugin already exists
00061                 Util::IncIndent();
00062                 _simulator->getTracer()->trace(Util::TraceLevel::simulatorDetailed, "Plugin already exists
and was not inserted again");
00063             }
00064         }
00065     }
00066 }
```

```

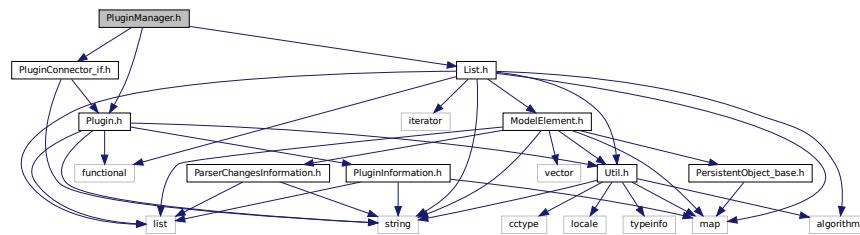
00063         Util::DecIndent();
00064         return false;
00065     }
00066     _plugins->insert(plugin);
00067     Util::IncIndent();
00068     this->_simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin successfully
00069     inserted");
00070     Util::DecIndent();
00071 } else {
00072     Util::IncIndent();
00073     this->_simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin could not be
00074     inserted");
00075     plugin->~Plugin(); // destroy the invalid plugin
00076     Util::DecIndent();
00077     return false;
00078 }
00079
00080 bool PluginManager::check(std::string dynamicLibraryFilename) {
00081     Plugin* plugin;
00082     try {
00083         plugin = _pluginConnector->check(dynamicLibraryFilename);
00084     } catch (...) {
00085         return false;
00086     }
00087     return (plugin != nullptr);
00088 }
00089
00090 Plugin* PluginManager::insert(std::string dynamicLibraryFilename) {
00091     Plugin* plugin;
00092     try {
00093         plugin = _pluginConnector->connect(dynamicLibraryFilename);
00094         if (plugin != nullptr)
00095             _insert(plugin);
00096         else {
00097             _simulator->getTracer()->trace(Util::TraceLevel::errorRecover, "Plugin from file \\" + 
dynamicLibraryFilename + "\\ could not be loaded.");
00098         }
00099     } catch (...) {
00100         return nullptr;
00101     }
00102     return plugin;
00103 }
00104
00105
00106 bool PluginManager::remove(std::string dynamicLibraryFilename) {
00107
00108     Plugin* pi = this->find(dynamicLibraryFilename);
00109     return remove(pi);
00110 }
00111
00112 bool PluginManager::remove(Plugin* plugin) {
00113     if (plugin != nullptr) {
00114         _plugins->remove(plugin);
00115         try {
00116             _pluginConnector->disconnect(plugin);
00117         } catch (...) {
00118             return false;
00119         }
00120         _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin successfully
00121     removed");
00122         return true;
00123     }
00124     _simulator->getTracer()->trace(Util::TraceLevel::simulatorResult, "Plugin could not be removed");
00125     return false;
00126 }
00127
00128 Plugin* PluginManager::find(std::string pluginTypeName) {
00129     for (std::list<Plugin*>::iterator it = this->_plugins->list()->begin(); it != 
00130     _plugins->list()->end(); it++) {
00131         if ((*it)->getPluginInfo()->getPluginTypename() == pluginTypeName) {
00132             return (*it);
00133         }
00134     }
00135     return nullptr;
00136 }
00137
00138 Plugin* PluginManager::front() {
00139     return this->_plugins->front();
00140 }
00141
00142 Plugin* PluginManager::next() {
00143     return _plugins->next();
00144 }
```

```

00145 }
00146
00147 Plugin* PluginManager::last() {
00148     return this->_plugins->last();
00149 }
```

9.331 PluginManager.h File Reference

```
#include "List.h"
#include "Plugin.h"
#include "PluginConnector_if.h"
Include dependency graph for PluginManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PluginManager](#)

9.332 PluginManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 30 de Maio de 2019, 17:49
00012 */
00013
00014 #ifndef PLUGINMANAGER_H
00015 #define PLUGINMANAGER_H
00016
00017 #include "List.h"
00018 #include "Plugin.h"
00019 #include "PluginConnector_if.h"
00020
00021 //namespace GenesysKernel {
00022     class Simulator;
00023
00024     class PluginManager {
00025     public:
00026         PluginManager(Simulator* simulator);
```

```

00027     virtual ~PluginManager() = default;
00028 public:
00029     bool check(const std::string dynamicLibraryFilename);
00030     Plugin* insert(const std::string dynamicLibraryFilename);
00031     bool remove(const std::string dynamicLibraryFilename);
00032     bool remove(Plugin* plugin);
00033     Plugin* find(std::string pluginTypeName);
00034 private:
00035     Plugin* front();
00036     Plugin* next();
00037     Plugin* last();
00038 private:
00039     bool _insert(Plugin* plugin);
00040 private:
00041     List<Plugin*>* _plugins = new List<Plugin*>();
00042     Simulator* _simulator;
00043     PluginConnector_if* _pluginConnector;
00044 };
00045 //namespace\\\
00046 #endif /* PLUGINMANAGER_H */
00047

```

9.333 Prob_Distrib_if.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [ProbDistrib_if](#)

9.334 Prob_Distrib_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Prob_Distrib_if.h
00009  * Author: rlcancian
00010  *
00011  * Created on 15 de setembro de 2020, 17:58
00012 */
00013
00014 #ifndef PROB_DISTRIB_IF_H
00015 #define PROB_DISTRIB_IF_H
00016
00017 class ProbDistrib_if {
00018 public:
00019     virtual double beta(double x, double alpha, double beta) = 0;
00020     virtual double chi2(double x, double m) = 0;
00021     virtual double erlang(double x, double shape, double scale) = 0; // int M
00022     virtual double exponential(double x, double mean) = 0;
00023     virtual double fFisher(double x, double k, double m) = 0;
00024     virtual double gamma(double x, double shape, double scale) = 0;
00025     virtual double logNormal(double x, double mean, double stddev) = 0;
00026     virtual double normal(double x, double mean, double stddev) = 0;
00027     virtual double poisson(double x, double mean) = 0;
00028     virtual double triangular(double x, double min, double mode, double max) = 0;
00029     virtual double tStudent(double x, double mean, double stddev, unsigned int degreeFreedom) = 0;
00030     virtual double uniform(double x, double min, double max) = 0;
00031     virtual double weibull(double x, double shape, double scale) = 0;
00032     // inverse
00033     virtual double inverseChi2(double cumulativeProbability, double m) = 0;
00034     virtual double inverseFisher(double cumulativeProbability, double k, double m) = 0;
00035     virtual double inverseNormal(double cumulativeProbability, double mean, double stddev) = 0;

```

```

00036     virtual double inverseTStudent(double cumulativeProbability, double mean, double stddev, double
00037         degreeFreedom) = 0;
00038     // ... other inverses ...
00039
00040 #endif /* PROB_DISTRIB_IF_H */
00041

```

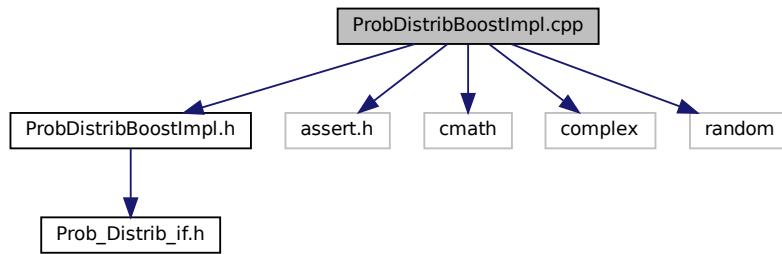
9.335 ProbDistribBoostImpl.cpp File Reference

```

#include "ProbDistribBoostImpl.h"
#include <assert.h>
#include <cmath>
#include <complex>
#include <random>

```

Include dependency graph for ProbDistribBoostImpl.cpp:



9.336 ProbDistribBoostImpl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ProbDistrib.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 17:25
00012 */
00013
00014 #include "ProbDistribBoostImpl.h"
00015 #include <assert.h>
00016 #include <cmath>
00017 #include <complex>
00018 #include <random>
00019
00020 //##include <boost/math/distributions.hpp>
00021
00022 //using namespace boost::math;
00023
00024 double ProbDistribBoostImpl::uniform(double x, double min, double max) {
00025     if (x >= min && x <= max)
00026         return 1.0 / (max - min);
00027     else
00028         return 0.0;
00029 }
00030
00031 double ProbDistribBoostImpl::exponential(double x, double mean) {
00032     assert(x >= 0);
00033     return 0.0; //mean * exp(-mean * x);
00034 }
00035
00036 double ProbDistribBoostImpl::erlang(double x, double shape, double scale) { //

```

```

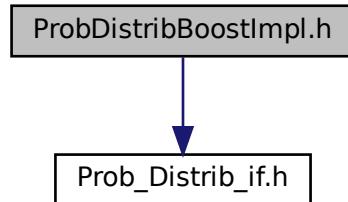
00037     return ProbDistribBoostImpl::gamma(x, shape, scale); // \todo: NOT EXACTLY THIS... REDO
00038 }
00039
00040 double ProbDistribBoostImpl::normal(double x, double mean, double stddev) {
00041     double p1 = 1 / (stddev * std::sqrt(2 * M_PI));
00042     double rdf = (x - mean) / stddev;
00043     double p2 = std::exp(-0.5 * rdf * rdf);
00044     return p1*p2;
00045 }
00046
00047 double ProbDistribBoostImpl::gamma(double x, double shape, double scale) {
00048     // gamma_distribution<> my_gamma(shape, scale);
00049     return 0.0; // pdf(my_gamma, x);
00050 }
00051
00052 double ProbDistribBoostImpl::beta(double x, double alpha, double beta) {
00053     // beta_distribution<> my_beta(alpha, beta);
00054     return 0.0; // pdf(my_beta, x);
00055 }
00056
00057 double ProbDistribBoostImpl::weibull(double x, double shape, double scale) {
00058     // weibull_distribution<> my_weibull(shape, scale);
00059     return 0.0; // pdf(my_weibull, x);
00060 }
00061
00062 double ProbDistribBoostImpl::logNormal(double x, double mean, double stddev) {
00063     // lognormal_distribution<> my_lognormal(mean, stddev);
00064     return 0.0; // pdf(my_lognormal, x);
00065 }
00066
00067 double ProbDistribBoostImpl::triangular(double x, double min, double mode, double max) {
00068     // triangular_distribution<> my_triangular(min, mode, max);
00069     return 0.0; // pdf(my_triangular, x);
00070 }
00071
00072 double ProbDistribBoostImpl::tStudent(double x, double mean, double stddev, unsigned int
    degreeFreedom) {
00073     // students_t_distribution<> my_students_t(degreeFreedom);
00074     return 0.0; // pdf(my_students_t, x)*mean + stddev; // t-student SEEKS to be based on NORM(0,1)
00075 }
00076
00077 double ProbDistribBoostImpl::fFisher(double x, double k, double m) {
00078     // fisher_f_distribution<> my_fisher_f(k, m);
00079     return 0.0; // pdf(my_fisher_f, x);
00080 }
00081
00082 double ProbDistribBoostImpl::chi2(double x, double m) {
00083     // chi_squared_distribution<> my_chi_squared(m);
00084     return 0.0; // pdf(my_chi_squared, x);
00085 }
00086
00087 double ProbDistribBoostImpl::poisson(double x, double mean) {
00088     // poisson_distribution<> my_poisson(mean);
00089     return 0.0; // pdf(my_poisson, x);
00090 }
00091
00092 double ProbDistribBoostImpl::inverseNormal(double cumulativeProbability, double mean, double stddev) {
00093     // normal_distribution<> my_normal(mean, stddev);
00094     return 0.0; // quantile(my_normal, cumulativeProbability);
00095 }
00096
00097 double ProbDistribBoostImpl::inverseTStudent(double cumulativeProbability, double mean, double stddev,
    double degreeFreedom) {
00098     // students_t_distribution<> my_students_t(degreeFreedom);
00099     return 0.0; // quantile(my_students_t, cumulativeProbability);
00100 }
00101
00102 double ProbDistribBoostImpl::inverseFFisher(double cumulativeProbability, double k, double m) {
00103     // fisher_f_distribution<> my_fisher_f(k, m);
00104     return 0.0; // quantile(my_fisher_f, cumulativeProbability);
00105 }
00106
00107 double ProbDistribBoostImpl::inverseChi2(double cumulativeProbability, double m) {
00108     // chi_squared_distribution<> my_chi_squared(m);
00109     return 0.0; // quantile(my_chi_squared, cumulativeProbability);
00110 }

```

9.337 ProbDistribBoostImpl.h File Reference

```
#include "Prob_Distrib_if.h"
```

Include dependency graph for ProbDistribBoostImpl.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ProbDistribBoostImpl](#)

9.338 ProbDistribBoostImpl.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  ProbDistrib_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 00:32
00012 */
00013
00014 #ifndef PROBDISTRIBBOOSTIMPL_H
00015 #define PROBDISTRIBBOOSTIMPL_H
00016
00017 //##include <boost/math/distributions.hpp>
00018 //##include <boost/math/distributions/normal.hpp>
00019 //##define libBoostInstalled
00020 #include "Prob_Distrib_if.h"
00021
00022 class ProbDistribBoostImpl : public ProbDistrib_if {
00023 public:
00024     double beta(double x, double alpha, double beta);
00025     double chisq(double x, double m);
00026     double erlang(double x, double shape, double scale); // int M
00027     double exponential(double x, double mean);
00028     double fFisher(double x, double k, double m);
00029     double gamma(double x, double shape, double scale);
00030     double logNormal(double x, double mean, double stddev);
00031     double normal(double x, double mean, double stddev);
00032     double poisson(double x, double mean);
00033     double triangular(double x, double min, double mode, double max);
00034     double tStudent(double x, double mean, double stddev, unsigned int degreeFreedom);
00035     double uniform(double x, double min, double max);
00036     double weibull(double x, double shape, double scale);
00037     // inverse
00038     double inverseChi2(double cumulativeProbability, double m);
  
```

```

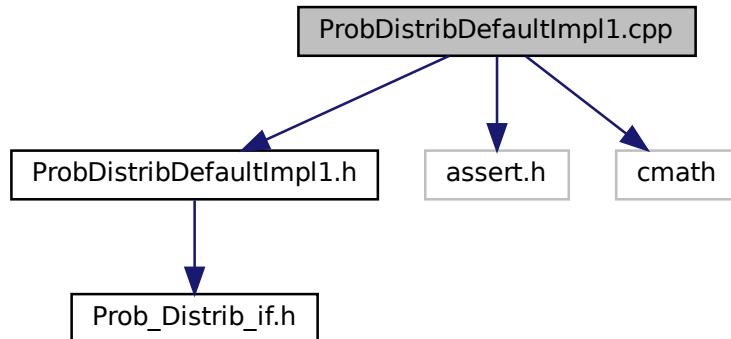
00039     double inverseFFisher(double cumulativeProbability, double k, double m);
00040     double inverseNormal(double cumulativeProbability, double mean, double stddev);
00041     double inverseTStudent(double cumulativeProbability, double mean, double stddev, double
00042         degreeFreedom);
00043     // ...
00044 };
00045 #endif /* PROBDISTRIBBOOSTIMPL_H */
00046

```

9.339 ProbDistribDefaultImpl1.cpp File Reference

```
#include "ProbDistribDefaultImpl1.h"
#include <assert.h>
#include <cmath>
```

Include dependency graph for ProbDistribDefaultImpl1.cpp:



9.340 ProbDistribDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  ProbDistrib.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 23 de Agosto de 2018, 17:25
00012 */
00013
00014 #include "ProbDistribDefaultImpl1.h"
00015 #include <assert.h>
00016 #include <cmath>
00017
00018 double ProbDistribDefaultImpl1::uniform(double x, double min, double max) {
00019     if (x >= min && x <= max)
00020         return 1.0 / (max - min);
00021     else
00022         return 0.0;
00023 }
00024
00025 double ProbDistribDefaultImpl1::exponential(double x, double mean) {
00026     assert(x >= 0);
00027     return 0.0; //mean * exp(-mean * x);
00028 }
00029
00030 double ProbDistribDefaultImpl1::erlang(double x, double shape, double scale) { //
```

```

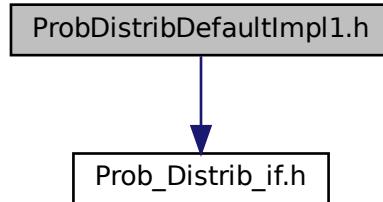
00031     return ProbDistribDefaultImpl1::gamma(x, shape, scale); // \todo: NOT EXACTLY THIS... REDO
00032 }
00033
00034 double ProbDistribDefaultImpl1::normal(double x, double mean, double stddev) {
00035     double p1 = 1 / (stddev * std::sqrt(2 * M_PI));
00036     double rdf = (x - mean) / stddev;
00037     double p2 = std::exp(-0.5 * rdf * rdf);
00038     return p1*p2;
00039 }
00040
00041 double ProbDistribDefaultImpl1::gamma(double x, double shape, double scale) {
00042     // gamma_distribution<> my_gamma(shape, scale);
00043     return 0.0; // pdf(my_gamma, x);
00044 }
00045
00046 double ProbDistribDefaultImpl1::beta(double x, double alpha, double beta) {
00047     // beta_distribution<> my_beta(alpha, beta);
00048     return 0.0; // pdf(my_beta, x);
00049 }
00050
00051 double ProbDistribDefaultImpl1::weibull(double x, double shape, double scale) {
00052     // weibull_distribution<> my_weibull(shape, scale);
00053     return 0.0; // pdf(my_weibull, x);
00054 }
00055
00056 double ProbDistribDefaultImpl1::logNormal(double x, double mean, double stddev) {
00057     // lognormal_distribution<> my_lognormal(mean, stddev);
00058     return 0.0; // pdf(my_lognormal, x);
00059 }
00060
00061 double ProbDistribDefaultImpl1::triangular(double x, double min, double mode, double max) {
00062     // triangular_distribution<> my_triangular(min, mode, max);
00063     return 0.0; // pdf(my_triangular, x);
00064 }
00065
00066 double ProbDistribDefaultImpl1::tStudent(double x, double mean, double stddev, unsigned int
    degreeFreedom) {
00067     // students_t_distribution<> my_students_t(degreeFreedom);
00068     return 0.0; // pdf(my_students_t, x)*mean + stddev; // t-student SEEKS to be based on NORM(0,1)
00069 }
00070
00071 double ProbDistribDefaultImpl1::fFisher(double x, double k, double m) {
00072     // fisher_f_distribution<> my_fisher_f(k, m);
00073     return 0.0; // pdf(my_fisher_f, x);
00074 }
00075
00076 double ProbDistribDefaultImpl1::chi2(double x, double m) {
00077     // chi_squared_distribution<> my_chi_squared(m);
00078     return 0.0; // pdf(my_chi_squared, x);
00079 }
00080
00081 double ProbDistribDefaultImpl1::poisson(double x, double mean) {
00082     // poisson_distribution<> my_poisson(mean);
00083     return 0.0; // pdf(my_poisson, x);
00084 }
00085
00086 double ProbDistribDefaultImpl1::inverseNormal(double cumulativeProbability, double mean, double
    stddev) {
00087     // normal_distribution<> my_normal(mean, stddev);
00088     return 0.0; // quantile(my_normal, cumulativeProbability);
00089 }
00090
00091 double ProbDistribDefaultImpl1::inverseTStudent(double cumulativeProbability, double mean, double
    stddev, double degreeFreedom) {
00092     // students_t_distribution<> my_students_t(degreeFreedom);
00093     return 0.0; // quantile(my_students_t, cumulativeProbability);
00094 }
00095
00096 double ProbDistribDefaultImpl1::inverseFFisher(double cumulativeProbability, double k, double m) {
00097     // fisher_f_distribution<> my_fisher_f(k, m);
00098     return 0.0; // quantile(my_fisher_f, cumulativeProbability);
00099 }
00100
00101 double ProbDistribDefaultImpl1::inverseChi2(double cumulativeProbability, double m) {
00102     // chi_squared_distribution<> my_chi_squared(m);
00103     return 0.0; // quantile(my_chi_squared, cumulativeProbability);
00104 }

```

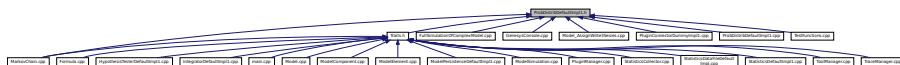
9.341 ProbDistribDefaultImpl1.h File Reference

```
#include "Prob_Distrib_if.h"
```

Include dependency graph for ProbDistribDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ProbDistribDefaultImpl1](#)

9.342 ProbDistribDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File:  ProbDistrib_if.h
00008  * Author: rafael.luiz.cancian
00009  *
00010  *
00011  * Created on 2 de Agosto de 2018, 00:32
00012 */
00013
00014 #ifndef PROBDISTRIBDEFAULTIMPL1_H
00015 #define PROBDISTRIBDEFAULTIMPL1_H
00016
00017 #include "Prob_Distrib_if.h"
00018
00019 class ProbDistribDefaultImpl1 : public ProbDistrib_if {
00020 public:
00021     double beta(double x, double alpha, double beta);
00022     double chi2(double x, double m);
00023     double erlang(double x, double shape, double scale); // int M
00024     double exponential(double x, double mean);
00025     double fFisher(double x, double k, double m);
00026     double gamma(double x, double shape, double scale);
00027     double logNormal(double x, double mean, double stddev);
00028     double normal(double x, double mean, double stddev);
00029     double poisson(double x, double mean);
00030     double triangular(double x, double min, double mode, double max);
00031     double tStudent(double x, double mean, double stddev, unsigned int degreeFreedom);
00032     double uniform(double x, double min, double max);
00033     double weibull(double x, double shape, double scale);
00034     // inverse
00035     double inverseChi2(double cumulativeProbability, double m);
00036     double inverseFFisher(double cumulativeProbability, double k, double m);
00037     double inverseNormal(double cumulativeProbability, double mean, double stddev);
  
```

```

00038     double inverseTStudent(double cumulativeProbability, double mean, double stdDev, double
00039         degreeFreedom);
00040     // ...
00041 }
00042 #endif /* PROBDISTRIBDEFAULTIMPL1_H */
00043

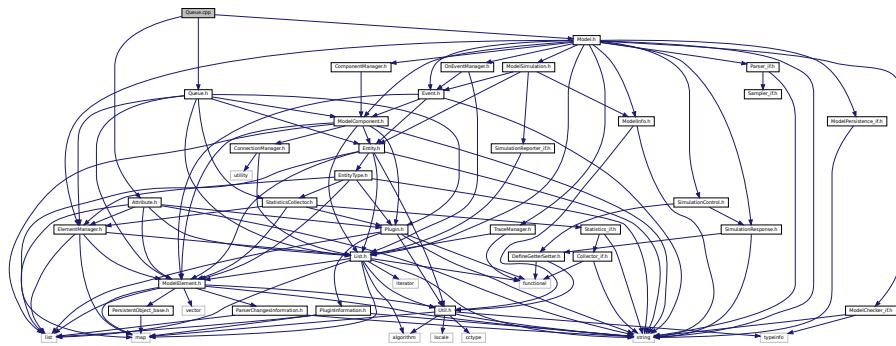
```

9.343 Queue.cpp File Reference

```

#include "Queue.h"
#include "Model.h"
#include "Attribute.h"
Include dependency graph for Queue.cpp:

```



9.344 Queue.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Queue.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 17:12
00012 */
00013
00014 #include "Queue.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 Queue::Queue(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Queue>(), name) {
00019 }
00020
00021 Queue::~Queue() {
00022     //parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatNumberInQueue);
00023     //parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatTimeInQueue);
00024 }
00025
00026 std::string Queue::show() {
00027     return ModelElement::show() +
00028         ",waiting=" + this->_list->show();
00029 }
00030
00031 void Queue::insertElement(Waiting* element) {
00032     _list->insert(element);
00033     if (_reportStatistics)
00034         this->_cstatNumberInQueue->getStatistics()->getCollector()->addValue(_list->size());
00035 }
00036
00037 void Queue::removeElement(Waiting* element) {
00038     double tnow = _parentModel->getSimulation()->getSimulatedTime();
00039     _list->remove(element);
00040     if (_reportStatistics) {

```

```

00041     this->_cstatNumberInQueue->getStatistics()->getCollector()->addValue(_list->size());
00042     double timeInQueue = tnow - element->getTimeStartedWaiting();
00043     this->_cstatTimeInQueue->getStatistics()->getCollector()->addValue(timeInQueue);
00044 }
00045 }
00046
00047 void Queue::initBetweenReplications() {
00048     this->_list->clear();
00049 }
00050
00051 unsigned int Queue::size() {
00052     return _list->size();
00053 }
00054
00055 Waiting* Queue::first() {
00056     return _list->front();
00057 }
00058
00059 Waiting* Queue::getAtRank(unsigned int rank) {
00060     return _list->getAtRank(rank);
00061 }
00062
00063 void Queue::setAttributeName(std::string _attributeName) {
00064     this->_attributeName = _attributeName;
00065 }
00066
00067 std::string Queue::getAttributeName() const {
00068     return _attributeName;
00069 }
00070
00071 void Queue::setOrderRule(OrderRule _orderRule) {
00072     this->_orderRule = _orderRule;
00073 }
00074
00075 OrderRule Queue::getOrderRule() const {
00076     return _orderRule;
00077 }
00078
00079 double Queue::sumAttributesFromWaiting(Util::identification attributeID) {
00080     double sum = 0.0;
00081     for (std::list<Waiting*>::iterator it = _list->list()->begin(); it != _list->list()->endgetEntity()->getAttributeValue(attributeID);
00084     }
00085     return sum;
00086 }
00087
00088 double Queue::getAttributeFromWaitingRank(unsigned int rank, Util::identification attributeID) {
00089     Waiting* wait = _list->getAtRank(rank);
00090     if (wait != nullptr) {
00091         return wait->getEntity()->getAttributeValue(attributeID);
00092     }
00093     return 0.0;
00094 }
00095 PluginInformation* Queue::GetPluginInformation() {
00096     PluginInformation* info = new PluginInformation(Util::TypeOf<Queue>(), &Queue::LoadInstance);
00097     return info;
00098 }
00099
00100 ModelElement* Queue::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00101     Queue* newElement = new Queue(model);
00102     try {
00103         newElement->_loadInstance(fields);
00104     } catch (const std::exception& e) {
00105     }
00106 }
00107     return newElement;
00108 }
00109
00110 bool Queue::_loadInstance(std::map<std::string, std::string>* fields) {
00111     bool res = ModelElement::_loadInstance(fields);
00112     if (res) {
00113         try {
00114             this->_attributeName = loadField(fields, "attributeName", "");
00115             this->_orderRule = static_cast<OrderRule>(std::stoi(loadField(fields, "orderRule",
00116                         std::string(static_cast<int>(OrderRule::FIFO)))));
00117         } catch (...) {
00118         }
00119     }
00120     return res;
00121 }
00122 std::map<std::string, std::string>* Queue::_saveInstance() {
00123     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00124     //Util::TypeOf<Queue>());
00125     if (_orderRule != OrderRule::FIFO) fields->emplace("orderRule", std::string(static_cast<int>

```

```

        (this->_orderRule));
00125     if (_attributeName != "") fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00126     return fields;
00127 }
00128
00129 bool Queue::_check(std::string* errorMessage) {
00130     return _parentModel->getElements()->check(Util::TypeOf<Attribute>(), _attributeName,
00131         "AttributeName", false, errorMessage);
00132 }
00133 void Queue::_createInternalElements() {
00134     if (_reportStatistics) {
00135         if (_cstatNumberInQueue == nullptr) {
00136             _cstatNumberInQueue = new StatisticsCollector(_parentModel, _name + "." + "NumberInQueue",
00137                 this); /* \todo: ++ WHY THIS INSERT "DISPOSE" AND "10ENTITYTYPE" STATCOLL ?? */
00138             _cstatTimeInQueue = new StatisticsCollector(_parentModel, _name + "." + "TimeInQueue",
00139                 this);
00140             _childrenElements->insert({"NumberInQueue", _cstatNumberInQueue});
00141             _childrenElements->insert({"TimeInQueue", _cstatTimeInQueue});
00142         }
00143     } else if (_cstatNumberInQueue != nullptr) {
00144         // \todo: remove
00145         _removeChildrenElements();
00146         //_childrenElements->remove(_cstatNumberInQueue);
00147         //_childrenElements->remove(_cstatTimeInQueue);
00148         //_cstatNumberInQueue->~StatisticsCollector();
00149         //_cstatTimeInQueue->~StatisticsCollector();
00150     }
00151 ParserChangesInformation * Queue::_getParserChangesInformation() {
00152     ParserChangesInformation* changes = new ParserChangesInformation();
00153     //changes->getProductionToAdd()->insert(...);
00154     //changes->getTokensToAdd()->insert(...);
00155     return changes;
00156 }

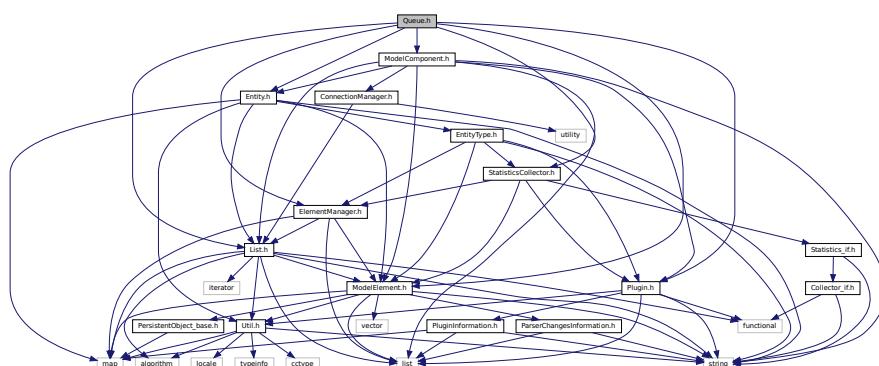
```

9.345 Queue.h File Reference

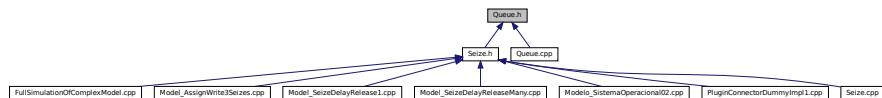
```

#include "ModelElement.h"
#include "List.h"
#include "Entity.h"
#include "ElementManager.h"
#include "StatisticsCollector.h"
#include "Plugin.h"
#include "ModelComponent.h"
Include dependency graph for Queue.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class Waiting
- class Queue

9.346 Queue.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Queue.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Agosto de 2018, 17:12
00012 */
00013
00014 #ifndef QUEUE_H
00015 #define QUEUE_H
00016
00017 #include "ModelElement.h"
00018 #include "List.h"
00019 #include "Entity.h"
00020 #include "ElementManager.h"
00021 #include "StatisticsCollector.h"
00022 #include "Plugin.h"
00023 #include "ModelComponent.h"
00024
00025 class Waiting {
00026 public:
00027
00028     Waiting(Entity* entity, ModelComponent* component, double timeStartedWaiting) {
00029         _entity = entity;
00030         _component = component;
00031         _timeStartedWaiting = timeStartedWaiting;
00032     }
00033
00034     virtual ~Waiting() = default;
00035 public:
00036
00037     virtual std::string show() {
00038         return //ModelElement::show() +
00039             ",entity=" + std::to_string(_entity->getId()) +
00040                 ",component=\"" + _component->getName() + "\" +
00041                 ",timeStatedWaiting=" + std::to_string(_timeStartedWaiting);
00042     }
00043 public:
00044
00045     double getTimeStartedWaiting() const {
00046         return _timeStartedWaiting;
00047     }
00048
00049     ModelComponent* getComponent() const {
00050         return _component;
00051     }
00052
00053     Entity* getEntity() const {
00054         return _entity;
00055     }
00056 private:
00057     Entity* _entity;
00058     ModelComponent* _component;
00059     double _timeStartedWaiting;
00060 };
00061
  
```

```

00091 class Queue : public ModelElement {
00092     public:
00093         enum class OrderRule : int {
00094             FIFO = 1, LIFO = 2, HIGHESTVALUE = 3, SMALLESTVALUE = 4
00095         };
00096     };
00097
00098     public:
00099         Queue(Model* model, std::string name = "");
00100         virtual ~Queue();
00101     public:
00102         virtual std::string show();
00103     public:
00104         static PluginInformation* GetPluginInformation();
00105         static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00106     public:
00107         void insertElement(Waiting* element);
00108         void removeElement(Waiting* element);
00109         unsigned int size();
00110         Waiting* first();
00111         Waiting* getAtRank(unsigned int rank);
00112         void setAttributeName(std::string _attributeName);
00113         std::string getAttributeName() const;
00114         void setOrderRule(OrderRule _orderRule);
00115         OrderRule getOrderRule() const;
00116     public: // to implement SIMAN functions
00117         double sumAttributesFromWaiting(Util::identification attributeID); // use to implement SIMAN SAQUE
00118         function
00119         double getAttributeFromWaitingRank(unsigned int rank, Util::identification attributeID);
00120     public:
00121         void initBetweenReplications();
00122     protected:
00123         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00124         virtual std::map<std::string, std::string>* _saveInstance();
00125         virtual bool _check(std::string* errorMessage);
00126         virtual void _createInternalElements();
00127         virtual ParserChangesInformation* _getParserChangesInformation();
00128     private:
00129         void _initCStats();
00130     private: //1::n
00131         List<Waiting*>* _list = new List<Waiting*>();
00132     private: //1::1
00133         OrderRule _orderRule = OrderRule::FIFO;
00134         std::string _attributeName = "";
00135     private: // inner children elements
00136         StatisticsCollector* _cstatNumberInQueue = nullptr;
00137         StatisticsCollector* _cstatTimeInQueue;
00138     };
00139
00140 #endif /* QUEUE_H */
00141

```

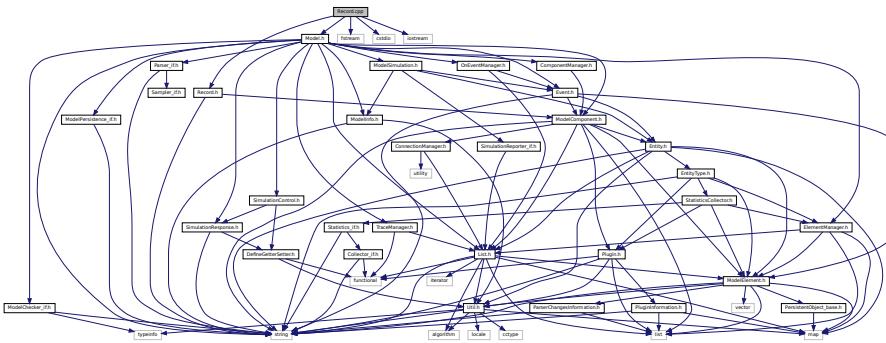
9.347 README.md File Reference**9.348 Record.cpp File Reference**

```

#include "Record.h"
#include "Model.h"
#include <fstream>
#include <cstdio>
#include <iostream>

```

Include dependency graph for Record.cpp:



9.349 Record.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Record.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 13:52
00012 */
00013
00014 #include "Record.h"
00015 #include "Model.h"
00016 #include <fstream>
00017 #include <cstdio>
00018 #include <iostream>
00019
00020 Record::Record(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Record>(), name) {
00021     _cstatExpression = new StatisticsCollector(_parentModel, _expressionName, this);
00022     _parentModel->getElements()->insert(_cstatExpression);
00023 }
00024
00025 Record::~Record() {
00026     _parentModel->getElements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatExpression);
00027 }
00028
00029 std::string Record::show() {
00030     return ModelComponent::show() +
00031         ",expressionName=\"" + this->_expressionName + "\"\"" +
00032         ",expression=\"\"\" + _expression + \"\"\" +
00033         "filename=\"\"\" + _filename + \"\"\"";
00034 }
00035
00036 void Record::setExpressionName(std::string expressionName) {
00037     this->_expressionName = expressionName;
00038     this->_cstatExpression->setName(expressionName);
00039 }
00040
00041 std::string Record::getExpressionName() const {
00042     return _expressionName;
00043 }
00044
00045 StatisticsCollector* Record::getCstatExpression() const {
00046     return _cstatExpression;
00047 }
00048
00049 void Record::setFilename(std::string filename) {
00050     this->_filename = filename;
00051 }
00052
00053 std::string Record::getFilename() const {
00054     return _filename;
00055 }
00056
00057 void Record::setExpression(std::string expression) {
00058     this->_expression = expression;
00059 }
```

```

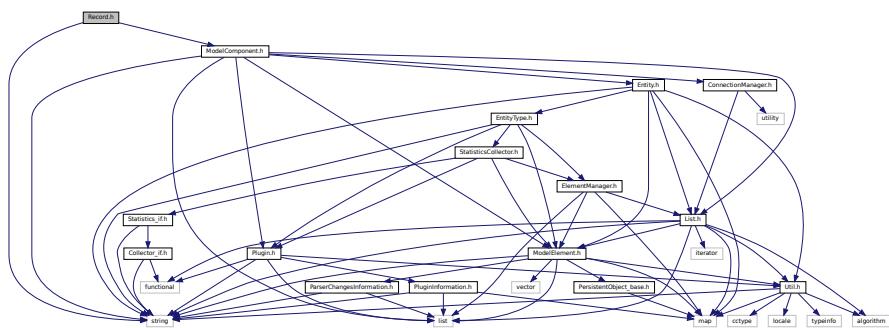
00061 std::string Record::getExpression() const {
00062     return _expression;
00063 }
00064
00065 void Record::_execute(Entity* entity) {
00066     double value = _parentModel->parseExpression(_expression);
00067     _cstatExpression->getStatistics()->getCollector()->addValue(value);
00068     std::ofstream file;
00069     file.open(_filename, std::ofstream::out | std::ofstream::app);
00070     file << value << std::endl;
00071     file.close(); // \todo: open and close for every data is not a good idea. Should open when
00072     // replication starts and close when it finishes.
00073     _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00074     entity, this, "Recording value " + std::to_string(value));
00075     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00076 }
00077
00078 std::map<std::string, std::string>* Record::_saveInstance() {
00079     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00080     //Util::TypeOf<Record>());
00081     fields->emplace("expression0", "\"" + this->_expression + "\"");
00082     fields->emplace("expressionName0", this->_expressionName);
00083     fields->emplace("fileName0", "\"" + this->_filename + "\"");
00084     return fields;
00085 }
00086
00087 bool Record::_loadInstance(std::map<std::string, std::string>* fields) {
00088     bool res = ModelComponent::_loadInstance(fields);
00089     if (res) {
00090         this->_expression = ((*fields->find("expression0"))).second;
00091         this->_expressionName = ((*fields->find("expressionName0"))).second;
00092         this->_filename = ((*fields->find("fileName0"))).second;
00093     }
00094     return res;
00095 }
00096
00097 void Record::_initBetweenReplications() {
00098 }
00099
00100 bool Record::_check(std::string* errorMessage) {
00101     // when cheking the model (before simulating it), remove the file if exists
00102     std::remove(_filename.c_str());
00103     return _parentModel->checkExpression(_expression, "expression", errorMessage);
00104 }
00105
00106 PluginInformation* Record::GetPluginInformation() {
00107     PluginInformation* info = new PluginInformation(Util::TypeOf<Record>(), &Record::LoadInstance);
00108     return info;
00109 }
00110
00111 ModelComponent* Record::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00112     Record* newComponent = new Record(model);
00113     try {
00114         newComponent->_loadInstance(fields);
00115     } catch (const std::exception& e) {
00116     }
00117     return newComponent;
00118 }
00119

```

9.350 Record.h File Reference

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

Include dependency graph for Record.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `Record`

9.351 Record.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Record.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 9 de Agosto de 2018, 13:52
00012 */
00013
00014 #ifndef RECORD_H
00015 #define RECORD_H
00016
00017 #include "ModelComponent.h"
00018 #include <string>
00019
00062 class Record : public ModelComponent {
00063 public:
00064     Record(Model* model, std::string name = "");
00065     virtual ~Record();
00066 public:
00067     void setFilename(std::string filename);
00068     std::string getFilename() const;
00069     void setExpression(std::string expression);
00070     std::string getExpression() const;
00071     void setExpressionName(std::string expressionName);
00072     std::string getExpressionName() const;
00073     StatisticsCollector* getCstatExpression() const;
00074 public:
00075     virtual std::string show();
00076 public:
00077     static PluginInformation* GetPluginInformation();
00078     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);

```

```

00079 protected:
00080     virtual void _execute(Entity* entity);
00081     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00082     virtual void _initBetweenReplications();
00083     virtual std::map<std::string, std::string>* _saveInstance();
00084     virtual bool _check(std::string* errorMessage);
00085 private:
00086     std::string _expression = "";
00087     std::string _expressionName = "";
00088     std::string _filename = "";
00089 private:
00090     // not a child element
00091     StatisticsCollector* _cstatExpression; /* \todo: Create an internal class to aggregate
00092     ExpressionStatisticsCollector, and change Record to get a list of it, so Record can record a set of
00093     expressions into a set of files */
00094 #endif /* RECORD_H */
00095

```

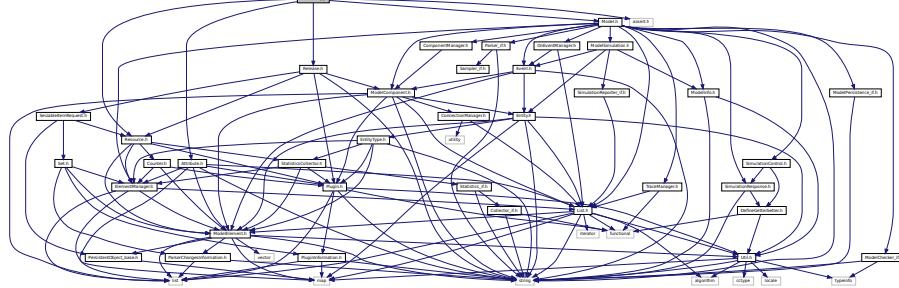
9.352 Release.cpp File Reference

```

#include "Release.h"
#include "Model.h"
#include "Resource.h"
#include "Attribute.h"
#include <assert.h>

```

Include dependency graph for Release.cpp:



9.353 Release.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Release.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #include "Release.h"
00015 #include "Model.h"
00016 #include "Resource.h"
00017 #include "Attribute.h"
00018 #include <assert.h>
00019
00020 Release::Release(Model* model, std::string name) : ModelComponent(model, Util::typeOf<Release>(),
00021     name) {
00022 }
00023 std::string Release::show() {
00024     std::string txt = ModelComponent::show() +
00025         "priority=" + std::to_string(_priority) +
00026         "releaseRequests=\"";

```

```

00027     unsigned short i = 0;
00028     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++, i++) {
00029         txt += "request" + std::to_string(i) + "[" + _releaseRequests->show() + "],";
00030     }
00031     txt = txt.substr(0, txt.length() - 1) + "}";
00032     return txt;
00033 }
00034
00035 void Release::setPriority(unsigned short _priority) {
00036     this->_priority = _priority;
00037 }
00038
00039 unsigned short Release::priority() const {
00040     return _priority;
00041 }
00042
00043 List<SeizableItemRequest*>* Release::getReleaseRequests() const {
00044     return _releaseRequests;
00045 }
00046
00047 //void Release::setResource(Resource* _resource) {
00048 //    this->_resource = _resource;
00049 //}
00050
00051 //Resource* Release::resource() const {
00052 //    return _resource;
00053 //}
00054
00055 void Release::_execute(Entity* entity) {
00056     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
00057         Resource* resource = (*it)->getResource();
00058         unsigned int quantity = _parentModel->parseExpression((*it)->getQuantityExpression());
00059         assert((*it)->getResource()->getNumberBusy() >= quantity);
00060         _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(),
00061             entity, this, "Entity frees " + std::to_string(quantity) + " units of resource \""
00062             + resource->getName() + "\" seized on time "
00063             + std::to_string((*it)->getResource()->getLastTimeSeized()));
00064         (*it)->getResource()->release(quantity, _parentModel->getSimulation()->getSimulatedTime());
00065         //{releases and sets the 'LastTimeSeized' property}
00066     }
00067     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00068 }
00069
00070 void Release::_initBetweenReplications() {
00071     for (std::list<SeizableItemRequest*>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
00072         (*it)->getResource()->initBetweenReplications();
00073     }
00074 }
00075
00076 bool Release::_loadInstance(std::map<std::string, std::string>* fields) {
00077     bool res = ModelComponent::_loadInstance(fields);
00078     if (res) {
00079         this->_priority = std::stoi(loadField(fields, "priority", "0"));
00080         //Util::identitification resourceId = std::stoi((*(fields->find("resourceId"))).second);
00081         //Resource* res = dynamic_cast<Resource*>
00082         (_model->elements()->element(Util::TypeOf<Resource>(), resourceId));
00083
00084         unsigned short numRequests = std::stoi((*(fields->find("releaseResquestSize"))).second);
00085         for (unsigned short i = 0; i < numRequests; i++) {
00086             //std::string resRequest = ((*(fields->find("resourceItemRequest"))).second);
00087             SeizableItemRequest::ResourceType resourceType =
00088                 static_cast<SeizableItemRequest::ResourceType>(std::stoi(loadField(fields, "resourceType" +
00089                     std::to_string(i), std::to_string(static_cast<int>(SeizableItemRequest::ResourceType::RESOURCE)))));
00090             std::string resourceName = ((*(fields->find("resourceName" + std::to_string(i)))).second);
00091             Resource* resource = dynamic_cast<Resource*>
00092                 (_parentModel->getElements()->element(Util::TypeOf<Resource>(), resourceName));
00093             std::string quantityExpression = loadField(fields, "quantity" + std::to_string(i), "1");
00094             SeizableItemRequest::SelectionRule rule = static_cast<SeizableItemRequest::SelectionRule>
00095                 (std::stoi(loadField(fields, "selectionRule" + std::to_string(i), std::to_string(static_cast<int>(SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)))));
00096             std::string saveAttribute = loadField(fields, "saveAttribute" + std::to_string(i), "");
00097             unsigned int index = std::stoi(loadField(fields, "index" + std::to_string(i), "0"));
00098             this->_releaseRequests->insert(new SeizableItemRequest(resource, quantityExpression,
00099                 resourceType, rule, saveAttribute, index));
00100         }
00101     }
00102     return res;
00103 }
00104
00105 std::map<std::string, std::string>* Release::_saveInstance() {
00106     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00107     //Util::TypeOf<Release>());
00108     if (_priority != 0) fields->emplace("priority", std::to_string(this->_priority));
00109     fields->emplace("releaseResquestSize", std::to_string(_releaseRequests->size()));

```

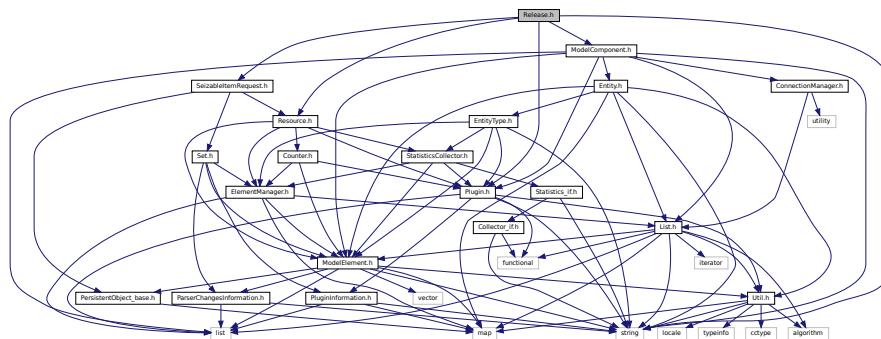
```

00099     unsigned short i = 0;
00100     for (std::list<SeizableItemRequest>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++, i++) {
00101         if ((*it)->getResourceType() != SeizableItemRequest::ResourceType::RESOURCE)
00102             fields->emplace("resourceType" + std::to_string(i), std::to_string(static_cast<int>((*it)->getResourceType())));
00103             //fields->emplace("resourceItemRequest" + std::to_string(i), "(" +
00104             map2str((*it)->_saveInstance()) + ")");
00105             fields->emplace("resourceId" + std::to_string(i),
00106             std::to_string((*it)->getResource()->getId()));
00107             fields->emplace("resourceName" + std::to_string(i), ((*it)->getResource()->getName()));
00108             if ((*it)->getQuantityExpression() != "1") fields->emplace("quantity" + std::to_string(i),
00109             (*it)->getQuantityExpression());
00110             if ((*it)->getSelectionRule() != SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)
00111                 fields->emplace("selectionRule" + std::to_string(i), std::to_string(static_cast<int>((*it)->getSelectionRule())));
00112                 if ((*it)->getSaveAttribute() != "") fields->emplace("saveAttribute" + std::to_string(i),
00113                 (*it)->getSaveAttribute());
00114                 if ((*it)->getIndex() != 0) fields->emplace("index" + std::to_string(i),
00115                 std::to_string((*it)->getIndex()));
00116             }
00117         return fields;
00118     }
00119
00120     bool Release::_check(std::string* errorMessage) {
00121         bool resultAll = true;
00122
00123         for (std::list<SeizableItemRequest>::iterator it = _releaseRequests->list()->begin(); it != _releaseRequests->list()->end(); it++) {
00124             resultAll &= _parentModel->checkExpression((*it)->getQuantityExpression(), "quantity",
00125             errorMessage);
00126             resultAll &= _parentModel->getElements()->check(Util::TypeOf<Resource>(),
00127             (*it)->getResource(), "Resource", errorMessage);
00128             resultAll &= _parentModel->getElements()->check(Util::TypeOf<Attribute>(),
00129             (*it)->getSaveAttribute(), "SaveAttribute", false, errorMessage);
00130         }
00131         return resultAll;
00132     }
00133
00134 //void Release::setResourceName(std::string resourceName) throw () {
00135 //    ModelElement* resource = _parentModel->elements()->element(Util::TypeOf<Resource>(),
00136 //        resourceName);
00137 //    if (resource != nullptr) {
00138 //        this->_resource = dynamic_cast<Resource*> (resource);
00139 //    } else {
00140 //        throw std::invalid_argument("Resource does not exist");
00141 //    }
00142
00143 ModelComponent* Release::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00144     Release* newComponent = new Release(model);
00145     try {
00146         newComponent->_loadInstance(fields);
00147     } catch (const std::exception& e) {
00148
00149     }
00150     return newComponent;
00151 }
00152 }
```

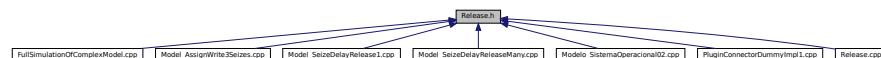
9.354 Release.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Resource.h"
#include "Plugin.h"
#include "SeizableItemRequest.h"
```

Include dependency graph for Release.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Release](#)

9.355 Release.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Release.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #ifndef RELEASE_H
00015 #define RELEASE_H
00016
00017 #include <string>
00018
00019 #include "ModelComponent.h"
00020 #include "Resource.h"
00021 #include "Plugin.h"
00022 #include "SeizableItemRequest.h"
00023
00064 class Release : public ModelComponent {
00065 public:
00066     Release(Model* model, std::string name = "");
00067     virtual ~Release() = default;
00068 public:
00069     virtual std::string show();
00070 public: //static
00071     static PluginInformation* GetPluginInformation();
00072     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00073 public: // get & set
00074     void setPriority(unsigned short _priority);
00075     unsigned short priority() const;
00076 public: // gets
00077     List<SeizableItemRequest*>* getReleaseRequests() const;
00078
00079 protected:

```

```

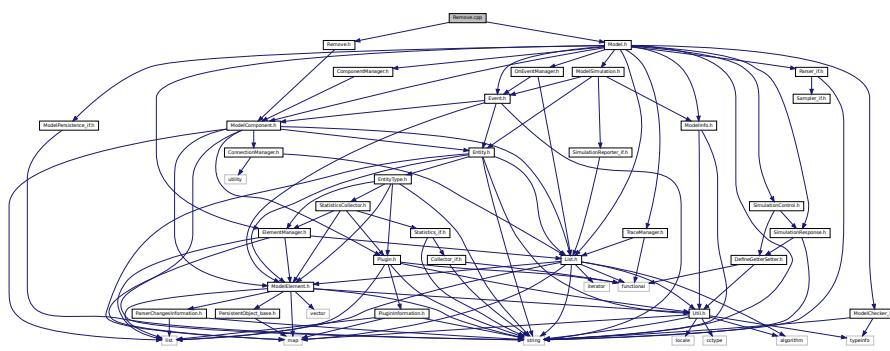
00080     virtual void _execute(Entity* entity);
00081     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00082     virtual void _initBetweenReplications();
00083     virtual std::map<std::string, std::string>* _saveInstance();
00084     virtual bool _check(std::string* errorMessage);
00085 private:
00086     // unsigned int _allocationType = 0; // uint ? enum?
00087     unsigned short _priority = 0;
00088     List<SeizableItemRequest*>* _releaseRequests = new List<SeizableItemRequest*>();
00089 };
00090
00091 #endif /* RELEASE_H */
00092

```

9.356 Remove.cpp File Reference

```
#include "Remove.h"
#include "Model.h"
```

Include dependency graph for Remove.cpp:



9.357 Remove.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Remove.cpp
00009  * Author: rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Remove.h"
00015 #include "Model.h"
00016
00017 Remove::Remove(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Remove>(), name) {
00018 }
00019
00020 std::string Remove::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* Remove::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     Remove* newComponent = new Remove(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030     return newComponent;
00031 }
00032
00033
00034 void Remove::_execute(Entity* entity) {
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);

```

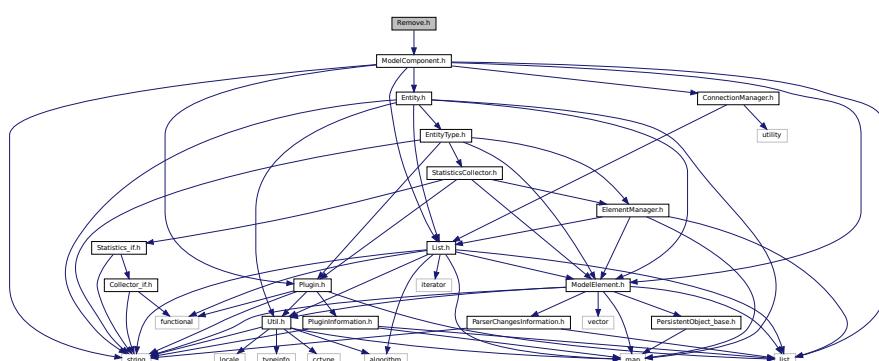
```

00037 }
00038
00039 bool Remove::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
00046
00047 void Remove::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Remove::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool Remove::_check(std::string* errorMessage) {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
00061
00062 PluginInformation* Remove::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Remove>(), &Remove::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067
00068

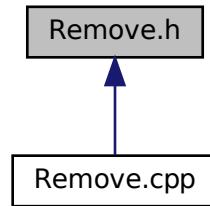
```

9.358 Remove.h File Reference

#include "ModelComponent.h"
Include dependency graph for Remove.h:



This graph shows which files directly or indirectly include this file:



Classes

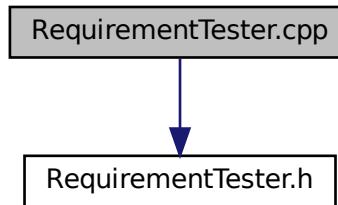
- class Remove

9.359 Remove.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Remove.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef REMOVE_H
00015 #define REMOVE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Remove : public ModelComponent {
00020 public: // constructors
00021     Remove(Model* model, std::string name = "");
00022     virtual ~Remove() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00028 protected: // virtual
00029     virtual void _execute(Entity* entity);
00030     virtual void _initBetweenReplications();
00031     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032     virtual std::map<std::string, std::string>* _saveInstance();
00033     virtual bool _check(std::string* errorMessage);
00034 private: // methods
00035 private: // attributes 1:1
00036 private: // attributes 1:n
00037 };
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058 #endif /* REMOVE_H */
00059
```

9.360 RequirementTester.cpp File Reference

```
#include "RequirementTester.h"
Include dependency graph for RequirementTester.cpp:
```

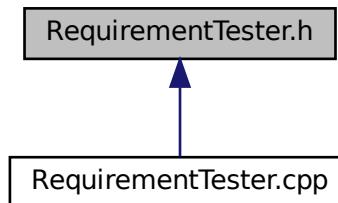


9.361 RequirementTester.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: RequirementTester.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 4 de Novembro de 2019, 14:13
00012 */
00013
00014 #include "RequirementTester.h"
00015
00016 RequirementTester::RequirementTester() {
00017 }
00018
00019 RequirementTester::RequirementTester(const RequirementTester& orig) {
00020 }
00021
00022 RequirementTester::~RequirementTester() {
00023 }
00024
```

9.362 RequirementTester.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class RequirementTester

9.363 RequirementTester.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: RequirementTester.h
00009  * Author: rlcancian
00010 *
00011 * Created on 4 de Novembro de 2019, 14:13
00012 */
00013
00014 #ifndef REQUIREMENTTESTER_H
00015 #define REQUIREMENTTESTER_H
00016
00017 class RequirementTester {
00018 public:
00019     RequirementTester();
00020     RequirementTester(const RequirementTester& orig);
00021     virtual ~RequirementTester();
00022 private:
00023 };
00024
00025
00026 #endif /* REQUIREMENTTESTER_H */
00027

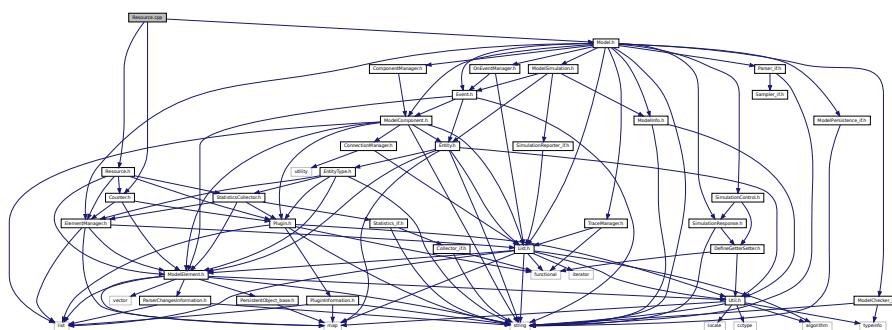
```

9.364 Resource.cpp File Reference

```

#include "Resource.h"
#include "Counter.h"
#include "Model.h"
Include dependency graph for Resource.cpp:

```



9.365 Resource.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Resource.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:52

```

```

00012  */
00013
00014 #include "Resource.h"
00015 #include "Counter.h"
00016 #include "Model.h"
00017
00018 Resource::Resource(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Resource>(),
00019     name) {
00020     GetterMember getter = DefineGetterMember<Resource>(this, &Resource::getCapacity);
00021     SetterMember setter = DefineSetterMember<Resource>(this, &Resource::setCapacity);
00022     model->getControls()->insert(new SimulationControl(Util::TypeOf<Resource>(), _name + ".Capacity",
00023         getter, setter));
00024
00025     GetterMember getter2 = DefineGetterMember<Resource>(this, &Resource::getCostPerUse);
00026     SetterMember setter2 = DefineSetterMember<Resource>(this, &Resource::setCostPerUse);
00027     model->getControls()->insert(new SimulationControl(Util::TypeOf<Resource>(), _name +
00028         ".CostPerUse", getter2, setter2));
00029     // ...
00030 }
00031
00032 std::string Resource::show() {
00033     return ModelElement::show() +
00034         ",capacity=" + std::to_string(_capacity) +
00035         ",costBusyByour=" + std::to_string(_costBusyHour) +
00036         ",costIdleByour=" + std::to_string(_costIdleHour) +
00037         ",costPerUse=" + std::to_string(_costPerUse) +
00038         ",state=" + std::to_string(static_cast<int> (_resourceState));
00039 }
00040
00041 void Resource::seize(unsigned int quantity, double tnow) {
00042     _numberBusy += quantity;
00043     if (_reportStatistics)
00044         _numSeizes->incCountValue(quantity);
00045     _lastTimeSeized = tnow;
00046     _resourceState = Resource::ResourceState::BUSY;
00047     // \todo implement costs
00048 }
00049
00050 void Resource::release(unsigned int quantity, double tnow) {
00051     if (_numberBusy >= quantity) {
00052         _numberBusy -= quantity;
00053     } else {
00054         _numberBusy = 0;
00055     }
00056     if (_numberBusy == 0) {
00057         _resourceState = Resource::ResourceState::IDLE;
00058     }
00059     double timeSeized = tnow - _lastTimeSeized;
00060     if (_reportStatistics) {
00061         _numReleases->incCountValue(quantity);
00062         _cstatTimeSeized->getStatistics()->getCollector()->addValue(timeSeized);
00063     }
00064     //
00065     _lastTimeSeized = timeSeized;
00066     _notifyReleaseEventHandlers();
00067 }
00068
00069 void Resource::initBetweenReplications() {
00070     this->_lastTimeSeized = 0.0;
00071     this->_numberBusy = 0;
00072     if (_reportStatistics) {
00073         this->_numSeizes->clear();
00074         this->_numReleases->clear();
00075     }
00076 }
00077
00078 void Resource::setResourceState(ResourceState _resourceState) {
00079     this->_resourceState = _resourceState;
00080 }
00081
00082 Resource::ResourceState Resource::getResourceState() const {
00083     return _resourceState;
00084 }
00085
00086 void Resource::setCapacity(unsigned int _capacity) {
00087     this->_capacity = _capacity;
00088 }
00089
00090 unsigned int Resource::getCapacity() const {
00091     return _capacity;
00092 }
00093
00094 void Resource::setCostBusyHour(double _costBusyHour) {
00095     this->_costBusyHour = _costBusyHour;

```

```
00096 }
00097
00098 double Resource::getCostBusyHour() const {
00099     return _costBusyHour;
00100 }
00101
00102 void Resource::setCostIdleHour(double _costIdleHour) {
00103     this->_costIdleHour = _costIdleHour;
00104 }
00105
00106 double Resource::getCostIdleHour() const {
00107     return _costIdleHour;
00108 }
00109
00110 void Resource::setCostPerUse(double _costPerUse) {
00111     this->_costPerUse = _costPerUse;
00112 }
00113
00114 double Resource::getCostPerUse() const {
00115     return _costPerUse;
00116 }
00117
00118 unsigned int Resource::getNumberBusy() const {
00119     return _numberBusy;
00120 }
00121
00122 void Resource::addReleaseResourceEventHandler(ResourceEventHandler eventHandler) {
00123     this->_resourceEventHandlers->insert(eventHandler); // \todo: priority should be registered as well, so handlers are invoked ordered by priority
00124 }
00125
00126 double Resource::getLastTimeSeized() const {
00127     return _lastTimeSeized;
00128 }
00129
00130 void Resource::_notifyReleaseEventHandlers() {
00131     for (std::list<ResourceEventHandler>::iterator it = this->_resourceEventHandlers->list()->begin();
00132         it != _resourceEventHandlers->list()->end(); it++) {
00133         (*it)(this);
00134     }
00135
00136 PluginInformation* Resource::GetPluginInformation() {
00137     PluginInformation* info = new PluginInformation(Util::TypeOf<Resource>(),
00138         &Resource::LoadInstance);
00139     return info;
00140 }
00141 ModelElement* Resource::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00142     Resource* newElement = new Resource(model);
00143     try {
00144         newElement->_loadInstance(fields);
00145     } catch (const std::exception& e) {
00146     }
00147 }
00148     return newElement;
00149 }
00150
00151 bool Resource::_loadInstance(std::map<std::string, std::string>* fields) {
00152     bool res = ModelElement::_loadInstance(fields);
00153     if (res) {
00154         //this->_capacity = (fields->find("capacity") != fields->end() ?
00155         std::stoi((*(fields->find("capacity"))).second) : 1);
00156         this->_capacity = std::stoi(loadField(fields, "capacity", "1"));
00157         this->_costBusyHour = std::stod(loadField(fields, "costBusyHour", "1.0"));
00158         //(*(fields->find("costBusyHour"))).second);
00159         this->_costIdleHour = std::stod(loadField(fields, "costIdleHour", "1.0"));
00160         //(*(fields->find("costIdleHour"))).second);
00161         this->_costPerUse = std::stod(loadField(fields, "costPerUse", "1.0"));
00162         //(*(fields->find("costPerUse"))).second);
00163     }
00164     return res;
00165 }
00166
00167 std::map<std::string, std::string>* Resource::_saveInstance() {
00168     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00169     //Util::TypeOf<Resource>());
00170     if (_capacity != 1) fields->emplace("capacity", std::to_string(this->_capacity));
00171     if (_costBusyHour != 1.0) fields->emplace("costBusyHour", std::to_string(this->_costBusyHour));
00172     if (_costIdleHour != 1.0) fields->emplace("costIdleHour", std::to_string(this->_costIdleHour));
00173     if (_costPerUse != 1.0) fields->emplace("costPerUse", std::to_string(this->_costPerUse));
00174 }
```

```

00175
00176 void Resource::createInternalElements() {
00177     if (_reportStatistics && _cstatTimeSeized == nullptr) {
00178         _cstatTimeSeized = new StatisticsCollector(_parentModel, _name + "." + "TimeSeized", this);
00179         _numSeizes = new Counter(_parentModel, _name + "." + "Seizes", this);
00180         _numReleases = new Counter(_parentModel, _name + "." + "Releases", this);
00181         _childrenElements->insert({"TimeSeized", _cstatTimeSeized});
00182         _childrenElements->insert({"Seizes", _numSeizes});
00183         _childrenElements->insert({"Releases", _numReleases});
00184     } else if (!_reportStatistics && _cstatTimeSeized != nullptr) {
00185         _removeChildrenElements();
00186     }
00187 }

```

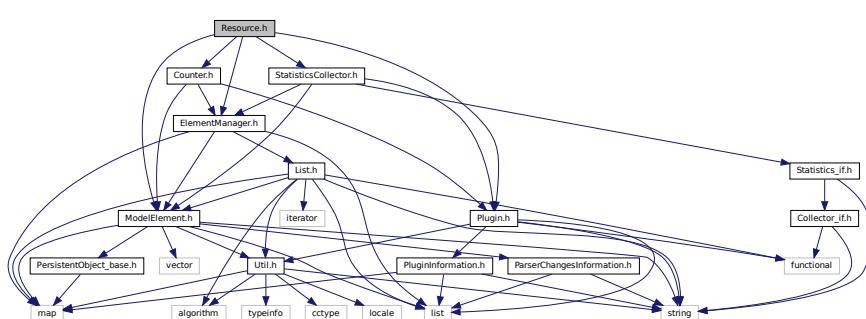
9.366 Resource.h File Reference

```

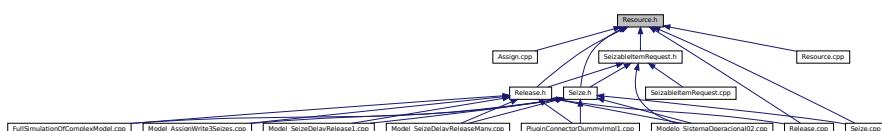
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
#include "Counter.h"
#include "Plugin.h"

```

Include dependency graph for Resource.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Resource](#)

9.367 Resource.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Resource.h

```

```
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:52
00012 */
00013
00014 #ifndef RESOURCE_H
00015 #define RESOURCE_H
00016
00017 #include "ModelElement.h"
00018 #include "StatisticsCollector.h"
00019 #include "ElementManager.h"
00020 #include "Counter.h"
00021 #include "Plugin.h"
00022
00023 class SeizableItemRequest;
00024
00025 class Resource : public ModelElement {
00026 public:
00027     typedef std::function<void(Resource*)> ResourceEventHandler;
00028
00029     template<typename Class>
00030     static ResourceEventHandler SetResourceEventHandler(void (Class::*function)(Resource*), Class *
00031     object) {
00032         return std::bind(function, object, std::placeholders::_1);
00033     }
00034
00035     enum class ResourceState : int {
00036         IDLE = 1, BUSY = 2, FAILED = 3, INACTIVE = 4, OTHER = 5
00037     };
00038
00039 public:
00040     //Resource(Model* model);
00041     Resource(Model* model, std::string name = "");
00042     virtual ~Resource();
00043
00044 public:
00045     virtual std::string show();
00046
00047 public:
00048     static PluginInformation* GetPluginInformation();
00049     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00050
00051 public:
00052     void seize(unsigned int quantity, double tnow);
00053     void release(unsigned int quantity, double tnow);
00054     void initBetweenReplications();
00055
00056 public: // g&s
00057     void setResourceState(ResourceState _resourceState);
00058     Resource::ResourceState getResourceState() const;
00059     void setCapacity(unsigned int _capacity);
00060     unsigned int getCapacity() const;
00061     void setCostBusyHour(double _costBusyHour);
00062     double getCostBusyHour() const;
00063     void setCostIdleHour(double _costIdleHour);
00064     double getCostIdleHour() const;
00065     void setCostPerUse(double _costPerUse);
00066     double getCostPerUse() const;
00067
00068 public: // gets
00069     unsigned int getNumberBusy() const;
00070
00071 public:
00072     void addReleaseResourceEventHandler(ResourceEventHandler eventHandler);
00073     double getLastTimeSeized() const;
00074
00075 protected:
00076     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00077     virtual std::map<std::string, std::string>* _saveInstance();
00078     virtual bool _check(std::string* errorMessage);
00079     virtual void _createInternalElements();
00080
00081 private:
00082     void _notifyReleaseEventHandlers();
00083     //private:
00084     // ElementManager* _elems;
00085
00086 private:
00087     unsigned int _capacity = 1;
00088     double _costBusyHour = 1.0;
00089     double _costIdleHour = 1.0;
00090     double _costPerUse = 1.0;
00091     ResourceState _resourceState = ResourceState::IDLE;
00092
00093 private: // only gets
00094     unsigned int _numberBusy = 0;
00095     //unsigned int _numberOut = 0;
00096     double _lastTimeSeized = 0.0; // \todo: It won't work for resources with capacity>1, when not all
00097     // capacity is seized and them some more are seized. Seized time of first units will be lost. I don't
00098     // have a solution so far
00099 private: // not gets nor sets
00100     //unsigned int _seizes = 0;
00101     //double _whenSeized; // same as last? check
00102
00103 private: //1::n
00104     List<ResourceEventHandler>* _resourceEventHandlers = new List<ResourceEventHandler>();
00105     //aFailures: TStringList;
```

```

00149     //std::list<Failure*>* _failures;
00150 private: // inner children elements
00151     StatisticsCollector* _cstatTimeSeized = nullptr;
00152     Counter* _numSeizes;
00153     Counter* _numReleases;
00154 };
00155
00156 #endif /* RESOURCE_H */
00157

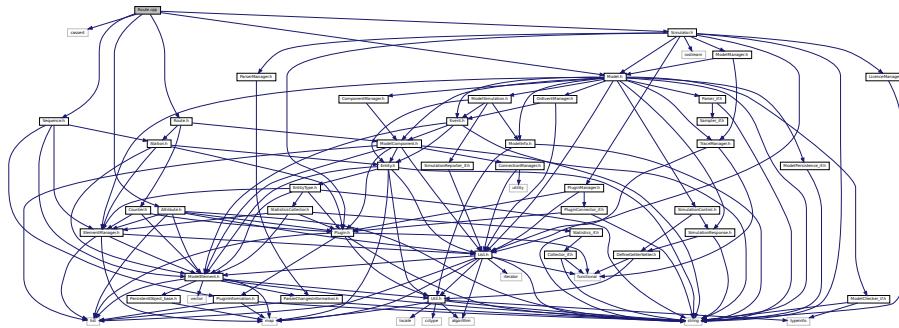
```

9.368 Route.cpp File Reference

```

#include <cassert>
#include "Route.h"
#include "Model.h"
#include "Attribute.h"
#include "Simulator.h"
#include "Sequence.h"
Include dependency graph for Route.cpp:

```



9.369 Route.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:   Route.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #include <cassert>
00015 #include "Route.h"
00016 #include "Model.h"
00017 #include "Attribute.h"
00018 #include "Simulator.h"
00019 #include "Sequence.h"
00020
00021 Route::Route(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Route>(), name) {
00022 }
00023
00024 std::string Route::show() {
00025     std::string msg = ModelComponent::show() +
00026         ",destinationType=" + std::to_string(static_cast<int> (this->_routeDestinationType)) +
00027         ",timeExpression=" + this->_routeTimeExpression + " " +
00028         Util::StrTimeUnit(this->_routeTimeTimeUnit);
00029     if (_station != nullptr)
00030         msg += ",station=" + this->_station->getName();
00031     return msg;
00032 }
00033 ModelComponent* Route::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00034     Route* newComponent = new Route(model);

```

```

00035     try {
00036         newComponent->_loadInstance(fields);
00037     } catch (const std::exception& e) {
00038     }
00039 }
00040     return newComponent;
00041 }
00042
00043 void Route::setStation(Station* _station) {
00044     this->_station = _station;
00045 }
00046
00047 Station* Route::getStation() const {
00048     return _station;
00049 }
00050
00051 void Route::setRouteTimeExpression(std::string _routeTimeExpression) {
00052     this->_routeTimeExpression = _routeTimeExpression;
00053 }
00054
00055 std::string Route::getRouteTimeExpression() const {
00056     return _routeTimeExpression;
00057 }
00058
00059 void Route::setRouteTimeTimeUnit(Util::TimeUnit _routeTimeTimeUnit) {
00060     this->_routeTimeTimeUnit = _routeTimeTimeUnit;
00061 }
00062
00063 Util::TimeUnit Route::getRouteTimeTimeUnit() const {
00064     return _routeTimeTimeUnit;
00065 }
00066
00067 void Route::setRouteDestinationType(DestinationType _routeDestinationType) {
00068     this->_routeDestinationType = _routeDestinationType;
00069 }
00070
00071 Route::DestinationType Route::getRouteDestinationType() const {
00072     return _routeDestinationType;
00073 }
00074
00075 void Route::_execute(Entity* entity) {
00076     Station* destinyStation = _station;
00077     if (_routeDestinationType == Route::DestinationType::BySequence) {
00078         unsigned int sequenceId = static_cast<unsigned int>
00079             (entity->getAttributeValue("Entity.Sequence"));
00080         unsigned int step = static_cast<unsigned int>
00081             (entity->getAttributeValue("Entity.SequenceStep"));
00082         Sequence* sequence = static_cast<Sequence*>
00083             (_parentModel->getElements()->getElement(Util::TypeOf<Sequence>(), sequenceId));
00084         SequenceStep* seqStep = sequence->getSteps()->getAtRank(step);
00085         if (seqStep == nullptr) {
00086             step = 0;
00087             seqStep = sequence->getSteps()->getAtRank(step);
00088             assert(seqStep != nullptr);
00089         }
00090         destinyStation = seqStep->getStation();
00091         for (std::list<std::string>::iterator it = seqStep->getAssignments()->begin(); it != seqStep->getAssignments()->end(); it++) {
00092             _parentModel->parseExpression(*it);
00093         }
00094         entity->setAttributeValue("Entity.SequenceStep", step + 1.0);
00095     }
00096     if (_reportStatistics)
00097         _numberIn->incCountValue();
00098     // adds the route time to the TransferTime statistics / attribute related to the Entity
00099     double routeTime = _parentModel->parseExpression(_routeTimeExpression) *
00100         Util::TimeUnitConvert(_routeTimeTimeUnit,
00101         _parentModel->getSimulation()->getReplicationLengthTimeUnit());
00102     if (entity->getEntityType()->isReportStatistics())
00103         entity->getEntityType()->addGetStatisticsCollector(entity->getEntityType() +
00104             ".TransferTime")->getStatistics()->getCollector()->addValue(routeTime);
00105     entity->setAttributeValue("Entity.TotalTransferTime",
00106         entity->getAttributeValue("Entity.TotalTransferTime") + routeTime);
00107     if (routeTime > 0.0) {
00108         // calculates when this Entity will reach the end of this route and schedule this Event
00109         double routeEndTime = _parentModel->getSimulation()->getSimulatedTime() + routeTime;
00110         Event* newEvent = new Event(routeEndTime, entity,
00111             destinyStation->getEnterIntoStationComponent());
00112         _parentModel->getFutureEvents()->insert(newEvent);
00113         _parentModel->getTracer()->trace("End of route of entity " +
00114             std::to_string(entity->entityNumber()) + " to the component \""
00115             destinyStation->getEnterIntoStationComponent()->getName() + "\" was scheduled to time " +
00116             std::to_string(routeEndTime));
00117     } else {
00118         // send without delay
00119         _parentModel->sendEntityToComponent(entity, destinyStation->getEnterIntoStationComponent(),
00120             0.0);

```

```

00109     }
00110 }
00111
00112 bool Route::_loadInstance(std::map<std::string, std::string>* fields) {
00113     bool res = ModelComponent::_loadInstance(fields);
00114     if (res) {
00115         this->_routeTimeExpression = loadField(fields, "routeTimeExpression", "0.0");
00116         this->_routeTimeTimeUnit = static_cast<Util::TimeUnit>
00117             (std::stoi((*fields->find("routeTimeTimeUnit")).second));
00118         this->_routeDestinationType = static_cast<Route::DestinationType> (std::stoi(loadField(fields,
00119             "routeDestinationType", std::to_string(static_cast<int> (DestinationType::Station))))) ;
00120         std::string stationName = ((*fields->find("stationName"))).second;
00121         Station* station = dynamic_cast<Station*>
00122             (_parentModel->getElements()->getElement(Util::TypeOf<Station>(), stationName));
00123         this->_station = station;
00124     }
00125     return res;
00126 }
00127
00128
00129 std::map<std::string, std::string>* Route::_saveInstance() {
00130     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00131     fields->emplace("stationId", std::to_string(this->_station->getId()));
00132     fields->emplace("stationName", "\\" + (this->_station->getName()) + "\\");
00133     if (_routeTimeExpression != "0.0") fields->emplace("routeTimeExpression", "\\"
00134         + this->_routeTimeExpression + "\\");
00135     if (_routeTimeTimeUnit != Util::TimeUnit::second) fields->emplace("routeTimeTimeUnit",
00136         std::to_string(static_cast<int> (this->_routeTimeTimeUnit)));
00137     if (_routeDestinationType != DestinationType::Station) fields->emplace("routeDestinationType",
00138         std::to_string(static_cast<int> (this->_routeDestinationType)));
00139     return fields;
00140 }
00141
00142 bool Route::_check(std::string* errorMessage) {
00143     //include attributes needed
00144     ElementManager* elements = _parentModel->getElements();
00145     std::vector<std::string> neededNames = {"Entity.TotalTransferTime", "Entity.Station"};
00146     std::string neededName;
00147     for (unsigned int i = 0; i < neededNames.size(); i++) {
00148         neededName = neededNames[i];
00149         if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00150             Attribute* attr1 = new Attribute(_parentModel, neededName);
00151             elements->insert(attr1);
00152         }
00153     }
00154     // include StatisticsCollector needed in EntityType
00155     std::list<ModelElement*>* enttypes = elements->getElementTypeList(Util::TypeOf<EntityType>())->list();
00156     for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00157         if ((*it)->isReportStatistics())
00158             static_cast<Entity* > ((*it))->addGetStatisticsCollector((*it)->getName() +
00159 ".TransferTime"); // force create this CStat before simulation starts
00160     }
00161     bool resultAll = true;
00162     resultAll &= _parentModel->checkExpression(_routeTimeExpression, "Route time expression",
00163         errorMessage);
00164     if (this->_routeDestinationType == Route::DestinationType::Station) {
00165         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Station>(), _station, "Station",
00166         errorMessage);
00167         if (resultAll) {
00168             resultAll &= _station->getEnterIntoStationComponent() != nullptr;
00169             if (!resultAll) {
00170                 errorMessage->append("Station has no component to enter into it");
00171             }
00172         }
00173     }
00174     PluginInformation* info = new PluginInformation(Util::TypeOf<Route>(), &Route::LoadInstance);
00175     info->setSendTransfer(true);
00176     info->insertDynamicLibFileDependence("station.so");
00177     return info;
00178 }
00179 void Route::_createInternalElements() {
00180     if (_reportStatistics) {
00181         if (_numberIn == nullptr) {
00182             _numberIn = new Counter(_parentModel, _name + "." + "CountNumberIn", this);
00183             _childrenElements->insert({ "CountNumberIn", _numberIn });
00184         }
00185     } else
00186         if (_numberIn != nullptr) {

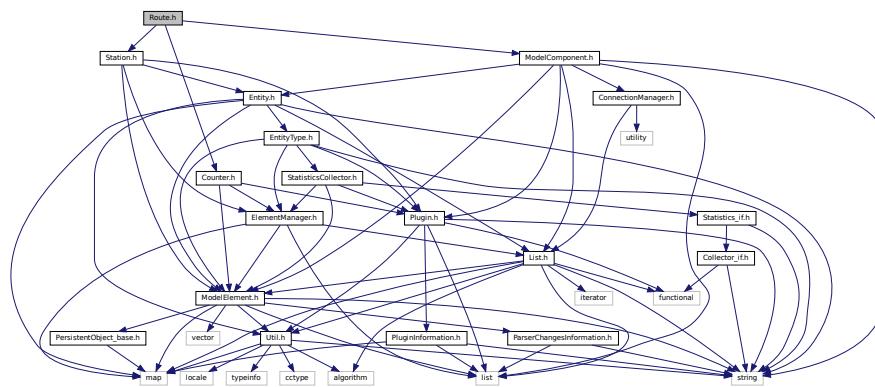
```

```
00187     _removeChildrenElements();  
00188 }  
00189 }
```

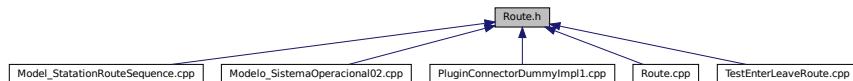
9.370 Route.h File Reference

```
#include "ModelComponent.h"  
#include "Station.h"  
#include "Counter.h"
```

Include dependency graph for Route.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Route](#)

9.371 Route.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: Route.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 03 de Junho de 2019, 15:15  
00012 */  
00013  
00014 #ifndef ROUTE_H  
00015 #define ROUTE_H  
00016  
00017 #include "ModelComponent.h"  
00018 #include "Station.h"  
00019 #include "Counter.h"
```

```

00020
00021
00022
00023 class Route : public ModelComponent {
00024     public:
00025
00026     enum class DestinationType : int {
00027         Station = 0, BySequence = 1
00028     };
00029     public:
00030     Route(Model* model, std::string name = "");
00031     virtual ~Route() = default;
00032     public:
00033     virtual std::string show();
00034     public:
00035     static PluginInformation* GetPluginInformation();
00036     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00037     public:
00038     void setStation(Station* _station);
00039     Station* getStation() const;
00040     void setRouteTimeExpression(std::string _routeTimeExpression);
00041     std::string getRouteTimeExpression() const;
00042     void setRouteTimeTimeUnit(Util::TimeUnit _routeTimeTimeUnit);
00043     Util::TimeUnit getRouteTimeTimeUnit() const;
00044     void setRouteDestinationType(DestinationType _routeDestinationType);
00045     DestinationType getRouteDestinationType() const;
00046     public:
00047     protected:
00048     virtual void _execute(Entity* entity);
00049     virtual void _initBetweenReplications();
00050     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00051     virtual std::map<std::string, std::string>* _saveInstance();
00052     virtual bool _check(std::string* errorMessage);
00053     virtual void _createInternalElements();
00054     private:
00055     std::string _routeTimeExpression = "0.0";
00056     Util::TimeUnit _routeTimeTimeUnit = Util::TimeUnit::second;
00057     Route::DestinationType _routeDestinationType = DestinationType::Station;
00058     private: // association
00059     Station* _station;
00060     private: // children elements
00061     Counter* _numberIn = nullptr;
00062 };
00063
00064 #endif /* ROUTE_H */
00065

```

9.372 Sampler_if.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [Sampler_if](#)
- struct [Sampler_if::RNG_Parameters](#)

9.373 Sampler_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Sampler_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 13:20
00012 */
00013

```

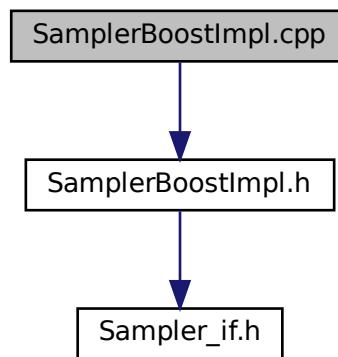
```

00014 #ifndef Sampler_IF_H
00015 #define Sampler_IF_H
00016
00020 class Sampler_if {
00021 public:
00022
00026     struct RNG_Parameters {
00027         virtual ~RNG_Parameters() = default;
00028     };
00029 public: // probability distributions
00030     virtual double random() = 0;
00031     virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit) = 0;
00032     virtual double sampleDiscrete(double acumProb, double value, ...) = 0;
00033     virtual double sampleErlang(double mean, int M) = 0;
00034     virtual double sampleExponential(double mean) = 0;
00035     virtual double sampleGamma(double mean, double alpha) = 0;
00036     virtual double sampleLogNormal(double mean, double stddev) = 0;
00037     virtual double sampleNormal(double mean, double stddev) = 0;
00038     virtual double sampleTriangular(double min, double mode, double max) = 0;
00039     virtual double sampleUniform(double min, double max) = 0;
00040     virtual double sampleWeibull(double alpha, double scale) = 0;
00041 public:
00042     virtual void setRNGparameters(RNG_Parameters* param) = 0;
00043     virtual RNG_Parameters* getRNGparameters() const = 0;
00044 };
00045
00046 #endif /* Sampler_IF_H */
00047

```

9.374 SamplerBoostImpl.cpp File Reference

#include "SamplerBoostImpl.h"
Include dependency graph for SamplerBoostImpl.cpp:



9.375 SamplerBoostImpl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SamplerBoostImpl.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 21 de Outubro de 2019, 17:24
00012 */
00013

```

```

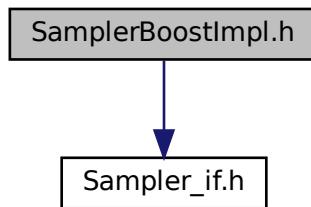
00014 #include "SamplerBoostImpl.h"
00015
00016 //using namespace boost::random;
00017
00018 SamplerBoostImpl::SamplerBoostImpl() {
00019 }
00020
00021 double SamplerBoostImpl::random() {
00022     //uniform_int_distribution<> dist(0.0, 1.0);
00023     return 0.0; // dist(_gen);
00024 }
00025
00026 double SamplerBoostImpl::sampleBeta(double alpha, double beta, double infLimit, double supLimit) {
00027     return 0.0; //dummy
00028 }
00029
00030 double SamplerBoostImpl::sampleDiscrete(double acumProb, double value, ...) {
00031     return 0.0; //dummy
00032 }
00033
00034 double SamplerBoostImpl::sampleErlang(double mean, int M) {
00035     return 0.0; //dummy
00036 }
00037
00038 double SamplerBoostImpl::sampleExponential(double mean) {
00039     return 0.0; //dummy
00040 }
00041
00042 double SamplerBoostImpl::sampleGamma(double mean, double alpha) {
00043     return 0.0; //dummy
00044 }
00045
00046 double SamplerBoostImpl::sampleLogNormal(double mean, double stddev) {
00047     return 0.0; //dummy
00048 }
00049
00050 double SamplerBoostImpl::sampleNormal(double mean, double stddev) {
00051     return 0.0; //dummy
00052 }
00053
00054 double SamplerBoostImpl::sampleTriangular(double min, double mode, double max) {
00055     return 0.0; //dummy
00056 }
00057
00058 double SamplerBoostImpl::sampleUniform(double min, double max) {
00059     //uniform_int_distribution<> dist(min, max);
00060     return 0.0; //dist(_gen);
00061 }
00062
00063 double SamplerBoostImpl::sampleWeibull(double alpha, double scale) {
00064     return 0.0; //dummy
00065 }
00066
00067 void SamplerBoostImpl::setRNGparameters(Sampler_if::RNG_Parameters* param) {
00068
00069 }
00070
00071 Sampler_if::RNG_Parameters* SamplerBoostImpl::getRNGparameters() const {
00072     // \todo: to implement
00073 }

```

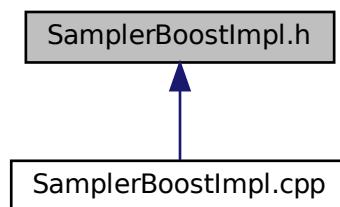
9.376 SamplerBoostImpl.h File Reference

```
#include "Sampler_if.h"
```

Include dependency graph for SamplerBoostImpl.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SamplerBoostImpl](#)
- struct [SamplerBoostImpl::BoostImplRNG_Parameters](#)

9.377 SamplerBoostImpl.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SamplerBoostImpl.h
00009  * Author: rlcancian
00010 *
00011  * Created on 21 de Outubro de 2019, 17:24
00012 */
00013
00014 #ifndef SAMPLERBOOSTIMPL_H
00015 #define SAMPLERBOOSTIMPL_H
00016
00017 #include "Sampler_if.h"
00018 //#include <boost/random.hpp>
00019
00020 class SamplerBoostImpl : public Sampler_if {
00021 public:
  
```

```

00022     struct BoostImplRNG_Parameters : public RNG_Parameters {
00023         double param;
00024     };
00025
00026
00027 public:
00028     SamplerBoostImpl();
00029     virtual ~SamplerBoostImpl() = default;
00030 public: // probability distributions
00031     virtual double random();
00032     virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit);
00033     virtual double sampleDiscrete(double acumProb, double value, ...);
00034     virtual double sampleErlang(double mean, int M);
00035     virtual double sampleExponential(double mean);
00036     virtual double sampleGamma(double mean, double alpha);
00037     virtual double sampleLogNormal(double mean, double stddev);
00038     virtual double sampleNormal(double mean, double stddev);
00039     virtual double sampleTriangular(double min, double mode, double max);
00040     virtual double sampleUniform(double min, double max);
00041     virtual double sampleWeibull(double alpha, double scale);
00042 public:
00043     void reset();
00044 public:
00045     virtual void setRNGparameters(Sampler_if::RNG_Parameters* param);
00046     virtual RNG_Parameters* getRNGparameters() const;
00047 private:
00048     //boost::random::mt19937 _gen;
00049 };
00050
00051 #endif /* SAMPLERBOOSTIMPL_H */
00052

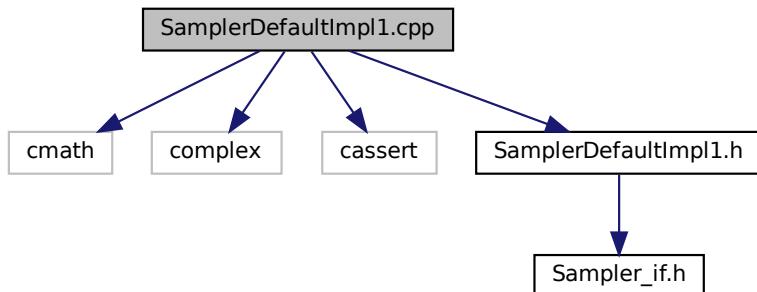
```

9.378 SamplerDefaultImpl1.cpp File Reference

```

#include <cmath>
#include <complex>
#include <cassert>
#include "SamplerDefaultImpl1.h"
Include dependency graph for SamplerDefaultImpl1.cpp:

```



9.379 SamplerDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SamplerDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *

```

```
00011 * Created on 2 de Agosto de 2018, 01:10
00012 * 22/10/2019 old genesys code reinserted
00013 */
00014
00015 #include <cmath>
00016 #include <complex>
00017 #include <cassert>
00018
00019 #include "SamplerDefaultImpl1.h"
00020
00021 //using namespace GenesysKernel;
00022
00023 SamplerDefaultImpl1::SamplerDefaultImpl1() {
00024     reset();
00025 }
00026
00027 void SamplerDefaultImpl1::reset() {
00028     _xi = static_cast<DefaultImpl1RNG_Parameters*>(_param)->seed;
00029     //_normalflag = true;
00030 }
00031
00032 double SamplerDefaultImpl1::random() {
00033     double module = (double) static_cast<DefaultImpl1RNG_Parameters*>(_param)->module;
00034     _xi *= static_cast<DefaultImpl1RNG_Parameters*>(_param)->multiplier;
00035     _xi -= std::trunc((double) _xi / module) * module;
00036     return (double) _xi / (double) static_cast<DefaultImpl1RNG_Parameters*>(_param)->module;
00037 }
00038
00039 double SamplerDefaultImpl1::sampleUniform(double min, double max) {
00040     return min + (max - min) * random();
00041 }
00042
00043 double SamplerDefaultImpl1::sampleExponential(double mean) {
00044     return mean * (-std::log(random()));
00045 }
00046
00047 double SamplerDefaultImpl1::sampleErlang(double mean, int M) {
00048     int i;
00049     double P;
00050     assert((mean >= 0.0) && (M > 0));
00051     P = 1;
00052     for (i = 1; i <= M; i++) {
00053         P *= random();
00054     }
00055     return (mean / M) * (-log(P));
00056 }
00057
00058 double SamplerDefaultImpl1::sampleNormal(double mean, double stddev) {
00059     double z = std::sqrt(-2 * std::log(random())) * std::cos(2 * M_PI * random());
00060     return mean + stddev*z;
00061 }
00062
00063 double SamplerDefaultImpl1::_gammaJonk(double alpha) {
00064     double R;
00065     double R1, R2, X, Y;
00066     do {
00067         do {
00068             R1 = random();
00069             R2 = random();
00070             } while (!(R1 > 1e-30) and (R2 > 1e-30));
00071             if (log(R2) / alpha < -1e3)
00072                 X = 0;
00073             else
00074                 X = exp(log(R2) / alpha);
00075             if ((log(R1) / (1 - alpha) < -1e3))
00076                 Y = 0;
00077             else
00078                 Y = exp(log(R1) / (1 - alpha));
00079         } while (!(X + Y <= 1));
00080         do {
00081             R = random();
00082         } while (!(R > 1e-20));
00083         return -log(R) * X / (Y + X);
00084 }
00085
00086 double SamplerDefaultImpl1::sampleGamma(double mean, double alpha) {
00087     int i;
00088     double P;
00089     int IntAlpha;
00090     double OstAlpha;
00091     assert(!((mean <= 0.0) || (alpha <= 0.0)));
00092     if (alpha < 1.0)
00093         return (mean / alpha) * _gammaJonk(alpha);
00094     else {
00095         if (alpha == 1.0)
00096             return mean * (-log(random()));
00097         else {
```

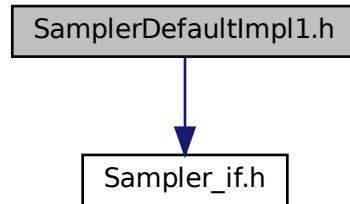
```

00098     IntAlpha = round(alpha);
00099     OstAlpha = alpha - IntAlpha;
00100     do {
00101         P = 1;
00102         for (i = 1; i <= IntAlpha; i++)
00103             P *= random();
00104         } while (!(P > 0));
00105         if (OstAlpha > 0)
00106             return (mean / alpha)*((-log(P)) + _gammaJong(OstAlpha));
00107         else
00108             return (mean / alpha)*(-log(P));
00109     };
00110 };
00111 }
00112
00113 double SamplerDefaultImpl1::sampleBeta(double alpha, double beta, double infLimit, double supLimit) {
00114     double X, Y1, Y2;
00115     assert(!((alpha <= 0.0) || (beta <= 0.0) || (infLimit > supLimit) || (infLimit < 0) || (supLimit <
00116 0)));
00117     do {
00118         Y1 = sampleGamma(alpha, alpha);
00119         Y2 = sampleGamma(beta, beta);
00120         X = Y1 / (Y1 + Y2);
00121     } while (!(X >= 0) && (X <= 1.0));
00122     return infLimit + (supLimit - infLimit) * X;
00123 }
00124 double SamplerDefaultImpl1::sampleWeibull(double alpha, double scale) {
00125     assert(!((alpha <= 0.0) || (scale <= 0.0)));
00126     return exp(log(scale * (-log(random()))) / alpha);
00127 }
00128
00129 double SamplerDefaultImpl1::sampleLogNormal(double mean, double stddev) {
00130     double meanNorm, DispNorm;
00131     assert(!((mean <= 0.0) || (stddev <= 0.0)));
00132     DispNorm = log((stddev * stddev) / (mean * mean) + 1.0);
00133     meanNorm = log(mean) - 0.5 * DispNorm;
00134     return exp(sampleNormal(meanNorm, sqrt(DispNorm)));
00135 }
00136
00137 double SamplerDefaultImpl1::sampleTriangular(double min, double mode, double max) {
00138     double Part1, Part2, Full, R;
00139     assert(!((min > mode) || (max < mode) || (min > max)));
00140     Part1 = mode - min;
00141     Part2 = max - mode;
00142     Full = max - min;
00143     R = random();
00144     if (R <= Part1 / Full)
00145         return min + sqrt(Part1 * Full * R);
00146     else
00147         return max - sqrt(Part2 * Full * (1.0 - R));
00148 }
00149
00150 double SamplerDefaultImpl1::sampleDiscrete(double acumProb, double value, ...) {
00151     // \todo: to implement
00152     return 0.0;
00153 }
00154
00155 void SamplerDefaultImpl1::setRNGparameters(Sampler_if::RNG_Parameters * param) {
00156
00157     _param = param; // there is a better solution for this...
00158 }
00159
00160 Sampler_if::RNG_Parameters * SamplerDefaultImpl1::getRNGparameters() const {
00161     return _param;
00162 }
```

9.380 SamplerDefaultImpl1.h File Reference

```
#include "Sampler_if.h"
```

Include dependency graph for SamplerDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SamplerDefaultImpl1](#)
- struct [SamplerDefaultImpl1::DefaultImpl1RNG_Parameters](#)

9.381 SamplerDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SamplerDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 2 de Agosto de 2018, 01:10
00012 */
00013
00014 #ifndef SAMPLERDEFAULTIMPL1_H
00015 #define SAMPLERDEFAULTIMPL1_H
00016
00017 #include "Sampler_if.h"
00018 //namespace GenesysKernel {
00019
00020     class SamplerDefaultImpl1 : public Sampler_if {
00021     public:
00022         struct DefaultImpl1RNG_Parameters : public RNG_Parameters {
00023             unsigned int seed = 666;
00024             unsigned int module = 2147483647;
00025             unsigned int multiplier = 950706376;
00026             ~DefaultImpl1RNG_Parameters() = default;
00027         };
00028     public:
00029         SamplerDefaultImpl1();
00030         virtual ~SamplerDefaultImpl1() = default;
00031     public: // probability distributions
00032         virtual double random();
00033         virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit);
00034         virtual double sampleDiscrete(double acumProb, double value, ...);
00035         virtual double sampleErlang(double mean, int M);
  
```

```

00037     virtual double sampleExponential(double mean);
00038     virtual double sampleGamma(double mean, double alpha);
00039     virtual double sampleLogNormal(double mean, double stddev);
00040     virtual double sampleNormal(double mean, double stddev);
00041     virtual double sampleTriangular(double min, double mode, double max);
00042     virtual double sampleUniform(double min, double max);
00043     virtual double sampleWeibull(double alpha, double scale);
00044 public:
00045     void reset();
00046 public:
00047     virtual void setRNGparameters(RNG_Parameters* param);
00048     virtual RNG_Parameters* getRNGparameters() const;
00049 private:
00050     double _gammaJong(double alpha);
00051 private:
00052     RNG_Parameters* _param = new DefaultImpl1RNG_Parameters();
00053     unsigned int _xi;
00054     //bool _normalflag;
00055 };
00056 //namespace\\}
00057 #endif /* SAMPLERDEFAULTIMPL1_H */
00058

```

9.382 ScenarioExperiment_if.h File Reference

Classes

- class [ScenarioExperiment_if](#)

9.383 ScenarioExperiment_if.h

```

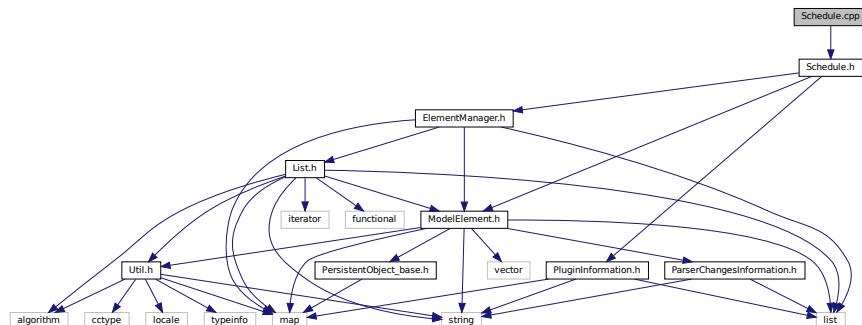
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ScenarioExperiment_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 10 de Outubro de 2018, 14:52
00012 */
00013
00014 #ifndef SCENARIOEXPERIMENT_IF_H
00015 #define SCENARIOEXPERIMENT_IF_H
00016
00017 class ScenarioExperiment_if {
00018 };
00019
00020 #endif /* SCENARIOEXPERIMENT_IF_H */
00021

```

9.384 Schedule.cpp File Reference

#include "Schedule.h"

Include dependency graph for Schedule.cpp:



9.385 Schedule.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Schedule.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:16
00012 */
00013
00014 #include "Schedule.h"
00015
00016 Schedule::Schedule(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Schedule>(),
00017   name) {
00018 }
00019 std::string Schedule::show() {
00020 }
00021
00022 PluginInformation* Schedule::GetPluginInformation() {
00023 }
00024
00025 ModelElement* Schedule::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026 }
00027
00028 bool Schedule::_loadInstance(std::map<std::string, std::string>* fields) {
00029 }
00030
00031 std::map<std::string, std::string>* Schedule::_saveInstance() {
00032 }
00033
00034 bool Schedule::_check(std::string* errorMessage) {
00035 }
00036

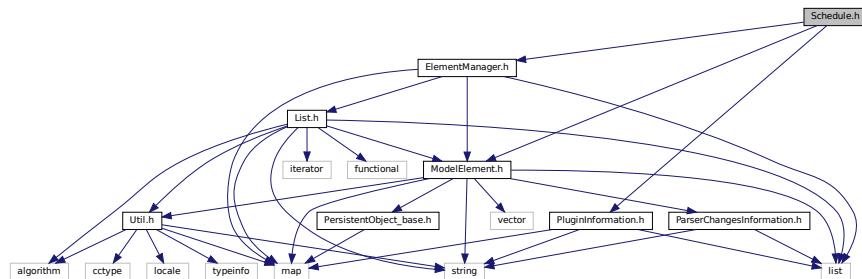
```

9.386 Schedule.h File Reference

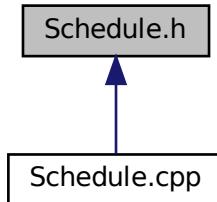
```

#include "ModelElement.h"
#include "ElementManager.h"
#include "PluginInformation.h"
Include dependency graph for Schedule.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [Schedule](#)

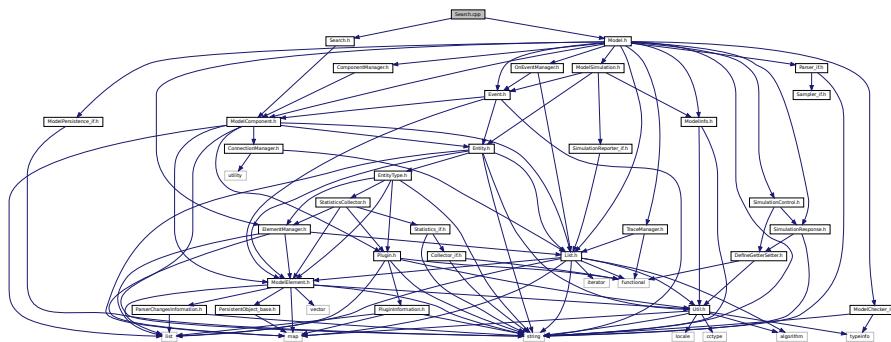
9.387 Schedule.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Schedule.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:16
00012 */
00013
00014 #ifndef SCHEDULE_H
00015 #define SCHEDULE_H
00016
00017 class Model;
00018
00068 #include "ModelElement.h"
00069 #include "ElementManager.h"
00070 #include "PluginInformation.h"
00071
00072 class Schedule : public ModelElement {
00073 public: // constructors
00074     Schedule(Model* model, std::string name = "");
00075     virtual ~Schedule() = default;
00076 public: // virtual
00077     virtual std::string show();
00078 public: // static
00079     static PluginInformation* GetPluginInformation();
00080     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00081 public:
00082 protected:
00083     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00084     virtual std::map<std::string, std::string>* _saveInstance();
00085     virtual bool _check(std::string* errorMessage);
00086 private:
00087 };
00088
00089 #endif /* SCHEDULE_H */
```

9.388 Search.cpp File Reference

```
#include "Search.h"
```

```
#include "Model.h"
Include dependency graph for Search.cpp:
```



9.389 Search.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Search.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Search.h"
00015 #include "Model.h"
00016
00017 Search::Search(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Search>(), name) {
00018 }
00019
00020 std::string Search::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* Search::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     Search* newComponent = new Search(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
00033
00034 void Search::_execute(Entity* entity) {
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
00038
00039 bool Search::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;
00045 }
00046
00047 void Search::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Search::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool Search::_check(std::string* errorMessage) {
00057     bool resultAll = true;
00058     //...
```

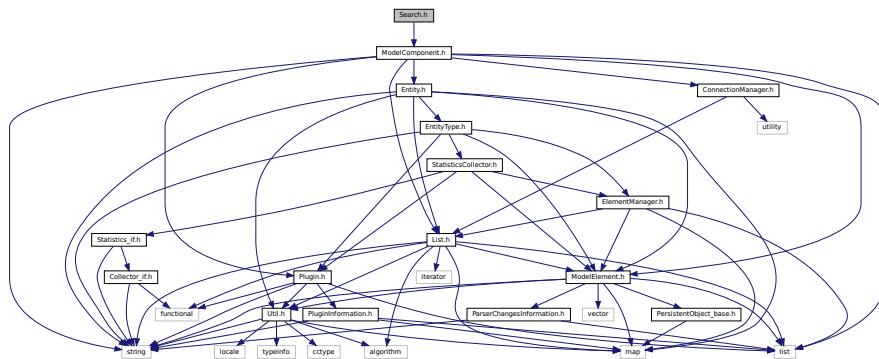
```

00059     return resultAll;
00060 }
00061
00062 PluginInformation* Search::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Search>(), &Search::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067
00068

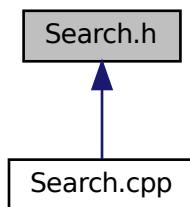
```

9.390 Search.h File Reference

#include "ModelComponent.h"
 Include dependency graph for Search.h:



This graph shows which files directly or indirectly include this file:



Classes

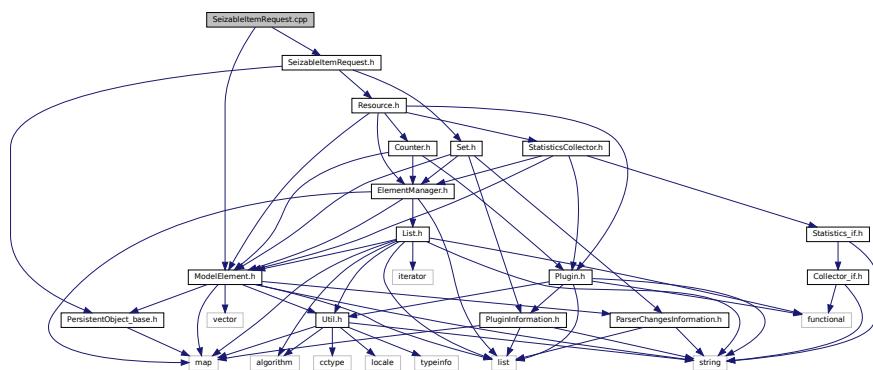
- class [Search](#)

9.391 Search.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: Search.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 03 de Junho de 2019, 15:20  
00012 */  
00013  
00014 #ifndef SEARCH_H  
00015 #define SEARCH_H  
00016  
00017 #include "ModelComponent.h"  
00018  
00055 class Search : public ModelComponent {  
00056 public: // constructors  
00057     Search(Model* model, std::string name = "");  
00058     virtual ~Search() = default;  
00059 public: // virtual  
00060     virtual std::string show();  
00061 public: // static  
00062     static PluginInformation* GetPluginInformation();  
00063     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);  
00064 protected: // virtual  
00065     virtual void _execute(Entity* entity);  
00066     virtual void _initBetweenReplications();  
00067     virtual bool _loadInstance(std::map<std::string, std::string>* fields);  
00068     virtual std::map<std::string, std::string>* _saveInstance();  
00069     virtual bool _check(std::string* errorMessage);  
00070 private: // methods  
00071 private: // attributes 1:1  
00072 private: // attributes 1:n  
00073 };  
00074  
00075  
00076 #endif /* SEARCH_H */  
00077
```

9.392 SeizableItemRequest.cpp File Reference

```
#include "SeizableItemRequest.h"  
#include "ModelElement.h"  
Include dependency graph for SeizableItemRequest.cpp:
```



9.393 SeizableItemRequest.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.
```

```

00005  */
00006
00007 /*
00008 * File: SeizableItemRequest.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 10 de abril de 2021, 08:45
00012 */
00013
00014 #include "SeizableItemRequest.h"
00015 #include "ModelElement.h"
00016
00017 SeizableItemRequest::SeizableItemRequest(ModelElement* resourceOrSet, std::string quantityExpression,
    SeizableItemRequest::ResourceType resourceType, SeizableItemRequest::SelectionRule selectionRule,
    std::string saveAttribute, unsigned int index) {
00018     _resourceType = resourceType;
00019     _resourceOrSet = resourceOrSet;
00020     _quantityExpression = quantityExpression;
00021     _selectionRule = selectionRule;
00022     _saveAttribute = saveAttribute;
00023     _index = index;
00024 }
00025
00026 bool SeizableItemRequest::_loadInstance(std::map<std::string, std::string>* fields) {
00027     bool res = true;
00028     try {
00029         _resourceType = static_cast<SeizableItemRequest::ResourceType>
            (std::stoi((*(fields->find("resourceType"))).second));
00030         _resourceName = ((*(fields->find("resourceName"))).second);
00031         //Resource* resource = dynamic_cast<Resource*>
            (_parentModel->getElements()->getElement(Util::TypeOf<Resource>(), resourceName));
00032         _quantityExpression = ((*(fields->find("quantity"))).second);
00033         _selectionRule = static_cast<SeizableItemRequest::SelectionRule>
            (std::stoi((*(fields->find("rule"))).second));
00034         _saveAttribute = ((*(fields->find("saveAttribute"))).second);
00035         _index = std::stoi((*(fields->find("index"))).second);
00036     } catch (const std::exception& e) {
00037         res = false;
00038     }
00039     return res;
00040 }
00041
00042 std::map<std::string, std::string>* SeizableItemRequest::_saveInstance() {
00043     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00044     fields->emplace("resourceType", std::to_string(static_cast<int>(_resourceType)));
00045     fields->emplace("resourceId", std::to_string(_resourceOrSet->getId()));
00046     fields->emplace("resourceName", _resourceOrSet->getName());
00047     fields->emplace("quantityExpression", _quantityExpression);
00048     fields->emplace("selectionRule", std::to_string(static_cast<int>(_selectionRule)));
00049     fields->emplace("saveAttribute", _saveAttribute);
00050     fields->emplace("index", std::to_string(_index));
00051     return fields;
00052 }
00053
00054 std::string SeizableItemRequest::show() {
00055     return "resourceType=" + std::to_string(static_cast<int>(_resourceType)) + ",resource=" +
        _resourceOrSet->getName() + "\",quantityExpression=\"" + _quantityExpression + "\", selectionRule=" +
        std::to_string(static_cast<int>(_selectionRule)) + ", _saveAttribute=\"" + _saveAttribute +
        "\",index=" + std::to_string(_index);
00056 }
00057
00058 void SeizableItemRequest::setIndex(unsigned int index) {
00059     this->_index = _index;
00060 }
00061
00062 unsigned int SeizableItemRequest::getIndex() const {
00063     return _index;
00064 }
00065
00066 void SeizableItemRequest::setSaveAttribute(std::string saveAttribute) {
00067     this->_saveAttribute = _saveAttribute;
00068 }
00069
00070 std::string SeizableItemRequest::getSaveAttribute() const {
00071     return _saveAttribute;
00072 }
00073
00074 void SeizableItemRequest::setSelectionRule(SeizableItemRequest::SelectionRule selectionRule) {
00075     this->_selectionRule = _selectionRule;
00076 }
00077
00078 SeizableItemRequest::SelectionRule SeizableItemRequest::getSelectionRule() const {
00079     return _selectionRule;
00080 }
00081
00082 void SeizableItemRequest::setQuantityExpression(std::string quantityExpression) {
00083     this->_quantityExpression = _quantityExpression;

```

```

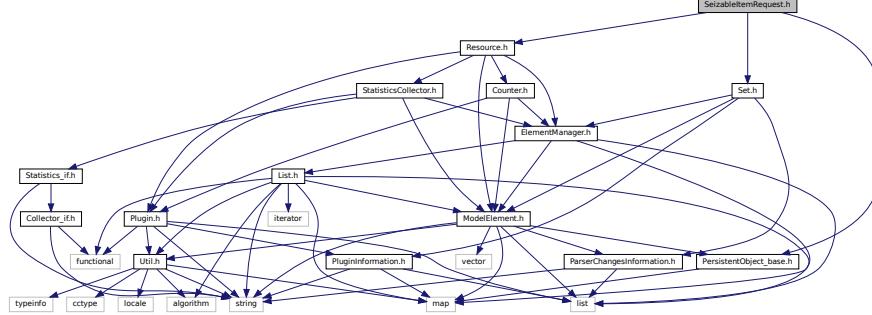
00084 }
00085
00086 std::string SeizableItemRequest::getQuantityExpression() const {
00087     return _quantityExpression;
00088 }
00089
00090 std::string SeizableItemRequest::getResourceName() const {
00091     return _resourceName;
00092 }
00093
00094 void SeizableItemRequest::setResource(Resource* resource) {
00095     this->_resourceOrSet = resource;
00096 }
00097
00098 Resource* SeizableItemRequest::getResource() const {
00099     return static_cast<Resource*> (_resourceOrSet);
00100 }
00101
00102 void SeizableItemRequest::setSet(Set* set) {
00103     this->_resourceOrSet = set;
00104 }
00105
00106 Set* SeizableItemRequest::getSet() const {
00107     return static_cast<Set*> (_resourceOrSet);
00108 }
00109
00110 void SeizableItemRequest::setResourceType(SeizableItemRequest::ResourceType resourceType) {
00111     this->_resourceType = _resourceType;
00112 }
00113
00114 SeizableItemRequest::ResourceType SeizableItemRequest::getResourceType() const {
00115     return _resourceType;
00116 }
00117
00118 void SeizableItemRequest::setLastMemberSeized(unsigned int lastMemberSeized) {
00119     this->_lastMemberSeized = lastMemberSeized;
00120 }
00121
00122 unsigned int SeizableItemRequest::getLastMemberSeized() const {
00123     return _lastMemberSeized;
00124 }
00125

```

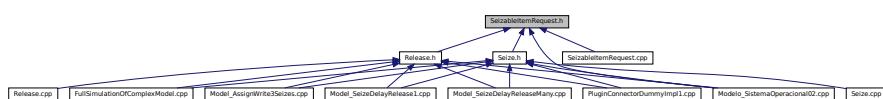
9.394 SeizableItemRequest.h File Reference

```
#include "PersistentObject_base.h"
#include "Resource.h"
#include "Set.h"
```

Include dependency graph for SeizableItemRequest.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SeizableItemRequest](#)

9.395 SeizableItemRequest.h

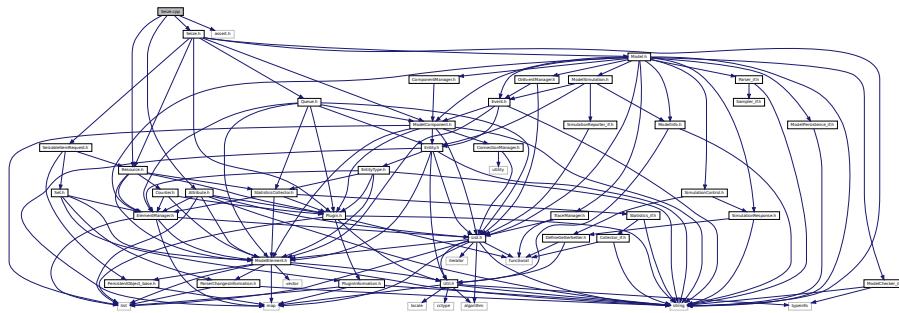
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SeizableItemRequest.h
00009  * Author: rlcancian
00010 *
00011 * Created on 10 de abril de 2021, 08:45
00012 */
00013
00014 #ifndef SEIZABLEITEMREQUEST_H
00015 #define SEIZABLEITEMREQUEST_H
00016
00017 #include "PersistentObject_base.h"
00018 #include "Resource.h"
00019 #include "Set.h"
00020
00021 class SeizableItemRequest : PersistentObject_base {
00022 public:
00023
00024     enum class SelectionRule : int {
00025         CYCLICAL = 1, RANDOM = 2, SPECIFICMEMBER = 3, LARGESTREMAININGCAPACITY = 4, SMALLESTNUMBERBUSY
00026         = 5
00027     };
00028
00029     enum class ResourceType : int {
00030         RESOURCE = 1, SET = 2
00031     };
00032
00033 public:
00034     SeizableItemRequest(ModelElement* resourceOrSet, std::string quantityExpression = "1",
00035                         SeizableItemRequest::ResourceType resourceType = SeizableItemRequest::ResourceType::RESOURCE,
00036                         SeizableItemRequest::SelectionRule selectionRule =
00037                         SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY, std::string saveAttribute = "",
00038                         unsigned int index = 0);
00039
00040     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00041     virtual std::map<std::string, std::string>* _saveInstance();
00042     std::string show();
00043     void setIndex(unsigned int index);
00044     unsigned int getIndex() const;
00045     void setSaveAttribute(std::string saveAttribute);
00046     std::string getSaveAttribute() const;
00047     void setSelectionRule(SelectionRule selectionRule);
00048     SelectionRule getSelectionRule() const;
00049     void setQuantityExpression(std::string quantityExpression);
00050     std::string getQuantityExpression() const;
00051     std::string getResourceName() const;
00052     void setResource(Resource* resource);
00053     Resource* getResource() const;
00054     void setSet(Set* set);
00055     Set* getSet() const;
00056     void setResourceType(ResourceType resourceType);
00057     ResourceType getResourceType() const;
00058     void setLastMemberSeized(unsigned int lastMemberSeized);
00059     unsigned int getLastMemberSeized() const;
00060
00061 private:
00062     ResourceType _resourceType;
00063     ModelElement* _resourceOrSet;
00064     std::string _resourceName;
00065     std::string _quantityExpression;
00066     SelectionRule _selectionRule;
00067     std::string _saveAttribute;
00068     unsigned int _index;
00069     unsigned int _lastMemberSeized = 0;
00070
00071 #endif /* SEIZABLEITEMREQUEST_H */
00072

```

9.396 Seize.cpp File Reference

```
#include "Seize.h"
#include "Resource.h"
#include "Attribute.h"
#include <assert.h>
Include dependency graph for Seize.cpp:
```



9.397 Seize.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Seize.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #include "Seize.h"
00015 #include "Resource.h"
00016 #include "Attribute.h"
00017 #include <assert.h>
00018
00019 Seize::Seize(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Seize>(), name) {
00020 }
00021
00022 std::string Seize::show() {
00023     std::string txt = ModelComponent::show() +
00024         "priority=" + std::to_string(_priority) +
00025         "seizeRequests={";
00026     unsigned short i = 0;
00027     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++, i++) {
00028         txt += "request" + std::to_string(i) + "[" + _seizeRequests->show() + "]";
00029     }
00030     txt = txt.substr(0, txt.length() - 1) + "}";
00031     return txt;
00032 }
00033
00034 void Seize::setPriority(unsigned short _priority) {
00035     this->_priority = _priority;
00036 }
00037
00038 unsigned short Seize::getPriority() const {
00039     return _priority;
00040 }
00041
00042 void Seize::setAllocationType(unsigned int _allocationType) {
00043     this->_allocationType = _allocationType;
00044 }
00045
00046 unsigned int Seize::getAllocationType() const {
00047     return _allocationType;
00048 }
00049
00050 void Seize::setQueueName(std::string queueName) throw () {
```

```

00051     Queue* queue = dynamic_cast<Queue*>
00052     (_parentModel->getElements()->getElement(Util::TypeOf<Queue>(), queueName));
00053     if (queue != nullptr) {
00054         _queue = queue;
00055     } else {
00056         throw std::invalid_argument("Queue does not exist");
00057     }
00058
00059 void Seize::_handlerForResourceEvent(Resource* resource) {
00060     Waiting* first = _queue->first();
00061     if (first != nullptr) { // there are entities waiting in the queue
00062         // find quantity requested for such resource
00063         for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++) {
00064             if ((*it)->getResource() == resource) {
00065                 unsigned int quantity = _parentModel->parseExpression((*it)->getQuantityExpression());
00066                 if ((resource->getCapacity() - resource->getNumberBusy()) >= quantity) { //enought
00067                     quantity to seize
00068                     double tnow = _parentModel->getSimulation()->getSimulatedTime();
00069                     resource->seize(quantity, tnow);
00070                     _parentModel->getFutureEvents()->insert(new Event(tnow, first->getEntity(),
00071                         this->getNextComponents()->getFrontConnection()));
00072                     _queue->removeElement(first);
00073                     _parentModel->getTracer()->traceSimulation(tnow, first->getEntity(), this,
00074                         "Waiting entity " + std::to_string(first->getEntity()->entityNumber()) + " now seizes " +
00075                         std::to_string(quantity) + " elements of resource \\" + resource->getName() + "\\");
00076                 }
00077             }
00078     }
00079     //void Seize::setResourceName(std::string resourceName) throw () {
00080     //    Resource* resource = dynamic_cast<Resource*>
00081     //        (_parentModel->elements()->element(Util::TypeOf<Resource>(), resourceName));
00082     //    if (resource != nullptr) {
00083     //        _resource = resource;
00084     //    } else {
00085     //        throw std::invalid_argument("Resource does not exist");
00086     //    }
00087     //std::string Seize::getResourceName() const {
00088     //    return _seizeRequest->resource()->name();
00089     //}
00090
00091     std::string Seize::getQueueName() const {
00092         return _queue->getName();
00093     }
00094
00095     //void Seize::setResource(Resource* resource) {
00096     //    this->_resource = resource;
00097     //
00098     //    _resource->addReleaseResourceEventHandler(Resource::SetResourceEventHandler<Seize>(&Seize:://_handlerForResourceEvent,
00099     //    this));
00100    //}
00101
00102    //Resource* Seize::getResource() const {
00103    //    return _resource;
00104    //}
00105
00106    void Seize::setQueue(Queue* queue) {
00107        this->_queue = queue;
00108    }
00109
00110    Queue* Seize::getQueue() const {
00111        return _queue;
00112    }
00113    List<SeizableItemRequest*>* Seize::getSeizeRequests() const {
00114        return _seizeRequests;
00115    }
00116    void Seize::_execute(Entity* entity) {
00117        for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++) {
00118            Resource* resource = (*it)->getResource();
00119            unsigned int quantity = _parentModel->parseExpression((*it)->getQuantityExpression());
00120            if (resource->getCapacity() - resource->getNumberBusy() < quantity) { // not enought free
00121                quantity to allocate. Entity goes to the queue
00122                WaitingResource* waitingRec = new WaitingResource(entity, this,
00123                    _parentModel->getSimulation()->getSimulatedTime(), quantity);
00124                this->_queue->insertElement(waitingRec); // ->list()->insert(waitingRec);
00125
00126                _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(), entity,
00127                    this, "Entity starts to wait for resource in queue \\" + _queue->getName() + "\\ with " +

```

```

    std::to_string(_queue->size()) + " elements";
00124     return;
00125 } else { // alocate the resource
00126     resource->seize(quantity, _parentModel->getSimulation()->getSimulatedTime());
00127
00128     _parentModel->getTracer()->traceSimulation(_parentModel->getSimulation()->getSimulatedTime(), entity,
00129     this, "Entity seizes " + std::to_string(quantity) + " elements of resource \'"
00130     + resource->getName() + "\' (capacity:" + std::to_string(resource->getCapacity()) + ", numberbusy:" +
00131     std::to_string(resource->getNumberBusy()) + ")");
00132 }
00133
00134     _parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(), 0.0);
00135 }
00136
00137 void Seize::_initBetweenReplications() {
00138     this->_queue->initBetweenReplications();
00139     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++) {
00140         (*it)->setLastMemberSeized(0);
00141         (*it)->getResource()->initBetweenReplications();
00142     }
00143 }
00144
00145 bool Seize::_loadInstance(std::map<std::string, std::string>* fields) {
00146     bool res = ModelComponent::_loadInstance(fields);
00147     if (res) {
00148         this->_allocationType = std::stoi(loadField(fields, "allocationType", "0"));
00149         this->_priority = std::stoi(loadField(fields, "priority", "0"));
00150         //Util::identitification queueId = std::stoi((*(fields->find("queueId"))).second);
00151         //Queue* queue = dynamic_cast<Queue*> (_model->elements()->element(Util::TypeOf<Queue>(),
00152         queueId));
00153         std::string queueName = ((*(fields->find("queueName"))).second);
00154         Queue* queue = dynamic_cast<Queue*>
00155             (_parentModel->getElements()->getElement(Util::TypeOf<Queue>(), queueName));
00156         this->_queue = queue;
00157         //Util::identitification resourceId = std::stoi((*(fields->find("resourceId"))).second);
00158         //Resource* resource = dynamic_cast<Resource*>
00159             (_model->elements()->element(Util::TypeOf<Resource>(), resourceId));
00160
00161         // \todo: next fields form a pair, and it should be a list of them
00162         unsigned short numRequests = std::stoi((*(fields->find("seizeRequestSize"))).second);
00163         for (unsigned short i = 0; i < numRequests; i++) {
00164             //std::string resRequest = ((*(fields->find("resourceItemRequest"))).second);
00165             SeizableItemRequest::ResourceType resourceType =
00166                 static_cast<SeizableItemRequest::ResourceType>(std::stoi(loadField(fields, "resourceType" +
00167                     std::to_string(i), std::to_string(static_cast<int> (SeizableItemRequest::ResourceType::RESOURCE))));;
00168             std::string resourceName = ((*(fields->find("resourceName" + std::to_string(i)))).second);
00169             Resource* resource = dynamic_cast<Resource*>
00170                 (_parentModel->getElements()->getElement(Util::TypeOf<Resource>(), resourceName));
00171             std::string quantityExpression = loadField(fields, "quantity" + std::to_string(i), "1");
00172             SeizableItemRequest::SelectionRule rule = static_cast<SeizableItemRequest::SelectionRule>
00173                 (std::stoi(loadField(fields, "selectionRule" + std::to_string(i), std::to_string(static_cast<int> (SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)))));
00174             std::string saveAttribute = loadField(fields, "saveAttribute" + std::to_string(i), "");
00175             unsigned int index = std::stoi(loadField(fields, "index" + std::to_string(i), "0"));
00176             this->_seizeRequests->insert(new SeizableItemRequest(resource, quantityExpression,
00177                 resourceType, rule, saveAttribute, index));
00178
00179             resource->addReleaseResourceEventHandler(Resource::SetResourceEventHandler<Seize>(&Seize::_handlerForResourceEvent,
00180             this));
00181         }
00182     }
00183     return res;
00184 }
00185
00186 std::map<std::string, std::string>* Seize::_saveInstance() {
00187     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00188     //Util::TypeOf<Seize>();
00189     if (_allocationType != 0) fields->emplace("allocationType",
00190         std::to_string(this->_allocationType));
00191     if (_priority != 0) fields->emplace("priority", std::to_string(this->_priority));
00192     fields->emplace("queueId", std::to_string(this->_queue->getId()));
00193     fields->emplace("queueName", " \"" + (this->_queue->getName()) + "\"");
00194     // \todo: put together as ResourceItemRequest ans it should be a list of them
00195     //
00196     fields->emplace("seizeRequestSize", std::to_string(_seizeRequests->size()));
00197     unsigned short i = 0;
00198     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++, i++) {
00199         //fields->emplace("resourceItemRequest" + std::to_string(i), "{" +
00200         map2str((*it)->_saveInstance()) + "}");
00201         if ((*it)->getResourceType() != SeizableItemRequest::ResourceType::RESOURCE)
00202             fields->emplace("resourceType" + std::to_string(i), std::to_string(static_cast<int> ((*it)->getResourceType())));
00203             fields->emplace("resourceId" + std::to_string(i),
00204                 std::to_string((*it)->getResource()->getId()));

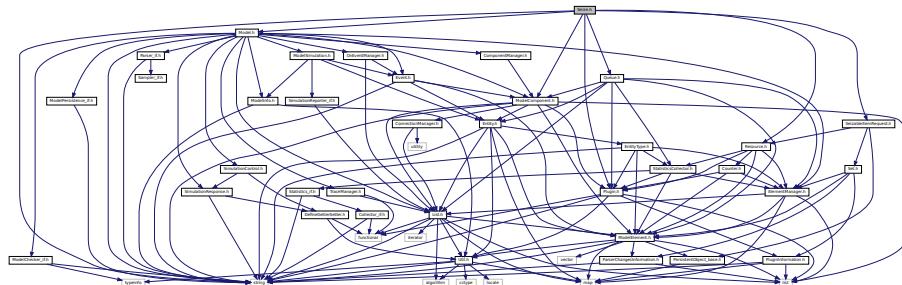
```

```

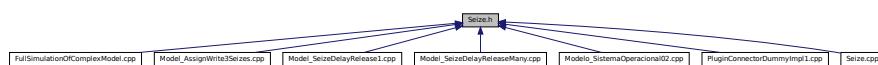
00187     fields->emplace("resourceName" + std::to_string(i), ((*it)->getResource()->getName()));
00188     if ((*it)->getQuantityExpression() != "1") fields->emplace("quantity" + std::to_string(i),
00189         "\\" + (*it)->getQuantityExpression() + "\\");
00190     if ((*it)->getSelectionRule() != SeizableItemRequest::SelectionRule::LARGESTREMAININGCAPACITY)
00191         fields->emplace("selectionRule" + std::to_string(i), std::to_string(static_cast<int>
00192             ((*it)->getSelectionRule())));
00193     if ((*it)->getSaveAttribute() != "") fields->emplace("saveAttribute" + std::to_string(i), "\\"
00194         + (*it)->getSaveAttribute() + "\\");
00195     fields->emplace("index" + std::to_string(i), std::to_string((*it)->getIndex()));
00196 }
00197 return fields;
00198
00199 bool Seize::_check(std::string* errorMessage) {
00200     bool resultAll = true;
00201     for (std::list<SeizableItemRequest*>::iterator it = _seizeRequests->list()->begin(); it != _seizeRequests->list()->end(); it++) {
00202         resultAll &= _parentModel->checkExpression((*it)->getQuantityExpression(), "quantity",
00203             errorMessage);
00204         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Resource>(),
00205             (*it)->getResource(), "Resource", errorMessage);
00206         resultAll &= _parentModel->getElements()->check(Util::TypeOf<Queue>(), _queue, "Queue",
00207             errorMessage);
00208         // \todo implement check of each ResourceItemRequest
00209         //resultAll &= _parentModel->getElements()->check(Util::TypeOf<Attribute>(),
00210         //    _saveAttribute,
00211         //    "SaveAttribute", false, errorMessage);
00212     }
00213     return resultAll;
00214 }
00215 ModelComponent* Seize::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00216     Seize* newComponent = new Seize(model);
00217     try {
00218         newComponent->_loadInstance(fields);
00219     } catch (const std::exception& e) {
00220     }
00221     return newComponent;
00222 }
00223
00224 }
```

9.398 Seize.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Model.h"
#include "Resource.h"
#include "Queue.h"
#include "Plugin.h"
#include "SeizableItemRequest.h"
Include dependency graph for Seize.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class WaitingResource
- class Seize

9.399 Seize.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Seize.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #ifndef SEIZE_H
00015 #define SEIZE_H
00016
00017 #include <string>
00018 #include "ModelComponent.h"
00019 #include "Model.h"
00020 #include "Resource.h"
00021 #include "Queue.h"
00022 #include "Plugin.h"
00023 #include "SeizableItemRequest.h"
00024
00025 class WaitingResource : public Waiting {
00026 public:
00027
00028     WaitingResource(Entity* entity, ModelComponent* component, double timeStartedWaiting, unsigned int
00029                     quantity) : Waiting(entity, component, timeStartedWaiting) {
00030         _quantity = quantity;
00031     }
00032     WaitingResource(const WaitingResource& orig) : Waiting(orig) {
00033     }
00034
00035     virtual ~WaitingResource() = default;
00036 public:
00037
00038     virtual std::string show() {
00039         return Waiting::show() +
00040                 ",quantity=" + std::to_string(this->_quantity);
00041     }
00042 public:
00043
00044     unsigned int getQuantity() const {
00045         return _quantity;
00046     }
00047 private:
00048     unsigned int _quantity;
00049 };
00050
00125 class Seize : public ModelComponent {
00126 public:
00127     Seize(Model* model, std::string name = "");
00128     virtual ~Seize() = default;
00129 public:
00130     virtual std::string show();
00131 public:
00132     static PluginInformation* GetPluginInformation();
00133     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00134 public: // get & set
00135     void setPriority(unsigned short _priority);
00136     unsigned short getPriority() const;
  
```

```

00137 void setAllocationType(unsigned int _allocationType);
00138 unsigned int getAllocationType() const;
00139 // indirect access to Queue* and Resource*
00140 //void setResourceName(std::string _resourceName) throw ();
00141 //std::string getResourceName() const;
00142 void setQueueName(std::string queueName) throw ();
00143 std::string getQueueName() const;
00144 void setQueue(Queue* queue);
00145 Queue* getQueue() const;
00146 List<SeizableItemRequest*>* getSeizeRequests() const;
00147 protected:
00148 virtual void _execute(Entity* entity);
00149 virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00150 virtual void _initBetweenReplications();
00151 virtual std::map<std::string, std::string>* _saveInstance();
00152 virtual bool _check(std::string* errorMessage);
00153 //virtual void _createInternalElements();
00154 private:
00155 void _handlerForResourceEvent(Resource* resource);
00156 private:
00157 unsigned int _allocationType = 0; // uint ? enum?
00158 unsigned short _priority = 0;
00159 Queue* _queue; // usually has a queue, but not always (it could be a hold)
00160 List<SeizableItemRequest*>* _seizeRequests = new List<SeizableItemRequest*>();
00161 private: // not gets or sets
00162 // unsigned int _lastMemberSeized = 0; // now _seizeRequest is a list and it was moved to
00163 // SeizableItemRequest
00163 };
00164
00165 #endif /* SEIZE_H */
00166

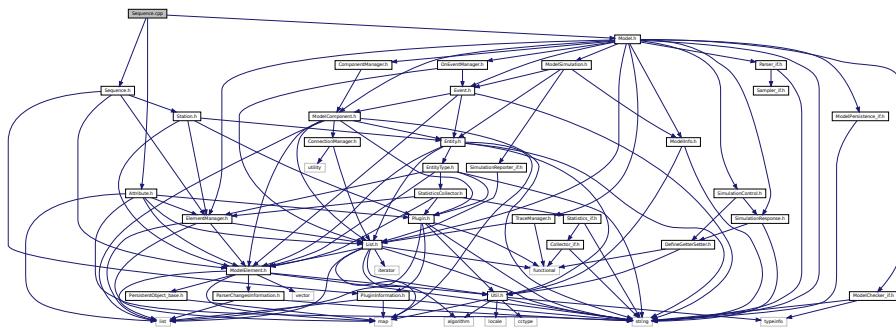
```

9.400 Sequence.cpp File Reference

```

#include "Sequence.h"
#include "Attribute.h"
#include "Model.h"
Include dependency graph for Sequence.cpp:

```



9.401 Sequence.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Sequence.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #include "Sequence.h"
00015 #include "Attribute.h"
00016 #include "Model.h"
00017

```

```

00018 Sequence::Sequence(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Sequence>(),
00019     name) {
00020
00021     std::string Sequence::show() {
00022         std::string msg = ModelElement::show();
00023         return msg;
00024     }
00025
00026     PluginInformation* Sequence::GetPluginInformation() {
00027         PluginInformation* info = new PluginInformation(Util::TypeOf<Sequence>(),
00028             &Sequence::LoadInstance);
00029         return info;
00030     }
00031     ModelElement* Sequence::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00032         Sequence* newElement = new Sequence(model);
00033         try {
00034             newElement->_loadInstance(fields);
00035         } catch (const std::exception& e) {
00036
00037         }
00038         return newElement;
00039     }
00040
00041     List<SequenceStep*>* Sequence::getSteps() const {
00042         return _steps;
00043     }
00044
00045     bool Sequence::_loadInstance(std::map<std::string, std::string>* fields) {
00046         bool res = ModelElement::_loadInstance(fields);
00047         if (res) {
00048             try {
00049             } catch (...) {
00050             }
00051         }
00052         return res;
00053     }
00054
00055     std::map<std::string, std::string>* Sequence::_saveInstance() {
00056         std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00057         //Util::TypeOf<Sequence>());
00058         return fields;
00059     }
00060     bool Sequence::_check(std::string* errorMessage) {
00061         /* include attributes needed */
00062         std::vector<std::string> neededNames = {"Entity.Sequence", "Entity.SequenceStep"};
00063         std::string neededName;
00064         for (unsigned int i = 0; i < neededNames.size(); i++) {
00065             neededName = neededNames[i];
00066             if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr)
00067             {
00068                 Attribute* attr1 = new Attribute(_parentModel, neededName);
00069                 //_parentModel->insert(attr1);
00070             }
00071             //
00072             return true;
00073     }
00074

```

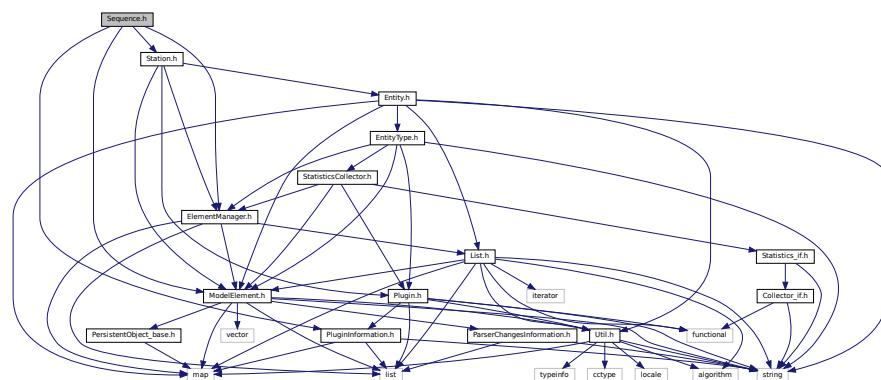
9.402 Sequence.h File Reference

```

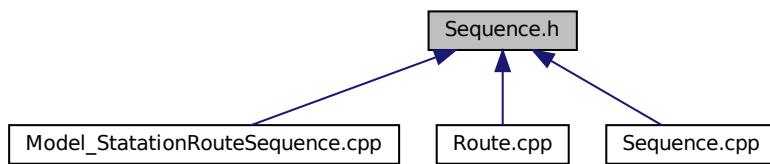
#include "ModelElement.h"
#include "ElementManager.h"
#include "PluginInformation.h"
#include "Station.h"

```

Include dependency graph for Sequence.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SequenceStep](#)
- class [Sequence](#)

9.403 Sequence.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Sequence.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #ifndef SEQUENCE_H
00015 #define SEQUENCE_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "PluginInformation.h"
00020 #include "Station.h"
00021
00022 class SequenceStep {
00023 public:
00024     SequenceStep(Station* station, std::list<std::string>* assignments = nullptr) {

```

```

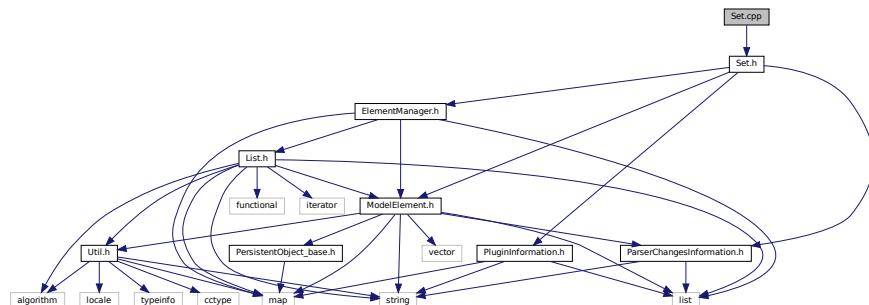
00026     _station = station;
00027     if (assignments == nullptr)
00028         _assignments = new std::list<std::string>();
00029     else
00030         _assignments = assignments;
00031 }
00032
00033 std::list<std::string>* getAssignments() const {
00034     return _assignments;
00035 }
00036
00037 void setStation(Station* _station) {
00038     this->_station = _station;
00039 }
00040
00041 Station* getStation() const {
00042     return _station;
00043 }
00044 private:
00045     Station* _station;
00046     std::list<std::string>* _assignments;
00047 };
00048
00074 class Sequence : public ModelElement {
00075 public:
00076
00077 public:
00078     Sequence(Model* model, std::string name = "");
00079     virtual ~Sequence() = default;
00080
00081 public:
00082     virtual std::string show();
00083 public: // static
00084     static PluginInformation* GetPluginInformation();
00085     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00086     List<SequenceStep*>* getSteps() const;
00087 public:
00088 protected:
00089     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00090     virtual std::map<std::string, std::string>* _saveInstance();
00091     virtual bool _check(std::string* errorMessage);
00092 private:
00093     List<SequenceStep*>* _steps = new List<SequenceStep*>();
00094 };
00095
00096 #endif /* SEQUENCE_H */
00097

```

9.404 Set.cpp File Reference

#include "Set.h"

Include dependency graph for Set.cpp:



9.405 Set.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates

```

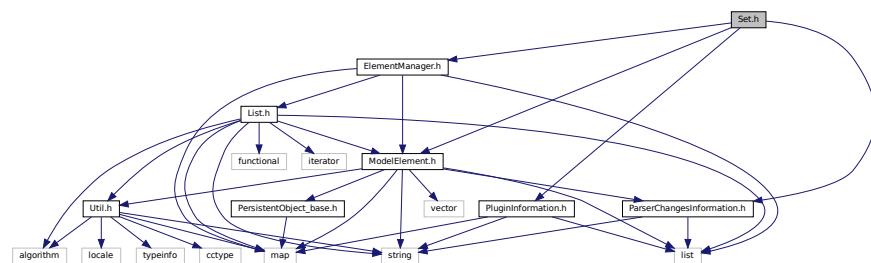
```

00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Set.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:11
00012 */
00013
00014 #include "Set.h"
00015
00016 Set::Set(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Set>(), name) {
00017 }
00018
00019 std::string Set::show() {
00020     return ModelElement::show() +
00021         "";
00022 }
00023
00024 void Set::setSetOfType(std::string _setOfType) {
00025     this->_setOfType = _setOfType;
00026 }
00027
00028 std::string Set::getSetOfType() const {
00029     return _setOfType;
00030 }
00031
00032 List<ModelElement*>* Set::getElementSet() const {
00033     return _elementSet;
00034 }
00035
00036 PluginInformation* Set::GetPluginInformation() {
00037     PluginInformation* info = new PluginInformation(Util::TypeOf<Set>(), &Set::LoadInstance);
00038     return info;
00039 }
00040
00041 ModelElement* Set::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00042     Set* newElement = new Set(model);
00043     try {
00044         newElement->_loadInstance(fields);
00045     } catch (const std::exception& e) {
00046     }
00047     return newElement;
00048 }
00049
00050
00051 bool Set::_loadInstance(std::map<std::string, std::string>* fields) {
00052     bool res = ModelElement::_loadInstance(fields);
00053     if (res) {
00054         try {
00055             //this->_attributeName = (*fields->find("attributeName")).second;
00056             //this->_orderRule = static_cast<OrderRule>
00057             (std::stoi((*fields->find("orderRule")).second));
00058         } catch (...) {
00059         }
00060     }
00061     return res;
00062 }
00063
00064 std::map<std::string, std::string>* Set::_saveInstance() {
00065     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00066     //Util::TypeOf<Set>());
00067     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00068     //fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00069     return fields;
00070 }
00071
00072 bool Set::_check(std::string* errorMessage) {
00073     bool resultAll = true;
00074     // resultAll |= ...
00075 }
00076
00077 ParserChangesInformation* Set::_getParserChangesInformation() {
00078     ParserChangesInformation* changes = new ParserChangesInformation();
00079     //changes->getProductionToAdd()->insert(...);
00080     //changes->getTokensToAdd()->insert(...);
00081 }
00082

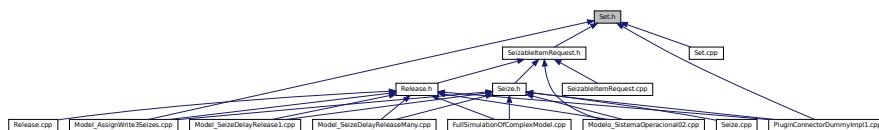
```

9.406 Set.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
Include dependency graph for Set.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Set

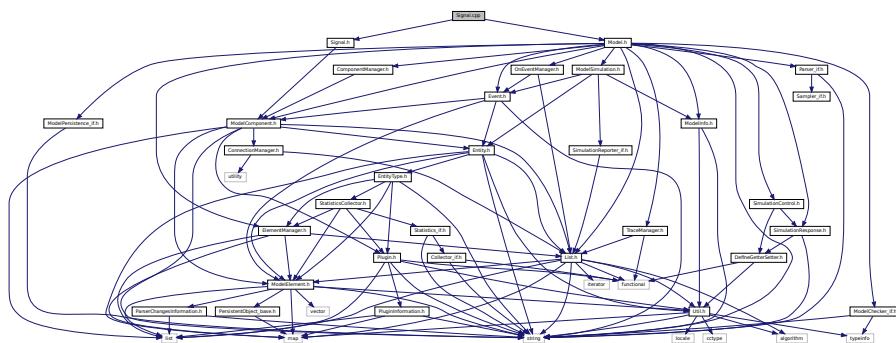
9.407 Set.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File: Set.h
00008  * Author: rlcancian
00009  *
00010  *
00011  * Created on 03 de Junho de 2019, 15:11
00012 */
00013
00014 #ifndef SET_H
00015 #define SET_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "ParserChangesInformation.h"
00020 #include "PluginInformation.h"
00021
00055 class Set : public ModelElement {
00056 public:
00057     Set(Model* model, std::string name = "");
00058     virtual ~Set() = default;
00059 public: // static
00060     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00061     static PluginInformation* GetPluginInformation();
00062 public:
```

```
00063     virtual std::string show();
00064 public:
00065     void setSetOfType(std::string _setOfType);
00066     std::string getSetOfType() const;
00067     List<ModelElement*>* getElementSet() const;
00068
00069 protected: // must be overridden by derived classes
00070     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00071     virtual std::map<std::string, std::string>* _saveInstance();
00072 protected: // could be overridden by derived classes
00073     virtual bool _check(std::string* errorMessage);
00074     virtual ParserChangesInformation* _getParserChangesInformation();
00075 private:
00076     //ElementManager* _elems;
00077     List<ModelElement*>* _elementSet = new List<ModelElement*>();
00078     std::string _setOfType;
00079 };
00080
00081 #endif /* SET_H */
00082
```

9.408 Signal.cpp File Reference

```
#include "Signal.h"
#include "Model.h"
Include dependency graph for Signal.cpp:
```



9.409 Signal.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Signal.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Signal.h"
00015
00016 #include "Model.h"
00017
00018 Signal::Signal(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Signal>(), name) {
00019 }
00020
00021 std::string Signal::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Signal::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Signal* newComponent = new Signal(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
```

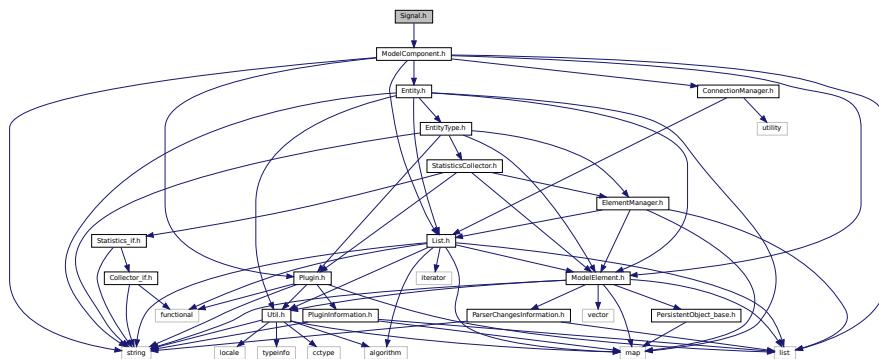
```

00030
00031     }
00032     return newComponent;
00033 }
00034
00035 void Signal::_execute(Entity* entity) {
00036     _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00038     0.0);
00039 }
00040 bool Signal::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Signal::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Signal::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Signal::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Signal::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Signal>(), &Signal::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

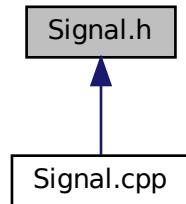
```

9.410 Signal.h File Reference

#include "ModelComponent.h"
Include dependency graph for Signal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Signal](#)

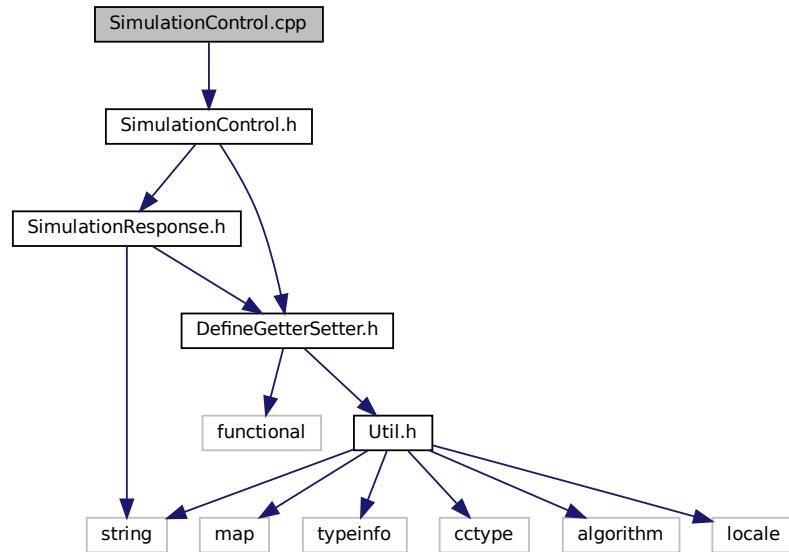
9.411 Signal.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Signal.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef SIGNAL_H
00015 #define SIGNAL_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Signal : public ModelComponent {
00020     public: // constructors
00021         Signal(Model* model, std::string name = "");
00022         virtual ~Signal() = default;
00023     public: // virtual
00024         virtual std::string show();
00025     public: // static
00026         static PluginInformation* GetPluginInformation();
00027         static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00028     protected: // virtual
00029         virtual void _execute(Entity* entity);
00030         virtual void _initBetweenReplications();
00031         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032         virtual std::map<std::string, std::string>* _saveInstance();
00033         virtual bool _check(std::string* errorMessage);
00034     private: // methods
00035     private: // attributes 1:1
00036     private: // attributes 1:n
00037 };
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059 #endif /* SIGNAL_H */
00060
  
```

9.412 SimulationControl.cpp File Reference

```
#include "SimulationControl.h"
Include dependency graph for SimulationControl.cpp:
```



9.413 SimulationControl.cpp

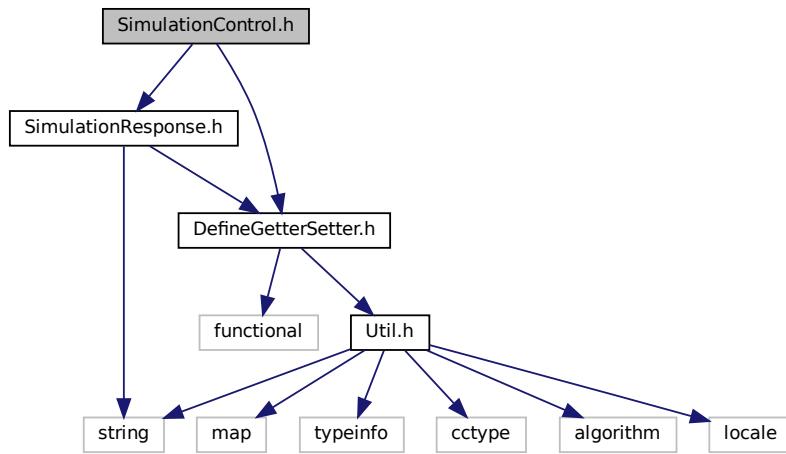
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationControl.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:01
00012 */
00013
00014 #include "SimulationControl.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 SimulationControl::SimulationControl(std::string type, std::string name, GetterMember getterMember,
00019     SetterMember setterMember) : SimulationResponse(type, name, getterMember) {
00020     this->_type = type;
00021     this->_setMemberFunction = setterMember;
00022 }
00023 std::string SimulationControl::show() {
00024     return "name=" + this->_name + ", type=" + this->_type;
00025 }
00026
00027 void SimulationControl::setValue(double value) {
00028     this->_setMemberFunction(value);
00029 }
00030
  
```

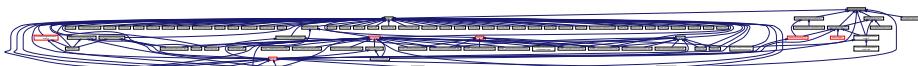
9.414 SimulationControl.h File Reference

```
#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](#)

9.415 `SimulationControl.h`

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SimulationControl.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 10 de Outubro de 2018, 18:01
00012 */
00013
00014 #ifndef SIMULATIONCONTROL_H
00015 #define SIMULATIONCONTROL_H
00016
00017 #include "SimulationResponse.h"
00018 #include "DefineGetterSetter.h"
00019 //namespace GenesysKernel {
00020
00024     class SimulationControl : public SimulationResponse {
00025     public:
00026         SimulationControl(std::string type, std::string name, GetterMember getterMember, SetterMember
00027                         setterMember);
00027         virtual ~SimulationControl() = default;
00028     public:
00029         std::string show();
00030     public:
00031         void setValue(double value);
  
```

```

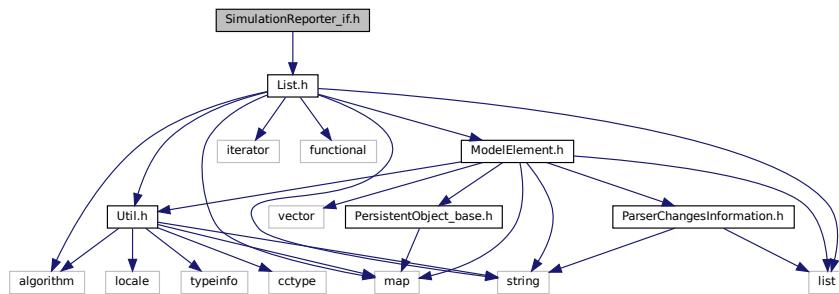
00032     private:
00033         SetterMember _setMemberFunction;
00034         //SetterMemberString _setMemberStringFunction;
00035     };
00036 //namespace\\}
00037 #endif /* SIMULATIONCONTROL_H */
00038

```

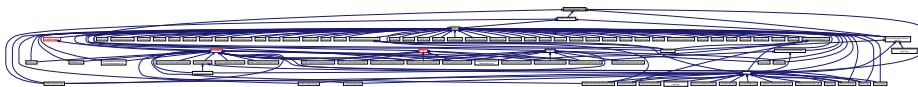
9.416 SimulationReporter_if.h File Reference

#include "List.h"

Include dependency graph for SimulationReporter_if.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SimulationReporter_if](#)

9.417 SimulationReporter_if.h

```

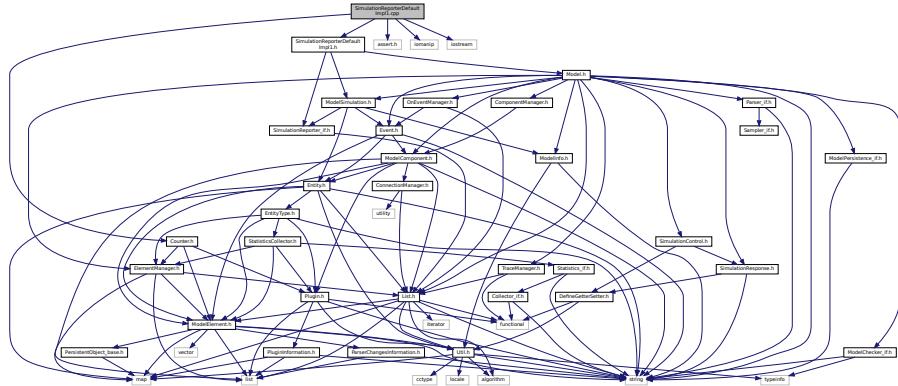
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SimulationReporter_if.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 8 de Agosto de 2018, 10:56
00012 */
00013
00014 #ifndef SIMULATIONREPORTER_IF_H
00015 #define SIMULATIONREPORTER_IF_H
00016
00017 #include "List.h"
00018 //#include "StatisticsCollector.h"
00019
00020 class SimulationReporter_if {
00021 public:
00022     virtual void showReplicationStatistics() = 0;
00023     virtual void showSimulationStatistics() = 0;
00024     virtual void showSimulationResponses() = 0;
00025     virtual void showSimulationControls() = 0;
00026 };
00027
00028 #endif /* SIMULATIONREPORTER_IF_H */
00029

```

9.418 SimulationReporterDefaultImpl1.cpp File Reference

```
#include "SimulationReporterDefaultImpl1.h"
#include <assert.h>
#include <iomanip>
#include <iostream>
#include "Counter.h"
```

Include dependency graph for SimulationReporterDefaultImpl1.cpp:



9.419 SimulationReporterDefaultImpl1.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationReporterDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 10:59
00012 */
00013
00014 #include "SimulationReporterDefaultImpl1.h"
00015 #include <assert.h>
00016 #include <iomanip>
00017 #include <iostream>
00018 #include "Counter.h"
00019
00020 //using namespace GenesysKernel;
00021
00022 SimulationReporterDefaultImpl1::SimulationReporterDefaultImpl1(ModelSimulation* simulation, Model*
00023     model, List<ModelElement*>* statsCountersSimulation) {
00024     _simulation = simulation;
00025     _model = model;
00026     _statsCountersSimulation = statsCountersSimulation;
00027 }
00028 void SimulationReporterDefaultImpl1::showSimulationControls() {
00029     _model->getTracer()->traceReport("Simulation Controls:");
00030     Util::IncIndent();
00031     {
00032         for (std::list<SimulationControl*>::iterator it = _model->getControls()->list()->begin(); it
00033 != _model->getControls()->list()->end(); it++) {
00034             SimulationControl* control = (*it);
00035             _model->getTracer()->traceReport(control->getName() + ": " +
00036                 std::to_string(control->getValue()));
00037         }
00038     }
00039     Util::DecIndent();
00040 }
00041 void SimulationReporterDefaultImpl1::showReplicationStatistics() {
00042     _model->getTracer()->traceReport("");
00043     _model->getTracer()->traceReport("Begin of Report for replication " +
00044         std::to_string(_simulation->getCurrentReplicationNumber()) + " of " +
00045         std::to_string(_model->getSimulation()->getNumberOfReplications()));
```

```

00043     /* \todo: StatisticsCollector and Counter should NOT be special classes. It should iterate classes
00044      looking for classes that can generate reports.
00045      StatisticsCollector and Counter should override an inherited attribute from ModelElement to
00046      specify they generate report information
00047      look for _generateReportInformation = true; using bool generateReportInformation() const;
00048      */
00049      const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00050      const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00051      // runs over all elements and list the statistics for each one, and then the statistics with no
00052      parent
00053      Util::IncIndent();
00054      //this->showSimulationControls();
00055      // copy the list of statistics and counters into a single new list
00056      std::list<ModelElement*>* statisticsAndCounters = new
00057      std::list<ModelElement*>(*(_model->getElements()->getElementList(UtilTypeOfStatisticsCollector)->list()));
00058      std::list<ModelElement*>* counters = new
00059      std::list<ModelElement*>(*(_model->getElements()->getElementList(UtilTypeOfCounter)->list()));
00060      statisticsAndCounters->merge(*counters);
00061      // organizes statistics into a map of maps
00062      std::map< std::string, std::map<std::string, std::list<ModelElement*>>> * mapMapTypeStat = new
00063      std::map<std::string, std::map<std::string, std::list<ModelElement*>>>();
00064      for (std::list<ModelElement*>::iterator it = statisticsAndCounters->begin(); it !=
00065      statisticsAndCounters->end(); it++) {
00066          std::string parentName, parentTypename;
00067          ModelElement* statOrCnt = (*it);
00068          //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00069          if ((*it)->getClassName() == UtilTypeOfStatisticsCollector) {
00070              StatisticsCollector* stat = dynamic_cast<StatisticsCollector*> (statOrCnt);
00071              parentName = stat->getParent()->getName();
00072              parentTypename = stat->getParent()->getClassName();
00073          } else {
00074              if ((*it)->getClassName() == UtilTypeOfCounter) {
00075                  Counter* cnt = dynamic_cast<Counter*> (statOrCnt);
00076                  parentName = cnt->getParent()->getName();
00077                  parentTypename = cnt->getParent()->getClassName();
00078              }
00079          }
00080          // look for key=parentTypename
00081          std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt =
00082          mapMapTypeStat->find(parentTypename);
00083          if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00084              std::pair< std::string, std::map<std::string, std::list<ModelElement*>>>* newPair = new
00085              std::pair<std::string, std::map<std::string, std::list<ModelElement*>>>(parentTypename, new
00086              std::map<std::string, std::list<ModelElement*>>());
00087              mapMapTypeStat->insert(*newPair);
00088              mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00089          }
00090          //assert(mapMapIt != mapMapTypeStat->end());
00091          std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00092          assert(mapTypeStat != nullptr);
00093          // look for key=parentName
00094          std::map<std::string, std::list<ModelElement*>>>::iterator mapIt =
00095          mapTypeStat->find(parentName);
00096          if (mapIt == mapTypeStat->end()) { // parentName does not exists in map. Include it.
00097              std::pair< std::string, std::list<ModelElement*>>>* newPair = new std::pair<std::string,
00098              std::list<ModelElement*>>(parentName, new std::list<ModelElement*>());
00099              mapTypeStat->insert(*newPair);
00100              mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00101          }
00102          // get the list and insert the stat in that list
00103          std::list<ModelElement*>* listStatAndCount = (*mapIt).second;
00104          assert(listStatAndCount != nullptr);
00105          listStatAndCount->insert(listStatAndCount->end(), statOrCnt);
00106          //_model->getTraceManager()->traceReport(parentTypename + " -> " + parentName + " -> " +
00107          stat->show());
00108      }
00109      // now runs over that map of maps showing the statistics
00110      Util::IncIndent();
00111      Util::IncIndent();
00112      _model->getTracer()->traceReport(Util::SetW("name", _nameW) + Util::SetW("elems", _w) +
Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) + Util::SetW("variance",
_w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
Util::SetW("confLevel", _w));
00113      Util::DecIndent();
00114      Util::DecIndent();
00115      for (auto const mapmapItem : *mapMapTypeStat) {
00116          _model->getTracer()->traceReport("Statistics for " + mapmapItem.first + ":");
00117          Util::IncIndent();
00118          {
00119              for (auto const mapItem : *(mapmapItem.second)) {
00120                  _model->getTracer()->traceReport(mapItem.first + ":");
00121                  Util::IncIndent();
00122                  {
00123                      // _model->getTracer()->traceReport(Util::SetW("name", _nameW) +
Util::SetW("elems", _w) + Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) +

```

```

    Util::SetW("variance", _w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) +
    Util::SetW("confInterv", _w) + Util::SetW("confLevel", _w));
00113     for (ModelElement * const item : *(mapItem.second)) {
00114         if (item->getclassname() == UtilTypeOfStatisticsCollector) {
00115             Statistics_if* stat = dynamic_cast<StatisticsCollector*>
00116             (item)->getStatistics();
00117             _model->getTracer()->traceReport(Util::TraceLevel::report,
00118                                         Util::SetW(item->getName() + std::string(_nameW, '.'), _nameW - 1)
00119                                         + " " +
00120                                         Util::SetW(std::to_string(stat->numElements()), _w) +
00121                                         Util::SetW(std::to_string(stat->min()), _w) +
00122                                         Util::SetW(std::to_string(stat->max()), _w) +
00123                                         Util::SetW(std::to_string(stat->average()), _w) +
00124                                         Util::SetW(std::to_string(stat->variance()), _w) +
00125                                         Util::SetW(std::to_string(stat->stddeviation()), _w) +
00126                                         Util::SetW(std::to_string(stat->variationCoef()), _w) +
00127                                         Util::SetW(std::to_string(stat->halfWidthConfidenceInterval()),
00128                                         _w) +
00129                                         Util::SetW(std::to_string(stat->getConfidenceLevel()), _w)
00130                                         );
00131         } else {
00132             if (item->getclassname() == UtilTypeOfCounter) {
00133                 Counter* count = dynamic_cast<Counter*> (item);
00134                 _model->getTracer()->traceReport(Util::TraceLevel::report,
00135                                         Util::SetW(count->getName() + std::string(_nameW, '.'), _nameW
00136                                         - 1) + " " +
00137                                         Util::SetW(std::to_string(count->getCountValue()), _w)
00138                                         );
00139             }
00140         }
00141     }
00142     Util::DecIndent();
00143 }
00144 //this->showSimulationResponses();
00145 Util::DecIndent();
00146 _model->getTracer()->traceReport("End of Report for replication " +
00147 std::to_string(_simulation->getCurrentReplicationNumber()) + " of " +
00148 std::to_string(_model->getSimulation()->getNumberOfReplications()));
00149 _model->getTracer()->traceReport("-----");
00150 void SimulationReporterDefaultImpl1::showSimulationResponses() {
00151     _model->getTracer()->traceReport("Simulation Responses:");
00152     Util::IncIndent();
00153     {
00154         for (std::list<SimulationResponse*>::iterator it = _model->getResponses()->list()->begin(); it
00155 != _model->getResponses()->list()->end(); it++) {
00156             SimulationResponse* response = (*it);
00157             _model->getTracer()->traceReport(response->getName() + ": " +
00158             std::to_string(response->getValue()));
00159         }
00160     }
00161     Util::DecIndent();
00162 void SimulationReporterDefaultImpl1::showSimulationStatistics() {//List<StatisticsCollector*>*
00163     cstatsSimulation) {
00164     _model->getTracer()->traceReport("");
00165     _model->getTracer()->traceReport("Begin of Report for Simulation (based on " +
00166     std::to_string(_model->getSimulation()->getNumberOfReplications()) + " replications)");
00167     const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00168     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00169     // runs over all elements and list the statistics for each one, and then the statistics with no
00170     parent
00171     Util::IncIndent();
00172     // COPY the list of statistics and counters into a single new list
00173     //std::list<ModelElement*>* statisticsAndCounters = /new
00174     std::list<ModelElement*>(*(this->_statsCountersSimulation->list()));
00175     // organizes statistics into a map of maps
00176     std::map< std::string, std::map<std::string, std::list<ModelElement*>*>* mapMapTypeStat = new
00177     std::map<std::string, std::map<std::string, std::list<ModelElement*>*>*();
00178
00179     for (std::list<ModelElement*>::iterator it = _statsCountersSimulation->list()->begin(); it !=
00180     _statsCountersSimulation->list()->end(); it++) {
00181         std::string parentName, parentTypename;
00182         ModelElement* statOrCnt = (*it);
00183         //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00184         if ((*it)->getclassname() == UtilTypeOfStatisticsCollector) {
00185             StatisticsCollector* stat = dynamic_cast<StatisticsCollector*> (statOrCnt);
00186             parentName = stat->getParent()->getName();
00187             parentTypename = stat->getParent()->getClassname();
00188         } else {
00189             if ((*it)->getclassname() == UtilTypeOfCounter) {

```

```

00184         Counter* cnt = dynamic_cast<Counter*> (statOrCnt);
00185         parentName = cnt->getParent()->getName();
00186         parentTypename = cnt->getParent()->getclassname();
00187     }
00188 }
00189 // look for key=parentTypename
00190 std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt =
00191 mapMapTypeStat->find(parentTypename);
00192 if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00193     std::pair< std::string, std::map<std::string, std::list<ModelElement*>>>* newPair = new
00194     std::pair<std::string, std::map<std::string, std::list<ModelElement*>>>(parentTypename, new
00195     std::map<std::string, std::list<ModelElement*>>());
00196     mapMapTypeStat->insert(*newPair);
00197     mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00198 }
00199 assert (mapMapIt != mapMapTypeStat->end());
00200 std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00201 assert (mapTypeStat != nullptr);
00202 // look for key=parentName
00203 std::map<std::string, std::list<ModelElement*>>>::iterator mapIt =
00204 mapTypeStat->find(parentName);
00205 if (mapIt == mapTypeStat->end()) { // parentName does not exists in map. Include it.
00206     std::pair< std::string, std::list<ModelElement*>>>* newPair = new std::pair<std::string,
00207     std::list<ModelElement*>> (parentName, new std::list<ModelElement*>());
00208     mapTypeStat->insert(*newPair);
00209     mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00210 }
00211 // get the list and insert the stat in that list
00212 std::list<ModelElement*>> listStat = (*mapIt).second;
00213 listStat->insert(listStat->end(), statOrCnt);
00214 //_model->getTraceManager()->traceReport (parentTypename + " -> " + parentName + " -> " +
00215 stat->show());
00216 }
00217 // now runs over that map of maps showing the statistics
00218 //int w = 12;
00219 Util::IncIndent();
00220 Util::IncIndent();
00221 _model->getTracer()->traceReport(Util::SetW("name", _nameW) + Util::SetW("elems", _w) +
00222 Util::SetW("min", _w) + Util::SetW("max", _w) + Util::SetW("average", _w) + Util::SetW("variance",
00223 _w) + Util::SetW("stddev", _w) + Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
00224 Util::SetW("confLevel", _w));
00225 Util::DecIndent();
00226 Util::DecIndent();
00227 for (auto const mapmapItem : *mapMapTypeStat) {
00228     _model->getTracer()->traceReport("Statistis for " + mapmapItem.first + ":" );
00229     Util::IncIndent();
00230     {
00231         for (auto const mapItem : *(mapmapItem.second)) {
00232             _model->getTracer()->traceReport(mapItem.first + ":" );
00233             Util::IncIndent();
00234             {
00235                 for (ModelElement * const item : *(mapItem.second)) {
00236                     if (item->getclassname() == UtilTypeOfStatisticsCollector) {
00237                         Statistics_if* stat = dynamic_cast<StatisticsCollector*>
00238                         (item)->getStatistics();
00239                         _model->getTracer()->traceReport(Util::TraceLevel::report,
00240                         Util::SetW(item->getName() + std::string(_nameW, '.'), _nameW - 1)
00241                         + " " +
00242                         Util::SetW(std::to_string(stat->numElements()), _w) +
00243                         Util::SetW(std::to_string(stat->min()), _w) +
00244                         Util::SetW(std::to_string(stat->max()), _w) +
00245                         Util::SetW(std::to_string(stat->average()), _w) +
00246                         Util::SetW(std::to_string(stat->variance()), _w) +
00247                         Util::SetW(std::to_string(stat->stddeviation()), _w) +
00248                         Util::SetW(std::to_string(stat->variationCoef()), _w) +
00249                         Util::SetW(std::to_string(stat->halfWidthConfidenceInterval()),
00250                         _w) +
00251                         Util::SetW(std::to_string(stat->getConfidenceLevel()), _w)
00252                         );
00253                 }
00254             }
00255         }
00256     }
00257     Util::DecIndent();

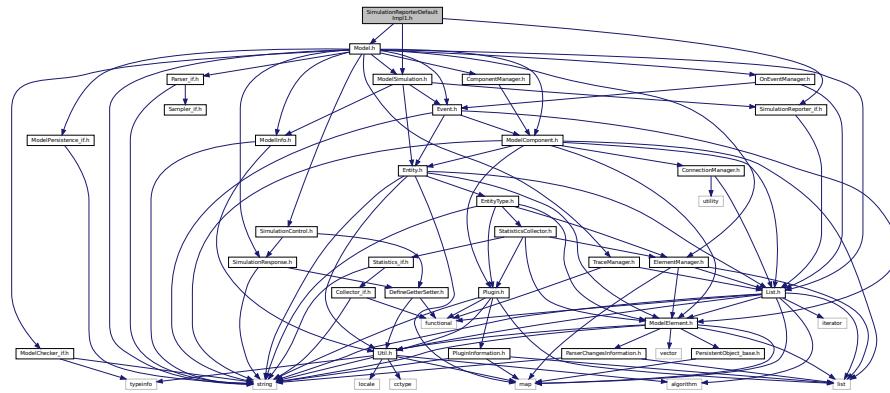
```

```

00258     }
00259
00260     Util::DecIndent();
00261     _model->getTracer()->traceReport("End of Report for Simulation");
00262 }
```

9.420 SimulationReporterDefaultImpl1.h File Reference

```
#include "SimulationReporter_if.h"
#include "ModelSimulation.h"
#include "Model.h"
Include dependency graph for SimulationReporterDefaultImpl1.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [SimulationReporterDefaultImpl1](#)

9.421 SimulationReporterDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SimulationReporterDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 8 de Agosto de 2018, 10:59
00012 */
00013
00014 #ifndef SIMULATIONREPORTERDEFAULTIMPL1_H
00015 #define SIMULATIONREPORTERDEFAULTIMPL1_H
00016
00017 #include "SimulationReporter_if.h"
00018 #include "ModelSimulation.h"
00019 #include "Model.h"
00020
```

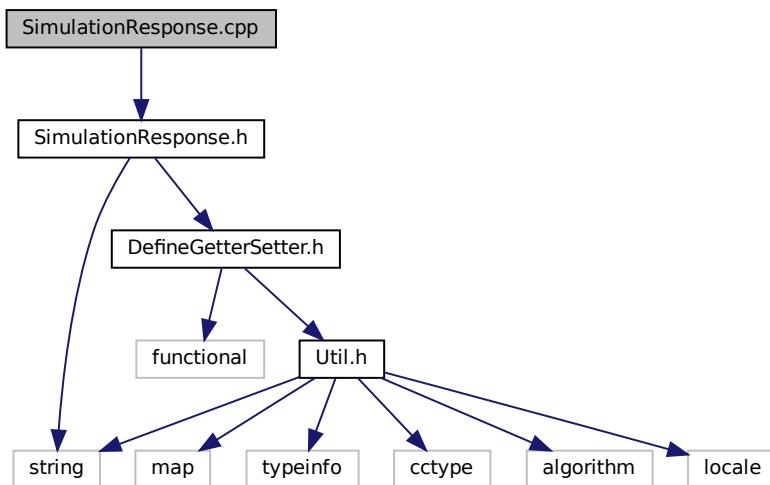
```

00021 //namespace GenesysKernel {
00022
00026     class SimulationReporterDefaultImpl1 : public SimulationReporter_if {
00027     public:
00028         SimulationReporterDefaultImpl1(ModelSimulation* simulation, Model* model, List<ModelElement*>*
00029         statsCountersSimulation);
00029         virtual ~SimulationReporterDefaultImpl1() = default;
00030     public:
00031         virtual void showReplicationStatistics();
00032         virtual void showSimulationStatistics();
00033         virtual void showSimulationResponses();
00034         virtual void showSimulationControls();
00035     private:
00036         ModelSimulation* _simulation;
00037         Model* _model;
00038     private:
00039         List<ModelElement*>* _statsCountersSimulation;
00040     private:
00041         const unsigned short _w = 12;
00042         const unsigned short _nameW = 40;
00043     };
00044 //namespace\\}
00045 /* SIMULATIONREPORTERDEFAULTIMPL1_H */
00046

```

9.422 SimulationResponse.cpp File Reference

#include "SimulationResponse.h"
Include dependency graph for SimulationResponse.cpp:



9.423 SimulationResponse.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationResponse.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 16:18
00012 */
00013
00014 #include "SimulationResponse.h"

```

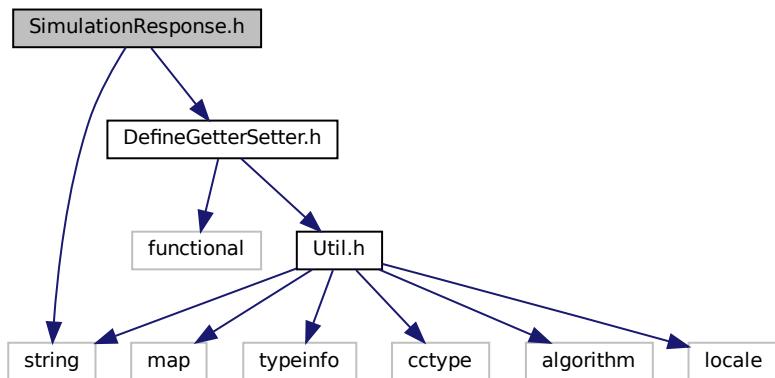
```

00015 //#include "AnalysisOfVariance_if.h"
00016
00017 //using namespace GenesysKernel;
00018
00019 SimulationResponse::SimulationResponse(std::string type, std::string name, GetterMember getterMember)
{
00020     _type = type;
00021     _name = name;
00022     _getterMemberFunction = getterMember;
00023 }
00024
00025 std::string SimulationResponse::show() {
00026     return "name=" + this->_name + ", type=" + this->_type;
00027 }
00028
00029 std::string SimulationResponse::getName() const {
00030     return _name;
00031 }
00032
00033 std::string SimulationResponse::getType() const {
00034     return _type;
00035 }
00036
00037 double SimulationResponse::getValue() {
00038     return this->_getterMemberFunction();
00039 }

```

9.424 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`

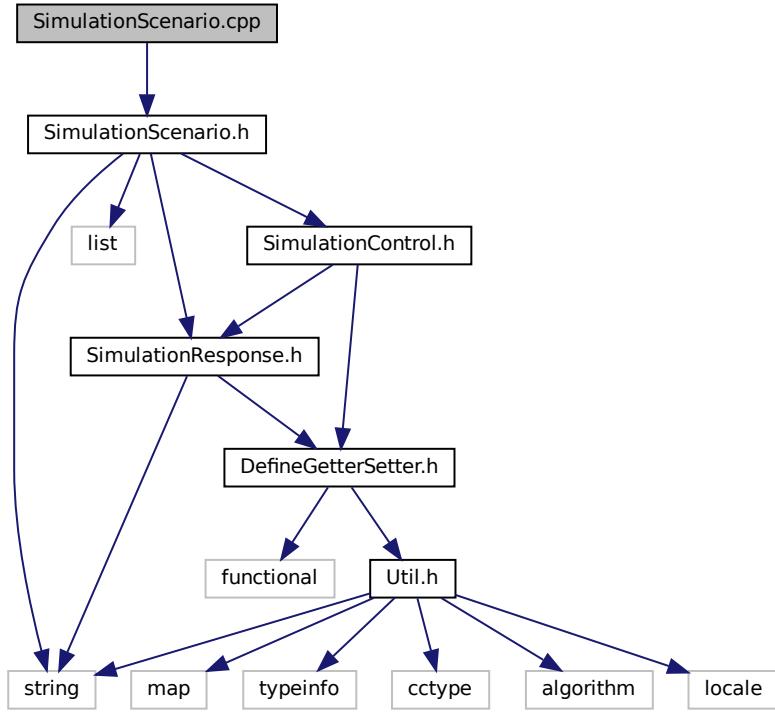
9.425 SimulationResponse.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SimulationResponse.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 16:18
00012 */
00013
00014 #ifndef SIMULATIONRESPONSE_H
00015 #define SIMULATIONRESPONSE_H
00016
00017 #include <string>
00018 #include "DefineGetterSetter.h"
00019
00020 //namespace GenesysKernel {
00021
00022 class SimulationResponse {
00023 public:
00024     SimulationResponse(std::string type, std::string name, GetterMember getterMember);
00025     virtual ~SimulationResponse() = default;
00026 public:
00027     std::string show();
00028 protected:
00029     double getValue();
00030     std::string getName() const;
00031     std::string getType() const;
00032     protected:
00033     std::string _type;
00034     std::string _name;
00035     GetterMember _getterMemberFunction;
00036 };
00037 //namespace\\}
00038 #endif /* SIMULATIONRESPONSE_H */
```

9.426 SimulationScenario.cpp File Reference

```
#include "SimulationScenario.h"
```

Include dependency graph for SimulationScenario.cpp:



9.427 SimulationScenario.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 /*
00008  * File:   SimulationScenario.cpp
00009  * Author: rafael.luiz.cancian
00010 /*
00011  * Created on 10 de Outubro de 2018, 18:21
00012 */
00013
00014 #include "SimulationScenario.h"
00015
00016 SimulationScenario::SimulationScenario() {
00017 }
00018
00019 bool SimulationScenario::startSimulation(std::string* errorMessage) {
00020     // model->loadmodel _modelName
00021     // set values for the _selectedControls
00022     // model->startSimulation
00023     // get the value of the _selectedResponses from the model and store results on _responseValues
00024     // clear selected controls and responses
00025     // close the model
00026     return false;
00027 }
00028
00029 void SimulationScenario::setScenarioName(std::string _name) {
00030     this->_scenarioName = _name;
00031 }
00032
00033 std::string SimulationScenario::getScenarioName() const {
00034     return _scenarioName;
00035 }
00036
00037 void SimulationScenario::setModelFilename(std::string _modelFilename) {
  
```

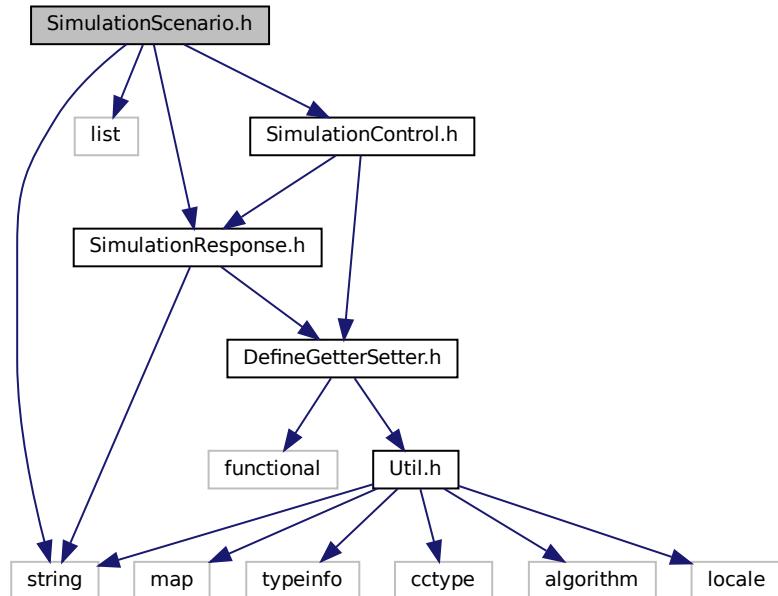
```

00038     this->_modelFilename = _modelFilename;
00039 }
00040
00041 std::string SimulationScenario::getModelFilename() const {
00042     return _modelFilename;
00043 }
00044
00045 std::list<std::string>* SimulationScenario::getSelectedResponses() const {
00046     return _selectedResponses;
00047 }
00048
00049 std::list<std::pair<std::string, double>>* SimulationScenario::getSelectedControls() const {
00050     return _selectedControls;
00051 }
00052
00053 void SimulationScenario::setScenarioDescription(std::string _scenarioDescription) {
00054     this->_scenarioDescription = _scenarioDescription;
00055 }
00056
00057 std::string SimulationScenario::getScenarioDescription() const {
00058     return _scenarioDescription;
00059 }

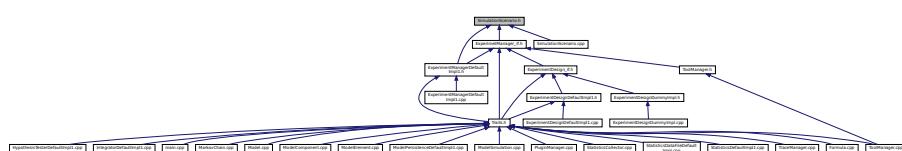
```

9.428 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](#)

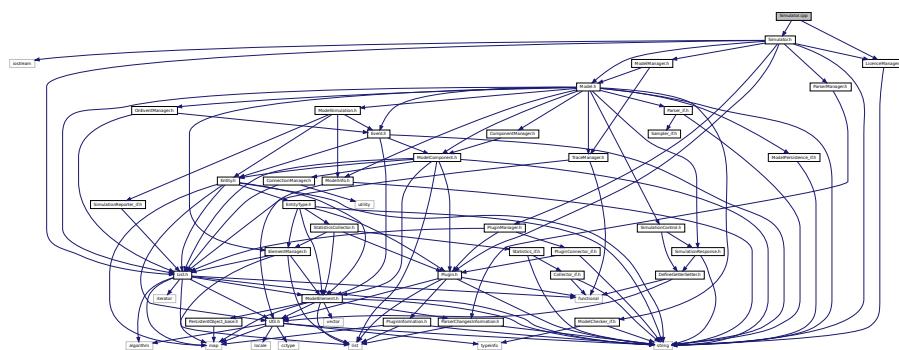
9.429 SimulationScenario.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SimulationScenario.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:21
00012 */
00013
00014 #ifndef SIMULATIONSCENARIO_H
00015 #define SIMULATIONSCENARIO_H
00016
00017 #include <string>
00018 #include <list>
00019 #include "SimulationResponse.h"
00020 #include "SimulationControl.h"
00021
00022 class SimulationScenario {
00023 public:
00024     SimulationScenario();
00025     virtual ~SimulationScenario() = default;
00026 public: // results
00027     bool startSimulation(std::string* errorMessage);
00028     std::list<std::pair<std::string, double>>* getResponseValues() const;
00029     std::list<std::pair<std::string, double>>* getControlValues() const;
00030     double getResponseValue(std::string responseName);
00031 public: // gets and sets
00032     void setModelFilename(std::string _modelFilename);
00033     std::string getModelFilename() const;
00034     void setScenarioName(std::string _name);
00035     std::string getScenarioName() const;
00036     void setScenarioDescription(std::string _scenarioDescription);
00037     std::string getScenarioDescription() const;
00038     std::list<std::pair<std::string, double>>* getSelectedControls() const; // access to the list to
00039     insert or remove controls
00040     double getControlValue(std::string controlName);
00041     void setControlValue(std::string controlName, double value);
00042     std::list<std::string>* getSelectedResponses() const; // access to the list to insert or remove
00043     responses
00044 private:
00045     std::string _scenarioName;
00046     std::string _scenarioDescription;
00047     std::string _modelFilename;
00048     std::list<std::pair<std::string, double>>* _selectedControls = new
00049         std::list<std::pair<std::string, double>>();
00050     std::list<std::string>* _selectedResponses = new std::list<std::string>();
00051     std::list<std::pair<std::string, double>>* _responseValues;
00052 };
00053
00054 #endif /* SIMULATIONSCENARIO_H */
00055
```

9.430 Simulator.cpp File Reference

```
#include "Simulator.h"
#include "LicenceManager.h"
```

Include dependency graph for Simulator.cpp:



Functions

- `GenesysSimulator CreateSimulator2 ()`
 - `Simulator * CreateSimulator ()`
 - `void DestroySimulator2 (GenesysSimulator p)`
 - `void DestroySimulator (Simulator *p)`

9.430.1 Function Documentation

9.430.1.1 CreateSimulator() `Simulator* CreateSimulator ()`

Definition at line 22 of file [Simulator.cpp](#).

```
00022                                     {  
00023     return new /*GenesysKernel::*/Simulator();  
00024 }
```

9.430.1.2 CreateSimulator2() GenesysSimulator CreateSimulator2 ()

Definition at line 18 of file [Simulator.cpp](#).

```
    return new /*GenesysKernel::*/Simulator();  
00020 }
```

9.430.1.3 DestroySimulator() void DestroySimulator (Simulator * p)

Definition at line 30 of file [Simulator.cpp](#).

```
Definition at line 30 of file Simulator.hpp.
00030                                         {
00031     delete p; // Can use a base class or derived class pointer here
00032 }
```

9.430.1.4 DestroySimulator2() void DestroySimulator2 (GenesysSimulator p)

Definition at line 26 of file Simulator.cpp.

```
00026     {
00027     delete p; // Can use a base class or derived class pointer here
00028 }
```

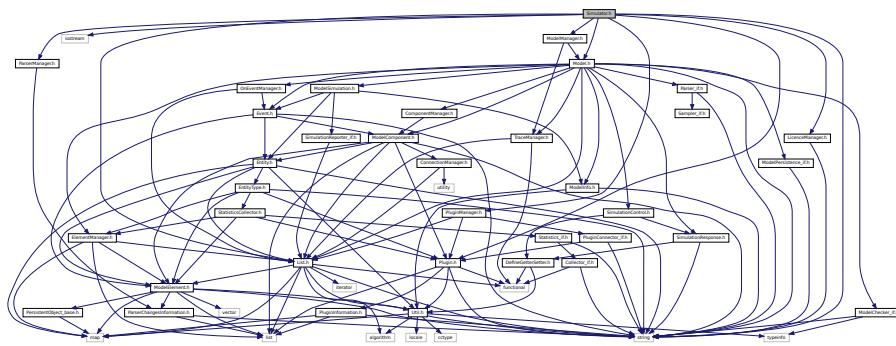
9.431 Simulator.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Genesys.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 12:48
00012 */
00013
00014 #include "Simulator.h"
00015 #include "LicenceManager.h"
00016 //#include "ToolManager.h"
00017
00018 extern "C" GenesysSimulator CreateSimulator2() {
00019     return new /*GenesysKernel::*/Simulator();
00020 }
00021
00022 extern "C" /*GenesysKernel::*/Simulator* CreateSimulator() {
00023     return new /*GenesysKernel::*/Simulator();
00024 }
00025
00026 extern "C" void DestroySimulator2(GenesysSimulator p) {
00027     delete p; // Can use a base class or derived class pointer here
00028 }
00029
00030 extern "C" void DestroySimulator(/*GenesysKernel::*/Simulator* p) {
00031     delete p; // Can use a base class or derived class pointer here
00032 }
00033
00034
00035 //using namespace GenesysKernel;
00036
00037 Simulator::Simulator() {
00038     // This is the ONLY method in the entire software where std::cout is allowed.
00039     std::cout << "STARTING " << _name << ", version " << getVersion() << std::endl;
00040     _licenceManager = new LicenceManager(this);
00041     _pluginManager = new PluginManager(this);
00042     _modelManager = new ModelManager(this);
00043     //_toolManager = new ToolManager(this);
00044     _traceManager = new TraceManager(this);
00045     std::cout << '|' << '\t' << _licenceManager->showLicence() << std::endl;
00046     //std::cout << '|' << '\t' << _licenceManager->showActivationCode() << std::endl;
00047     //std::cout << '|' << '\t' << _licenceManager->showLimits() << std::endl;
00048 }
00049
00050 PluginManager* Simulator::getPlugins() const {
00051     return _pluginManager;
00052 }
00053
00054 ModelManager* Simulator::getModels() const {
00055     return _modelManager;
00056 }
00057
00058 TraceManager* Simulator::getTracer() const {
00059     return _traceManager;
00060 }
00061
00062 ParserManager* Simulator::getParser() const {
00063     return _parserManager;
00064 }
00065
00066 std::string Simulator::getVersion() const {
00067     return std::to_string(_versionNumber) + " (" + _versionName + ")";
00068 }
00069
00070 unsigned int Simulator::getVersionNumber() const {
00071     return _versionNumber;
00072 }
00073
```

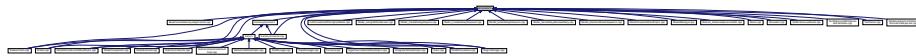
```
00074 std::string Simulator::getName() const {
00075     return _name;
00076 }
00077
00078 LicenceManager* Simulator::getLicenceManager() const {
00079     return _licenceManager;
00080 }
00081
```

9.432 Simulator.h File Reference

```
#include <string>
#include <iostream>
#include "Model.h"
#include "Plugin.h"
#include "List.h"
#include "LicenceManager.h"
#include "PluginManager.h"
#include "ModelManager.h"
#include "ParserManager.h"
Include dependency graph for Simulator.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Simulator

Typedefs

- `typedef Simulator * GenesysSimulator`

9.432.1 Typedef Documentation

9.432.1.1 GenesysSimulator `typedef Simulator* GenesysSimulator`

Definition at line 62 of file [Simulator.h](#).

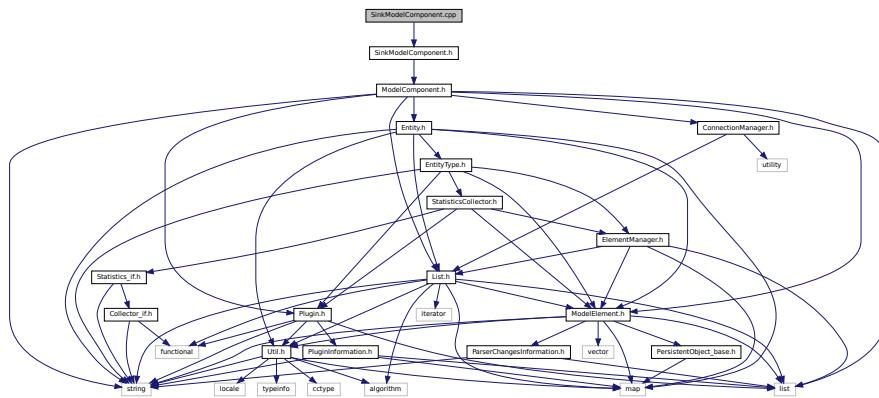
9.433 Simulator.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Genesys.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:48
00012 */
00013
00014 #ifndef GENESYS_H
00015 #define GENESYS_H
00016
00017 #include <string>
00018 #include <iostream>
00019 #include "Model.h"
00020 #include "Plugin.h"
00021 #include "List.h"
00022 #include "LicenceManager.h"
00023 #include "PluginManager.h"
00024 #include "ModelManager.h"
00025 //#include "ToolManager.h"
00026 #include "ParserManager.h"
00027
00028 //namespace GenesysKernel {
00029     class Simulator {
00030         typedef void (*eventHandler)();
00031     public:
00032         Simulator();
00033         virtual ~Simulator() = default;
00034     public: // only get
00035         std::string getVersion() const;
00036         unsigned int getVersionNumber() const;
00037         std::string getName() const;
00038         LicenceManager* getLicenceManager() const;
00039         PluginManager* getPlugins() const;
00040         ModelManager* getModels() const;
00041         TraceManager* getTracer() const;
00042         ParserManager* getParser() const;
00043     private:
00044         private: // attributes 1:1 objects
00045             LicenceManager* _licenceManager;
00046             PluginManager* _pluginManager;
00047             ModelManager* _modelManager;
00048             TraceManager* _traceManager;
00049             ParserManager* _parserManager;
00050         private: // attributes 1:1 native
00051             const std::string _name = "GenESyS - GENeric and Expansible SYstem Simulator";
00052             const std::string _versionName = "backtotrack";
00053             const unsigned int _versionNumber = 210408;
00054     };
00055 //namespace\\}
00056
00057 // passing a "C" class factory function which instantiates the class.
00058 #ifdef /*GENESYS_H*/
00059     typedef /*GenesysKernel::*/Simulator* GenesysSimulator;
00060 #endif /*GENESYS_H*/
00061
00062
00063
00064
```

9.434 SinkModelComponent.cpp File Reference

```
#include "SinkModelComponent.h"
```

Include dependency graph for SinkModelComponent.cpp:



9.435 SinkModelComponent.cpp

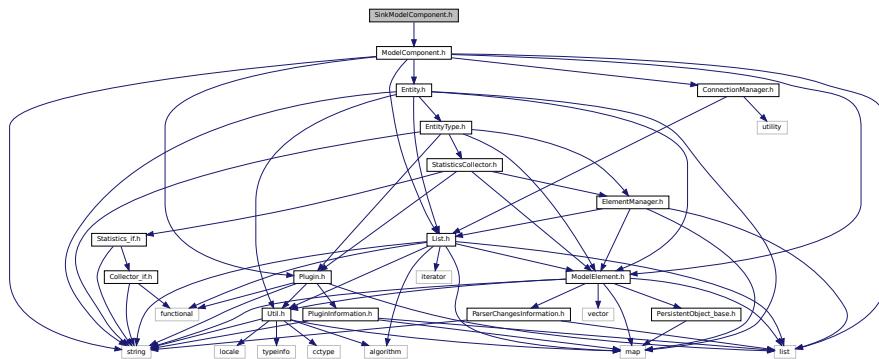
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SinkModelComponent.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 14:29
00012 */
00013
00014 #include "SinkModelComponent.h"
00015
00016 //using namespace GenesysKernel;
00017
00018 SinkModelComponent::SinkModelComponent(Model* model, std::string componentTypename, std::string name)
00019 : ModelComponent(model, componentTypename, name) {
00020
00021 bool SinkModelComponent::_loadInstance(std::map<std::string, std::string>* fields) {
00022     bool res = ModelComponent::_loadInstance(fields);
00023     if (res) {
00024         //this->_reportStatistics = std::stoi((*fields->find("collectStatistics")).second) != 0;
00025     }
00026     return res;
00027 }
00028
00029 void SinkModelComponent::_initBetweenReplications() {
00030 }
00031
00032 std::map<std::string, std::string>* SinkModelComponent::_saveInstance() {
00033     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00034     //fields->emplace("collectStatistics", std::to_string(this->_reportStatistics));
00035     return fields;
00036 }
00037
00038 bool SinkModelComponent::_check(std::string* errorMessage) {
00039     return true;
00040 }
```

9.436 SinkModelComponent.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for SinkModelComponent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SinkModelComponent](#)

9.437 SinkModelComponent.h

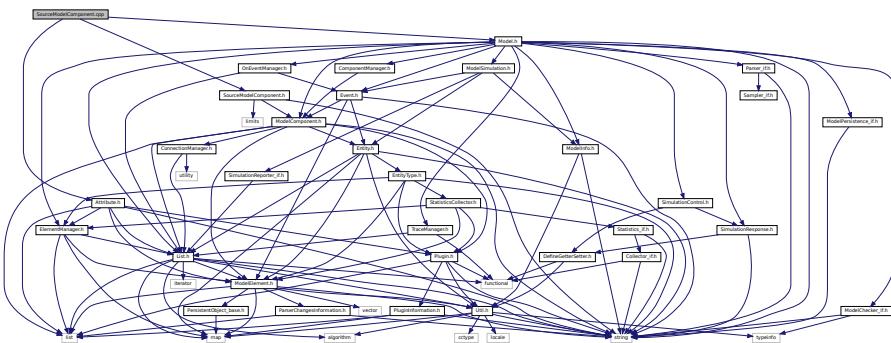
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:   SinkModelComponent.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 14:29
00012 */
00013
00014 #ifndef SINKMODELCOMPONENT_H
00015 #define SINKMODELCOMPONENT_H
00016
00017 #include "ModelComponent.h"
00018
00019 //namespace GenesysKernel {
00020
00025     class SinkModelComponent : public ModelComponent {
00026     public:
00027         SinkModelComponent(Model* model, std::string componentTypename, std::string name = "");
00028         virtual ~SinkModelComponent() = default;
00029     public:
00030     protected:
00031         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00032         virtual void _initBetweenReplications();
00033         virtual std::map<std::string, std::string>* _saveInstance();
00034         virtual bool _check(std::string* errorMessage);
00035     private:
00036     };
00037 //namespace\\}
00038 #endif /* SINKMODELCOMPONENT_H */
00039

```

9.438 SourceModelComponent.cpp File Reference

```
#include "SourceModelComponent.h"
#include "Model.h"
#include "Attribute.h"
Include dependency graph for SourceModelComponent.cpp:
```



9.439 SourceModelComponent.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SourceModelComponent.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:50
00012 */
00013
00014 #include "SourceModelComponent.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 //using namespace GenesysKernel;
00019
00020 SourceModelComponent::SourceModelComponent(Model* model, std::string componentTypename, std::string
name) : ModelComponent(model, componentTypename, name) {
00021 }
00022
00023 std::string SourceModelComponent::show() {
00024     std::string text = ModelComponent::show() +
00025         ",entityType=\"" + _entityType->getName() + "\"" +
00026         ",firstCreation=" + std::to_string(_firstCreation);
00027     return text;
00028 }
00029
00030 bool SourceModelComponent::_loadInstance(std::map<std::string, std::string>* fields) {
00031     bool res = ModelComponent::_loadInstance(fields);
00032     if (res) {
00033         this->_entitiesPerCreation = std::stoi(loadField(fields, "entitiesPerCreation", "1"));
00034         this->_firstCreation = std::stod(loadField(fields, "firstCreation", "0.0"));
00035         this->_timeBetweenCreationsExpression = (*fields->find("timeBetweenCreations")).second;
00036         this->_timeBetweenCreationsTimeUnit = static_cast<Util::TimeUnit>(std::stoi(loadField(fields,
"timeBetweenCreationsTimeUnit", std::to_string(static_cast<int>(Util::TimeUnit::second)))));
00037         this->_maxCreationsExpression = loadField(fields, "maxCreations",
std::to_string(std::numeric_limits<unsigned int>::max()));
00038         std::string entityTypename = (*fields->find("entityTypename")).second;
00039         this->_entityType = dynamic_cast<EntityType*>
(_parentModel->getElements()->getElement(Util::TypeOf<EntityType>(), entityTypename));
00040     }
00041     return res;
00042 }
00043
00044 void SourceModelComponent::_initBetweenReplications() {
00045     this->entitiesCreatedSoFar = 0;
00046 }
00047
```

```

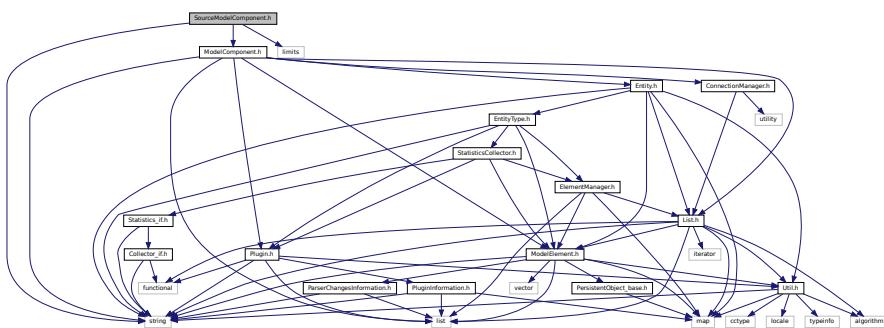
00048 std::map<std::string, std::string>* SourceModelComponent::_saveInstance() {
00049     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00050     if (_entitiesPerCreation != 1) fields->emplace("entitiesPerCreation",
00051         std::to_string(this->_entitiesPerCreation));
00052     if (_firstCreation != 0.0) fields->emplace("firstCreation", std::to_string(this->_firstCreation));
00053     if (_timeBetweenCreations != Util::TimeUnit::second)
00054         fields->emplace("timeBetweenCreationsTimeUnit", std::to_string(static_cast<int>
00055             (this->_timeBetweenCreationsTimeUnit)));
00056     if (_maxCreationsExpression != std::to_string(std::numeric_limits<unsigned int>::max()))
00057         fields->emplace("maxCreations", "\\" + this->_maxCreationsExpression + "\\");
00058     fields->emplace("entityTypename", (this->_entityType->getName())); // save the name
00059     //fields->emplace("collectStatistics", std::to_string(this->_collectStatistics));
00060     return fields;
00061 }
00062
00063 bool SourceModelComponent::_check(std::string* errorMessage) {
00064     /* include attributes needed */
00065     ElementManager* elements = _parentModel->getElements();
00066     std::vector<std::string> neededNames = {"Entity.ArrivalTime"};
00067     std::string neededName;
00068     for (unsigned int i = 0; i < neededNames.size(); i++) {
00069         neededName = neededNames[i];
00070         if (elements->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00071             Attribute* attr = new Attribute(_parentModel, neededName);
00072             elements->insert(attr);
00073         }
00074     }
00075     bool resultAll = true;
00076     resultAll &= _parentModel->getElements()->check(Util::TypeOf<EntityType>(), _entityType,
00077         "entitytype", errorMessage);
00078     resultAll &= _parentModel->checkExpression(this->_timeBetweenCreationsExpression, "time between
00079         creations", errorMessage);
00080     return resultAll;
00081 }
00082 void SourceModelComponent::setFirstCreation(double _firstCreation) {
00083     this->_firstCreation = _firstCreation;
00084 }
00085
00086 //void SourceModelComponent::setCollectStatistics(bool _collectStatistics) {
00087 //    this->_collectStatistics = _collectStatistics;
00088 //}
00089 //
00090 //bool SourceModelComponent::isCollectStatistics() const {
00091 //    return _collectStatistics;
00092 //}
00093
00094 void SourceModelComponent::setEntityType(EntityType* entityType) {
00095     _entityType = entityType;
00096 }
00097
00098 EntityType* SourceModelComponent::getEntityType() const {
00099     return _entityType;
00100 }
00101
00102 void SourceModelComponent::setTimeUnit(Util::TimeUnit _timeUnit) {
00103     this->_timeBetweenCreationsTimeUnit = _timeUnit;
00104 }
00105
00106 Util::TimeUnit SourceModelComponent::getTimeUnit() const {
00107     return this->_timeBetweenCreationsTimeUnit;
00108 }
00109
00110 void SourceModelComponent::setTimeBetweenCreationsExpression(std::string _timeBetweenCreations) {
00111     this->_timeBetweenCreationsExpression = _timeBetweenCreations;
00112 }
00113
00114 std::string SourceModelComponent::getTimeBetweenCreationsExpression() const {
00115     return _timeBetweenCreationsExpression;
00116 }
00117
00118 void SourceModelComponent::setMaxCreations(std::string _maxCreationsExpression) {
00119     this->_maxCreationsExpression = _maxCreationsExpression;
00120 }
00121
00122 std::string SourceModelComponent::getMaxCreations() const {
00123     return _maxCreationsExpression;
00124 }
00125
00126 unsigned int SourceModelComponent::getEntitiesCreated() const {
00127     return _entitiesCreatedSoFar;
00128 }

```

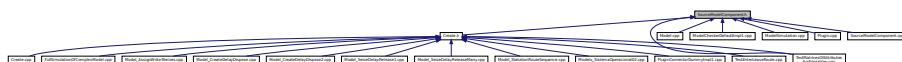
```
00129
00130 void SourceModelComponent::setEntitiesCreated(unsigned int _entitiesCreated) {
00131     this->_entitiesCreatedSoFar = _entitiesCreated;
00132 }
00133
00134 void SourceModelComponent::setEntitiesPerCreation(unsigned int _entitiesPerCreation) {
00135     this->_entitiesPerCreation = _entitiesPerCreation;
00136     //this->_entitiesCreatedSoFar = _entitiesPerCreation; // that's because "entitiesPerCreation"
00137     // entities are included in the future events list BEFORE replication starts (to initialize events list)
00138 }
00139 unsigned int SourceModelComponent::getEntitiesPerCreation() const {
00140     return _entitiesPerCreation;
00141 }
```

9.440 SourceModelComponent.h File Reference

```
#include "ModelComponent.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`

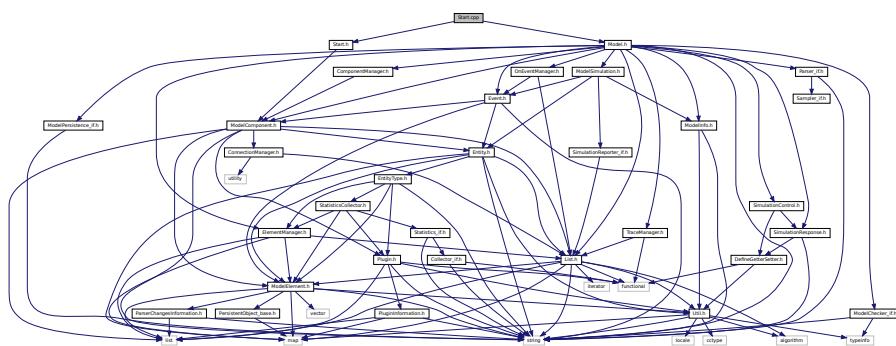
9.441 SourceModelComponent.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:     SourceModelCOMPONENT.h
00009 * Author:   rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:50
00012 */
00013
00014 #ifndef SOURCEMODELCOMPONENT_H
```

```
00015 #define SOURCEMODELCOMPONENT_H
00016
00017 #include "ModelComponent.h"
00018 #include <string>
00019 #include <limits>
00020 //#include <numeric_limits>
00021 //namespace GenesysKernel {
00022
00023     class SourceModelComponent : public ModelComponent {
00024     public:
00025         SourceModelComponent(Model* model, std::string componentTypename, std::string name = "");
00026         virtual ~SourceModelComponent() = default;
00027     public: // get & set
00028         void setFirstCreation(double _firstCreation);
00029         double getFirstCreation() const;
00030         void setEntityType(EntityType* _entityType);
00031         EntityType* getEntityType() const;
00032         void setTimeUnit(Util::TimeUnit _timeUnit);
00033         Util::TimeUnit getTimeUnit() const;
00034         void setTimeBetweenCreationsExpression(std::string _timeBetweenCreations);
00035         std::string getTimeBetweenCreationsExpression() const;
00036         void setMaxCreations(std::string _maxCreationsExpression);
00037         std::string getMaxCreations() const;
00038         unsigned int getEntitiesCreated() const;
00039         void setEntitiesCreated(unsigned int _entitiesCreated);
00040         void setEntitiesPerCreation(unsigned int _entitiesPerCreation);
00041         unsigned int getEntitiesPerCreation() const;
00042     public:
00043         virtual std::string show();
00044     protected:
00045         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00046         virtual void _initBetweenReplications();
00047         virtual std::map<std::string, std::string>* _saveInstance();
00048         virtual bool _check(std::string* errorMessage);
00049     protected: // get & set
00050         EntityType* _entityType;
00051         double _firstCreation = 0.0;
00052         unsigned int _entitiesPerCreation = 1;
00053         std::string _maxCreationsExpression = std::to_string(std::numeric_limits<unsigned
00054             int>::max()); // std::numeric_limits<unsigned int>::max();
00055         std::string _timeBetweenCreationsExpression = "EXPO(1)";
00056         Util::TimeUnit _timeBetweenCreationsTimeUnit = Util::TimeUnit::second;
00057         unsigned int _entitiesCreatedSoFar = 0;
00058     };
00059 //namespace\\}
00060 #endif /* SOURCEMODELCOMPONENT_H */
00061
```

9.442 Start.cpp File Reference

```
#include "Start.h"  
#include "Model.h"  
Include dependency graph for Start.cpp:
```



9.443 Start.cpp

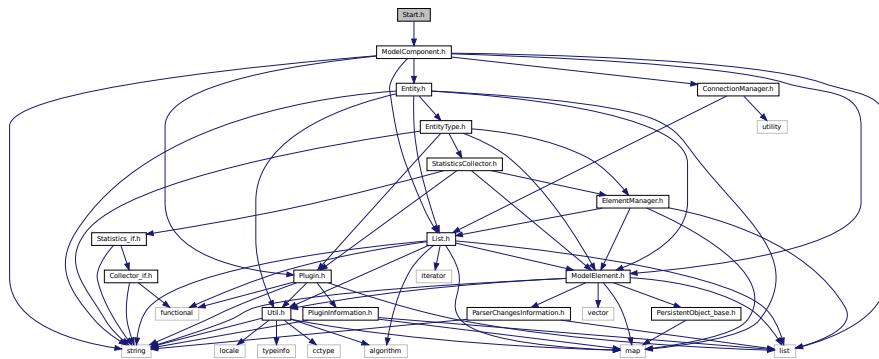
```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.
```

```
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Start.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #include "Start.h"
00015
00016 #include "Model.h"
00017
00018 Start::Start(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Start>(), name) {
00019 }
00020
00021 std::string Start::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Start::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Start* newComponent = new Start(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031     return newComponent;
00032 }
00033
00034
00035 void Start::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Start::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Start::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Start::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Start::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Start::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Start>(), &Start::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069
```

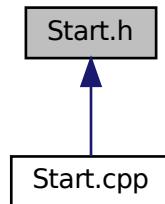
9.444 Start.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for Start.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Start](#)

9.445 Start.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Start.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef START_H
00015 #define START_H
00016
00017 #include "ModelComponent.h"
00018
00039 class Start : public ModelComponent {
00040 public: // constructors
00041     Start(Model* model, std::string name = "");
00042     virtual ~Start() = default;
00043 public: // virtual

```

```

00044     virtual std::string show();
00045 public: // static
00046     static PluginInformation* GetPluginInformation();
00047     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00048 protected: // virtual
00049     virtual void _execute(Entity* entity);
00050     virtual void _initBetweenReplications();
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual std::map<std::string, std::string>* _saveInstance();
00053     virtual bool _check(std::string* errorMessage);
00054 private: // methods
00055 private: // attributes 1:1
00056 private: // attributes 1:n
00057 };
00058
00059
00060 #endif /* START_H */
00061

```

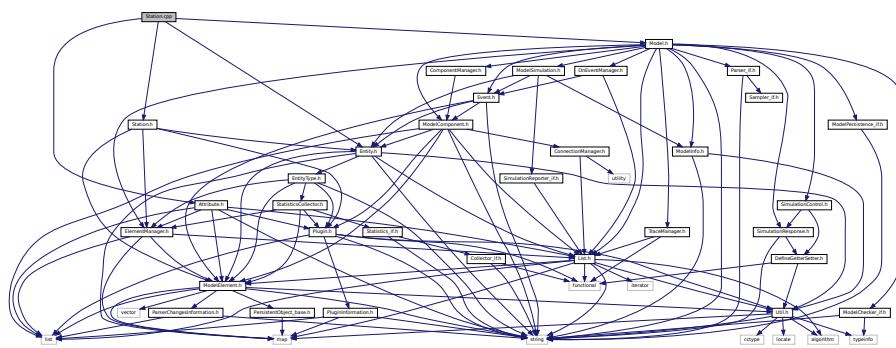
9.446 Station.cpp File Reference

```

#include "Station.h"
#include "Entity.h"
#include "Model.h"
#include "Attribute.h"

```

Include dependency graph for Station.cpp:



9.447 Station.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Station.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #include "Station.h"
00015 #include "Entity.h"
00016 #include "Model.h"
00017 #include "Attribute.h"
00018
00019 Station::Station(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Station>(), name)
00020 {
00021 }
00022 Station::~Station() {
00023     //parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatNumberInStation);
00024     //parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatTimeInStation);
00025 }
00026
00027 std::string Station::show() {

```

```

00028     std::string msg = ModelElement::show() + ",enterIntoStationComponent=";
00029     if (_enterIntoStationComponent == nullptr)
00030         msg += "NULL";
00031     else
00032         msg += _enterIntoStationComponent->getName();
00033     return msg;
00034 }
00035
00036 void Station::initBetweenReplications() {
00037     _cstatNumberInStation->getStatistics()->getCollector()->clear();
00038     _cstatTimeInStation->getStatistics()->getCollector()->clear();
00039 }
00040 }
00041
00042 void Station::enter(Entity* entity) {
00043     std::string attributeName = "Entity.ArrivalAt" + this->getName();
00044     trimwithin(attributeName);
00045     entity->setAttributeValue(attributeName, _parentModel->getSimulation()->getSimulatedTime());
00046     entity->setAttributeValue("Entity.Station", _id);
00047     _numberInStation++;
00048     if (_reportStatistics)
00049         this->_cstatNumberInStation->getStatistics()->getCollector()->addValue(_numberInStation);
00050 }
00051
00052 void Station::leave(Entity* entity) {
00053     std::string attributeName = "Entity.ArrivalAt" + this->getName();
00054     trimwithin(attributeName);
00055     double arrivalTime = entity->getAttributeValue(attributeName);
00056     double timeInStation = _parentModel->getSimulation()->getSimulatedTime() - arrivalTime;
00057     entity->setAttributeValue("Entity.Station", 0.0);
00058     _numberInStation--;
00059     if (_reportStatistics) {
00060         _cstatNumberInStation->getStatistics()->getCollector()->addValue(_numberInStation);
00061         _cstatTimeInStation->getStatistics()->getCollector()->addValue(timeInStation);
00062         if (entity->getEntityType()->isReportStatistics())
00063             entity->getEntityType()->addGetStatisticsCollector(entity->getEntityTypeName() +
00064 ".TimeInStations")->getStatistics()->getCollector()->addValue(timeInStation); // \todo: should check
00065         if entitytype reports (?)
00066     }
00067 }
00068
00069 void Station::setEnterIntoStationComponent(ModelComponent* _enterIntoStationComponent) {
00070     this->_enterIntoStationComponent = _enterIntoStationComponent;
00071 }
00072
00073 ModelComponent* Station::getEnterIntoStationComponent() const {
00074     return _enterIntoStationComponent;
00075 }
00076
00077 PluginInformation* Station::GetPluginInformation() {
00078     PluginInformation* info = new PluginInformation(Util::TypeOf<Station>(), &Station::LoadInstance);
00079 }
00080
00081 ModelElement* Station::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00082     Station* newElement = new Station(model);
00083     try {
00084         newElement->_loadInstance(fields);
00085     } catch (const std::exception& e) {
00086     }
00087     return newElement;
00088 }
00089
00090 bool Station::_loadInstance(std::map<std::string, std::string>* fields) {
00091     bool res = ModelElement::_loadInstance(fields);
00092     if (res) {
00093         try {
00094             } catch (...) {
00095         }
00096     }
00097     return res;
00098 }
00099
00100 std::map<std::string, std::string>* Station::_saveInstance() {
00101     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00102     //Util::TypeOf<Station>());
00103     return fields;
00104 }
00105
00106 bool Station::_check(std::string* errorMessage) {
00107     /* include attributes needed */
00108     std::vector<std::string> neededNames = {"Entity.Station"};
00109     neededNames.insert(neededNames.begin(), "Entity.ArrivalAt" + this->getName());
00110     std::string neededName;
00111     for (unsigned int i = 0; i < neededNames.size(); i++)
00112         neededName = neededNames[i];

```

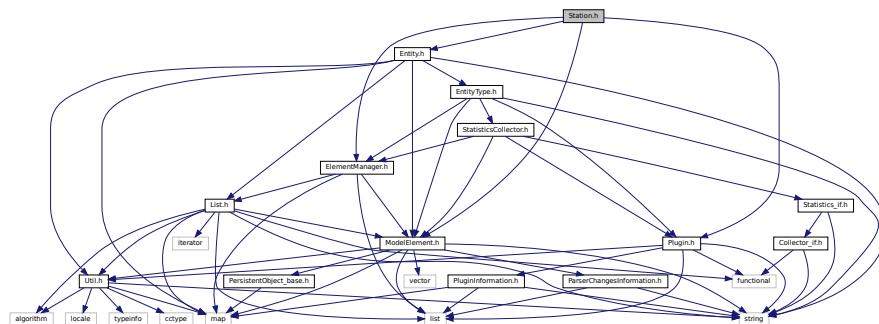
```

00112         if (_parentModel->getElements()->getElement(Util::TypeOf<Attribute>(), neededName) == nullptr)
00113     {
00114         new Attribute(_parentModel, neededName);
00115     }
00116     //
00117     return true;
00118 }
00119
00120 void Station::_createInternalElements() {
00121     if (_reportStatistics) {
00122         if (_cstatNumberInStation == nullptr) {
00123             _cstatNumberInStation = new StatisticsCollector(_parentModel, _name + "." +
00124 "NumberInStation", this);
00125             _cstatTimeInStation = new StatisticsCollector(_parentModel, _name + "." + "TimeInStation",
00126 this);
00127             _childrenElements->insert({"NumberInStation", _cstatNumberInStation});
00128             _childrenElements->insert({"TimeInStation", _cstatTimeInStation});
00129             //
00130             // include StatisticsCollector needed in EntityType
00131             std::list<ModelElement*>* enttypes =
00132             _parentModel->getElements()->getElementList(Util::TypeOf<EntityType>())->list();
00133             for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end();
00134             it++) {
00135                 if ((*it)->isReportStatistics())
00136                     static_cast<EntityType*> ((*it))->addGetStatisticsCollector((*it)->getName() +
00137 ".TimeInStations"); // force create this CStat before simulation starts
00138             }
00139         }
00140     }

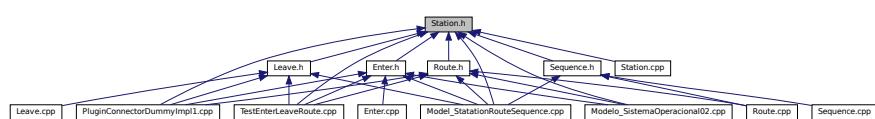
```

9.448 Station.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
#include "Entity.h"
Include dependency graph for Station.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class Station

9.449 Station.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Station.h
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #ifndef STATION_H
00015 #define STATION_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020 #include "Entity.h"
00021
00022 /*
00023 Station module
00024 DESCRIPTION
00025 The Station module defines a station (or a set of stations) corresponding to a physical
00026 or logical location where processing occurs. If the Station module defines a station
00027 set, it is effectively defining multiple processing locations.
00028 The station (or each station within the defined set) has a matching Activity Area that
00029 is used to report all times and costs accrued by the entities in this station. This
00030 Activity Area's name is the same as the station. If a parent Activity Area is defined,
00031 then it also accrues any times and costs by the entities in this station.
00032 TYPICAL USES
00033 Defining a lathe area
00034 Defining a set of toll booths
00035 Defining a food preparation area
00036 PROMPTS
00037 Prompt Description
00038 Name Unique name of the module that will be displayed in the
00039 flowchart.
00040 Station Type Type of station being defined, either as an individual Station or a
00041 station Set.
00042 Station Name Name of the individual station.
00043 Set Name Name of the station set.
00044 Parent Activity Area Name of the Activity Area's parent.
00045 Associated Intersection Name of the intersection associated with this station in a guided
00046 transporter network.
00047 Report Statistics Specifies whether or not statistics will automatically be collected
00048 and stored in the report database for this station and its
00049 corresponding activity area.
00050 Save Attribute Attribute name used to store the index number into the station set
00051 of the member that is selected.
00052 Station Set Members Names of the stations that are members of this station set.
00053 Station Name A given station can only exist once within a model. Therefore, an
00054 individual station can only be the member of one station set, and
00055 that individual station may not be the name of a station in
00056 another module.
00057 Parent Activity Area Name of the Activity Area's parent for the station set member.
00058 Associated Intersection Name of the intersection associated with this station set in a
00059 guided transporter network.
00060 Report Statistics Specifies whether or not statistics will automatically be collected
00061 and stored in the report database for this station set member and
00062 its corresponding activity area.
00063 */
00064 class Station : public ModelElement {
00065 public:
00066     Station(Model* model, std::string name = "");
00067     virtual ~Station();
00068 public:
00069     virtual std::string show();
00070 public: // static
00071     static PluginInformation* GetPluginInformation();
00072     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00073 public:
00074     void initBetweenReplications();
00075     void enter(Entity* entity);
00076     void leave(Entity* entity);
00077     void setEnterIntoStationComponent(ModelComponent* _enterIntoStationComponent);

```

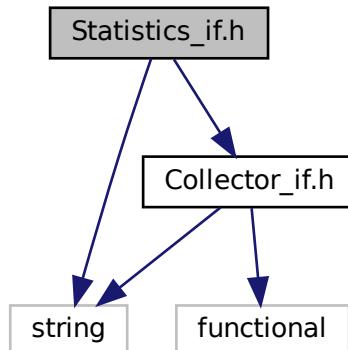
```

00078     ModelComponent* _getEnterIntoStationComponent() const;
00079 protected:
00080     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00081     virtual std::map<std::string, std::string>* _saveInstance();
00082     virtual bool _check(std::string* errorMessage);
00083     virtual void _createInternalElements();
00084 private:
00085     unsigned int _numberInStation = 0;
00086     ModelComponent* _enterIntoStationComponent;
00087 private: // inner elements
00088     StatisticsCollector* _cstatNumberInStation = nullptr;
00089     StatisticsCollector* _cstatTimeInStation;
00090 };
00091
00092 #endif /* STATION_H */
00093

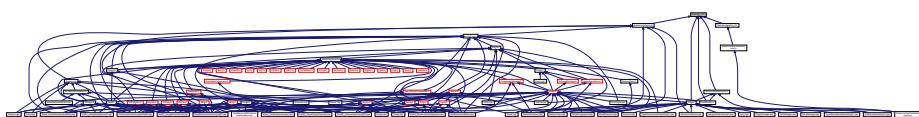
```

9.450 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](#)

9.451 Statistics_if.h

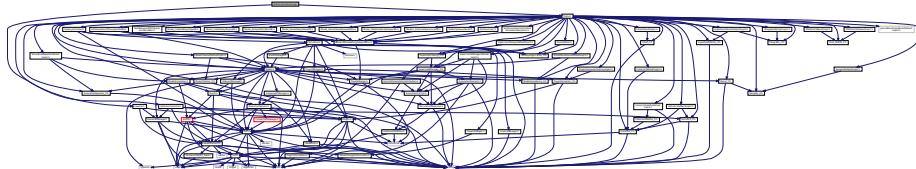
```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: Statistics_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 13:47
00012 */
00013
00014 #ifndef STATISTICS_IF_H
00015 #define STATISTICS_IF_H
00016
00017 #include <string>
00018 #include "Collector_if.h"
00019
00020 class Statistics_if {
00021 public:
00022     virtual Collector_if* getCollector() = 0;
00023     virtual void setCollector(Collector_if* collector) = 0;
00024 public:
00025     virtual unsigned int numElements() = 0;
00026     virtual double min() = 0;
00027     virtual double max() = 0;
00028     virtual double average() = 0;
00029     virtual double variance() = 0;
00030     virtual double stdDeviation() = 0;
00031     virtual double variationCoef() = 0;
00032     virtual double halfWidthConfidenceInterval() = 0;
00033     virtual unsigned int newSampleSize(double halfWidth) = 0;
00034     virtual double getConfidenceLevel() = 0;
00035     virtual void setConfidenceLevel(double confidencelevel) = 0;
00036 };
00037
00038 #endif /* STATISTICS_IF_H */
00039

```

9.452 StatisticsCollector.cpp File Reference

```
#include "StatisticsCollector.h"
#include "Traits.h"
Include dependency graph for StatisticsCollector.cpp:
```



TypeDefs

- `typedef Traits< ModelComponent >::StatisticsCollector_StatisticsImplementation StatisticsClass`

9.452.1 Typedef Documentation

9.452.1.1 StatisticsClass `typedef Traits<ModelComponent>::StatisticsCollector_Statistics< Implementation > StatisticsClass`

Definition at line 20 of file [StatisticsCollector.cpp](#).

9.453 StatisticsCollector.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: StatisticsCollector.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Agosto de 2018, 17:24
00012 */
00013
00014 #include "StatisticsCollector.h"
00015 #include "Traits.h"
00016
00017 //using namespace GenesysKernel;
00018
00019
00020 typedef Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation StatisticsClass;
00021
00022 StatisticsCollector::StatisticsCollector(Model* model, std::string name, ModelElement* parent, bool
00023     insertIntoModel) : ModelElement(model, Util::TypeOf<StatisticsCollector>(), name, insertIntoModel) {
00024     _parent = parent;
00025     _initStaticsAndCollector();
00026     _addSimulationResponses();
00027 }
00028 void StatisticsCollector::_addSimulationResponses() {
00029     GetterMember getterMemberAverage =
00030         DefineGetterMember<StatisticsClass>(static_cast<StatisticsClass*> (this->_statistics),
00031         &StatisticsClass::average);
00032     SimulationResponse* resp = new SimulationResponse(Util::TypeOf<StatisticsClass>(), _name +
00033         ".average", getterMemberAverage);
00034     _parentModel->getResponses()->insert(resp);
00035     // add the halfwidth as response
00036     GetterMember getterMemberHalfWidth =
00037         DefineGetterMember<StatisticsClass>(static_cast<StatisticsClass*> (this->_statistics),
00038         &StatisticsClass::halfWidthConfidenceInterval);
00039     resp = new SimulationResponse(Util::TypeOf<StatisticsClass>(), /*parentName + ":" + */_name +
00040         ".halfWidth", getterMemberHalfWidth);
00041     _parentModel->getResponses()->insert(resp);
00042 }
00043 std::string StatisticsCollector::show() {
00044     std::string parentStr = "";
00045     if (_parent != nullptr) {
00046         try {
00047             parentStr = _parent->getName();
00048         } catch (...) { // if parent changed or deleted, can cause seg fault
00049             parentStr = "<INCONSISTENT>"; /* \todo: +++ */
00050         }
00051     }
00052     return ModelElement::show() +
00053         ",parent=\"" + parentStr + "\"\n" +
00054         ",numElements=" + std::to_string(_statistics->numElements());
00055 }
00056
00057 ModelElement* StatisticsCollector::getParent() const {
00058     return _parent;
00059 }
00060
00061 Statistics_if* StatisticsCollector::getStatistics() const {
00062     return _statistics;
00063 }
00064
00065 PluginInformation* StatisticsCollector::GetPluginInformation() {
00066     PluginInformation* info = new PluginInformation(Util::TypeOf<StatisticsCollector>(),
00067         &StatisticsCollector::LoadInstance);
00068     info->setGenerateReport(true);
00069     return info;
00070 }
00071 ModelElement* StatisticsCollector::LoadInstance(Model* model, std::map<std::string, std::string>*
00072     fields) {
00073     StatisticsCollector* newElement = new StatisticsCollector(model);
00074     try {
00075         newElement->_loadInstance(fields);
00076     }
00077 }

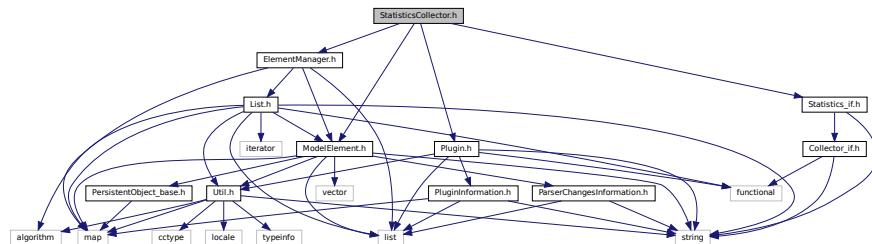
```

```

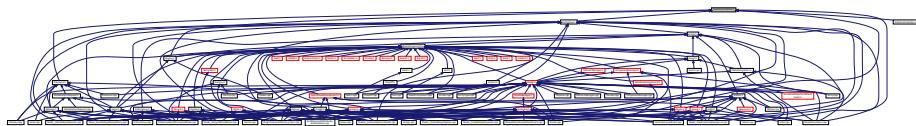
00075     } catch (const std::exception& e) {
00076     }
00077     return newElement;
00078 }
00079
00080 bool StatisticsCollector::_loadInstance(std::map<std::string, std::string>* fields) {
00081     bool res = ModelElement::_loadInstance(fields);
00082     if (res) {
00083     }
00084     return res;
00085 }
00086 }
00087
00088 std::map<std::string, std::string>* StatisticsCollector::_saveInstance() {
00089     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00090     /*Util::TypeOf<StatisticsCollector>()*/;
00091     std::string parentId = "", parentTypename = "";
00092     if (this->parent != nullptr) {
00093         parentId = std::to_string(_parent->getId());
00094         parentTypename = _parent->getClassName();
00095     }
00096     fields->emplace("parentTypename", parentTypename);
00097     fields->emplace("parentId", parentId);
00098     return fields;
00099 }
00100
00101 bool StatisticsCollector::_check(std::string* errorMessage) {
00102     return true;
00103 }
```

9.454 StatisticsCollector.h File Reference

```
#include "ModelElement.h"
#include "Statistics_if.h"
#include "ElementManager.h"
#include "Plugin.h"
Include dependency graph for StatisticsCollector.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [StatisticsCollector](#)

9.455 StatisticsCollector.h

```

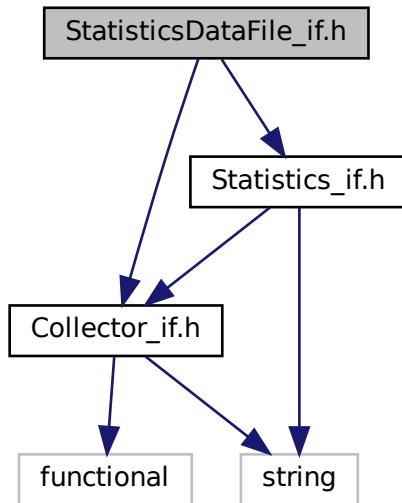
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: StatisticsCollector.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Agosto de 2018, 17:24
00012 */
00013
00014 #ifndef STATISTICSCOLLECTOR_H
00015 #define STATISTICSCOLLECTOR_H
00016
00017 #include "ModelElement.h"
00018 #include "Statistics_if.h"
00019 #include "ElementManager.h"
00020 #include "Plugin.h"
00021
00022 //namespace GenesysKernel {
00023
00024 class StatisticsCollector : public ModelElement { //, public Statistics_if {
00025     public:
00026         StatisticsCollector(Model* model, std::string name = "", ModelElement* parent = nullptr, bool
00027         insertIntoModel = true);
00028         virtual ~StatisticsCollector() = default;
00029     public:
00030         virtual std::string show();
00031     public:
00032         static PluginInformation* GetPluginInformation();
00033         static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00034     public:
00035         ModelElement* getParent() const;
00036         Statistics_if* getStatistics() const;
00037
00038     protected:
00039         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00040         virtual std::map<std::string, std::string>* _saveInstance();
00041         virtual bool _check(std::string* errorMessage);
00042     protected:
00043         void _addSimulationResponses();
00044     private:
00045         void _initStaticsAndCollector();
00046     private:
00047         ModelElement* _parent;
00048         Statistics_if* _statistics; //= new
00049         Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation();
00050     };
00051 //namespace\\}
00052 #endif /* STATISTICSCOLLECTOR_H */
00053

```

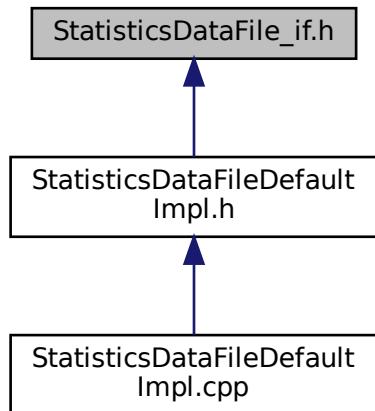
9.456 StatisticsDataFile_if.h File Reference

```
#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](#)

9.457 StatisticsDataFile_if.h

```

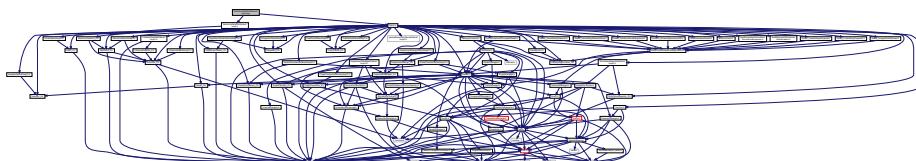
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: StatisticsDataFile.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:16
00012 */
00013
00014 #ifndef STATISTICSFILE_IF_H
00015 #define STATISTICSFILE_IF_H
00016
00017 #include "Collector_if.h"
00018 #include "Statistics_if.h"
00019
00020 class StatisticsDatafile_if : public Statistics_if {
00021 public:
00022     virtual double mode() = 0;
00023     virtual double mediane() = 0;
00024     virtual double quartil(unsigned short num) = 0;
00025     virtual double decil(unsigned short num) = 0;
00026     virtual double centil(unsigned short num) = 0;
00027     virtual void setHistogramNumClasses(unsigned short num) = 0;
00028     virtual unsigned short histogramNumClasses() = 0;
00029     virtual double histogramClassLowerLimit(unsigned short classNum) = 0;
00030     virtual unsigned int histogramClassFrequency(unsigned short classNum) = 0;
00031
00032 };
00033
00034 #endif /* STATISTICSFILE_IF_H */
00035

```

9.458 StatisticsDataFileDefaultImpl.cpp File Reference

```
#include "StatisticsDataFileDefaultImpl.h"
#include "Traits.h"

Include dependency graph for StatisticsDataFileDefaultImpl.cpp:
```



9.459 StatisticsDataFileDefaultImpl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: StatisticsDataFileDummyImpl.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:24
00012 */
00013
00014 #include "StatisticsDataFileDefaultImpl.h"
00015
00016 #include "Traits.h"
00017
00018 StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl() {
00019     _collector = new Traits<Statistics_if>::CollectorImplementation();
00020 }
00021

```

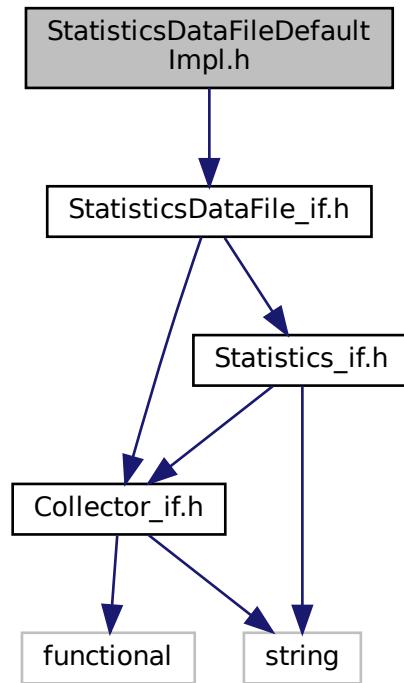
```

00022 unsigned int StatisticsDataFileDummyImpl::numElements() {
00023     return 0; // dummy
00024 }
00025
00026 double StatisticsDataFileDummyImpl::min() {
00027     return 0.0; // dummy
00028 }
00029
00030 double StatisticsDataFileDummyImpl::max() {
00031     return 0.0; // dummy
00032 }
00033
00034 double StatisticsDataFileDummyImpl::average() {
00035 }
00036 }
00037
00038 double StatisticsDataFileDummyImpl::mode() {
00039     return 0.0; // dummy
00040 }
00041
00042 double StatisticsDataFileDummyImpl::mediane() {
00043     return 0.0; // dummy
00044 }
00045
00046 double StatisticsDataFileDummyImpl::variance() {
00047     return 0.0; // dummy
00048 }
00049
00050 double StatisticsDataFileDummyImpl::stddeviation() {
00051     return 0.0; // dummy
00052 }
00053
00054 double StatisticsDataFileDummyImpl::variationCoef() {
00055     return 0.0; // dummy
00056 }
00057
00058 double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval(double confidencelevel) {
00059     return 0.0; // dummy
00060 }
00061
00062 unsigned int StatisticsDataFileDummyImpl::newSampleSize(double confidencelevel, double halfWidth) {
00063     return 0; // dummy
00064 }
00065
00066 double StatisticsDataFileDummyImpl::quartil(unsigned short num) {
00067     return 0.0; // dummy
00068 }
00069
00070 double StatisticsDataFileDummyImpl::decil(unsigned short num) {
00071     return 0.0; // dummy
00072 }
00073
00074 double StatisticsDataFileDummyImpl::centil(unsigned short num) {
00075     return 0.0; // dummy
00076 }
00077
00078 void StatisticsDataFileDummyImpl::setHistogramNumClasses(unsigned short num) {
00079 }
00080
00081 unsigned short StatisticsDataFileDummyImpl::histogramNumClasses() {
00082     return 0; // dummy
00083 }
00084
00085 double StatisticsDataFileDummyImpl::histogramClassLowerLimit(unsigned short classNum) {
00086     return 0.0; // dummy
00087 }
00088
00089 unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency(unsigned short classNum) {
00090     return 0; // dummy
00091 }
00092
00093 Collector_if* StatisticsDataFileDummyImpl::getCollector() {
00094     return this->_collector;
00095 }
00096
00097 void StatisticsDataFileDummyImpl::setCollector(Collector_if* collector) {
00098 }
00099 }
```

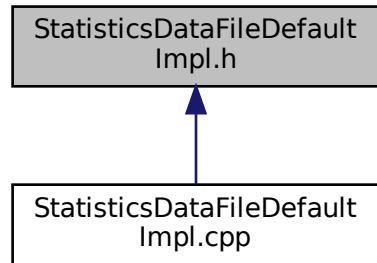
9.460 StatisticsDataFileDefaultImpl.h File Reference

```
#include "StatisticsDataFile_if.h"
```

Include dependency graph for StatisticsDataFileDefaultImpl.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [StatisticsDataFileDummyImpl](#)

9.461 StatisticsDataFileDefaultImpl.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: StatisticsDataFileDefaultImpl.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:24
00012 */
00013
00014 #ifndef STATISTICSFILEDEFAULTRIMPL_H
00015 #define STATISTICSFILEDEFAULTRIMPL_H
00016
00017 #include "StatisticsDataFile_if.h"
00018
00019 class StatisticsDataFileDummyImpl : public StatisticsDatafile_if {
00020 public:
00021     StatisticsDataFileDummyImpl();
00022     virtual ~StatisticsDataFileDummyImpl() = default;
00023 public:
00024     virtual Collector_if* getCollector();
00025     void setCollector(Collector_if* collector);
00026 public:
00027     virtual unsigned int numElements();
00028     virtual double min();
00029     virtual double max();
00030     virtual double average();
00031     virtual double variance();
00032     virtual double stddeviation();
00033     virtual double variationCoef();
00034     virtual double halfWidthConfidenceInterval(double confidencelevel);
00035     virtual unsigned int newSampleSize(double confidencelevel, double halfWidth);
00036
00037     virtual double mode();
00038     virtual double mediane();
00039     virtual double quartil(unsigned short num);
00040     virtual double decil(unsigned short num);
00041     virtual double centil(unsigned short num);
00042     virtual void setHistogramNumClasses(unsigned short num);
00043     virtual unsigned short histogramNumClasses();
00044     virtual double histogramClassLowerLimit(unsigned short className);
00045     virtual unsigned int histogramClassFrequency(unsigned short className);
00046 private:
00047     Collector_if* _collector;
00048 };
00049
00050 #endif /* STATISTICSFILEDEFAULTRIMPL_H */
00051

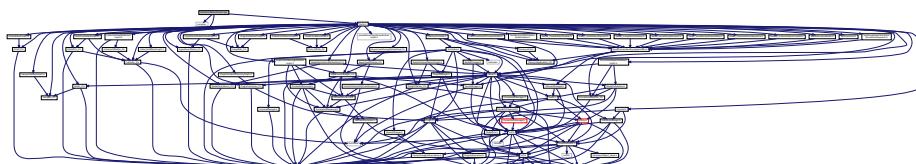
```

9.462 StatisticsDefaultImpl1.cpp File Reference

```

#include <complex>
#include "StatisticsDefaultImpl1.h"
#include "Traits.h"
Include dependency graph for StatisticsDefaultImpl1.cpp:

```



9.463 StatisticsDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */

```

```

00005  */
00006
00007 /*
00008 * File: StatisticsDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 1 de Agosto de 2018, 21:03
00012 */
00013
00014 #include <complex>
00015
00016 #include "StatisticsDefaultImpl1.h"
00017 #include "Traits.h"
00018 //##include "Integrator_if.h"
00019 //##include "ProbDistribDefaultImpl1.h"
00020
00021 //using namespace GenesysKernel;
00022
00023 StatisticsDefaultImpl1::StatisticsDefaultImpl1() {
00024     //_collector = new Traits<Statistics_if>::CollectorImplementation();
00025     _collector = new Traits<ModelComponent>::StatisticsCollector_CollectorImplementation();
00026
00027     _collector->setAddValueHandler(setCollectorAddValueHandler(&StatisticsDefaultImpl1::collectorAddHandler,
00028                                     this));
00029     _collector->setClearHandler(setCollectorClearHandler(&StatisticsDefaultImpl1::collectorClearHandler,
00030                                     this));
00031     //_collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00032                                     std::placeholders::_1));
00033     this->initStatistics();
00034 }
00035
00036 StatisticsDefaultImpl1::StatisticsDefaultImpl1(Collector_if* collector) {
00037     _collector = collector;
00038
00039     _collector->setAddValueHandler(setCollectorAddValueHandler(&StatisticsDefaultImpl1::collectorAddHandler,
00040                                     this));
00041     _collector->setClearHandler(setCollectorClearHandler(&StatisticsDefaultImpl1::collectorClearHandler,
00042                                     this));
00043     //_collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00044                                     std::placeholders::_1));
00045     this->initStatistics();
00046 }
00047
00048 void StatisticsDefaultImpl1::collectorAddHandler(double newValue) {
00049     _elems = _collector->numElements();
00050     if (newValue < _min) {
00051         _min = newValue;
00052     }
00053     if (newValue > _max) {
00054         _max = newValue;
00055     }
00056     // alternative 1
00057     _sum += newValue;
00058     _sumSquare += newValue*newValue;
00059     _average = _sum / _elems;
00060     _variance = _sumSquare / _elems - _average*_average;
00061     // alternative 2
00062     //_average = (_average * (elems - 1) + newValue) / elems; // this approach propagates the numeric
00063     // error
00064     //_variance = (_variance * (elems - 1) + pow(newValue - _average, 2)) / elems; // this approach
00065     // propagates the numeric error
00066     _stddeviation = sqrt(_variance);
00067     _variationCoef = (_average != 0 ? _stddeviation / _average : 0.0);
00068     _halfWidth = _criticalTn_1 * (_stddeviation / std::sqrt(_elems));
00069 }
00070
00071 void StatisticsDefaultImpl1::collectorClearHandler() {
00072     this->initStatistics();
00073 }
00074
00075 void StatisticsDefaultImpl1::initStatistics() {
00076     _elems = 0;
00077     _min = +1e+99;
00078     _max = -1e+99;
00079     _sum = 0.0;
00080     _sumSquare = 0.0;
00081     _average = 0.0;
00082     _variance = 0.0;
00083     _stddeviation = 0.0;
00084     _variationCoef = 0.0;
00085     _halfWidth = 0.0;
00086 }
00087
00088 unsigned int StatisticsDefaultImpl1::numElements() {
00089     return this->getCollector()->numElements();

```

```

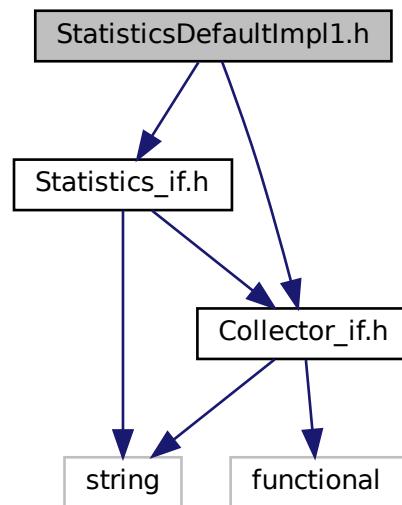
00080 }
00081
00082 double StatisticsDefaultImpl1::min() {
00083     if (_elems > 0)
00084         return _min;
00085     else
00086         return 0.0;
00087 }
00088
00089 double StatisticsDefaultImpl1::max() {
00090     if (_elems > 0)
00091         return _max;
00092     else
00093         return 0.0;
00094 }
00095
00096 double StatisticsDefaultImpl1::average() {
00097     return _average;
00098 }
00099
00100 double StatisticsDefaultImpl1::variance() {
00101     return _variance;
00102 }
00103
00104 double StatisticsDefaultImpl1::stddeviation() {
00105     return _stddeviation;
00106 }
00107
00108 double StatisticsDefaultImpl1::variationCoef() {
00109     return _variationCoef;
00110 }
00111
00112 double StatisticsDefaultImpl1::halfWidthConfidenceInterval() {
00113     return _halfWidth;
00114 }
00115
00116 void StatisticsDefaultImpl1::setConfidenceLevel(double confidencelevel) {
00117     _confidenceLevel = confidencelevel;
00118     //Integrator_if* integrator = new Traits<Integrator_if>::Implementation();
00119     _criticalTn_1 = 1.96; //integrator->integrate()
00120
00121 }
00122
00123 double StatisticsDefaultImpl1::getConfidenceLevel() {
00124     return _confidenceLevel;
00125 }
00126
00127 unsigned int StatisticsDefaultImpl1::newSampleSize(double halfWidth) {
00128     return 0;
00129 }
00130
00131 Collector_if* StatisticsDefaultImpl1::getCollector() {
00132     return this->_collector;
00133 }
00134
00135 void StatisticsDefaultImpl1::setCollector(Collector_if* collector) {
00136     this->_collector = collector;
00137 }

```

9.464 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`

9.465 StatisticsDefaultImpl1.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: StatisticsDefaultImpl1.h  
00009 * Author: rafael.luiz.cancian  
00010 *  
00011 * Created on 1 de Agosto de 2018, 21:03  
00012 */  
00013  
00014 #ifndef STATISTICSDEFAULTIMPL1_H  
00015 #define STATISTICSDEFAULTIMPL1_H  
00016  
00017 #include "Statistics_if.h"  
00018 #include "Collector_if.h"  
00019 //namespace GenesysKernel {  
00020  
00021     class StatisticsDefaultImpl1 : public Statistics_if {  
00022     public:  
00023         StatisticsDefaultImpl1();  
00024         StatisticsDefaultImpl1(Collector_if* collector);
```

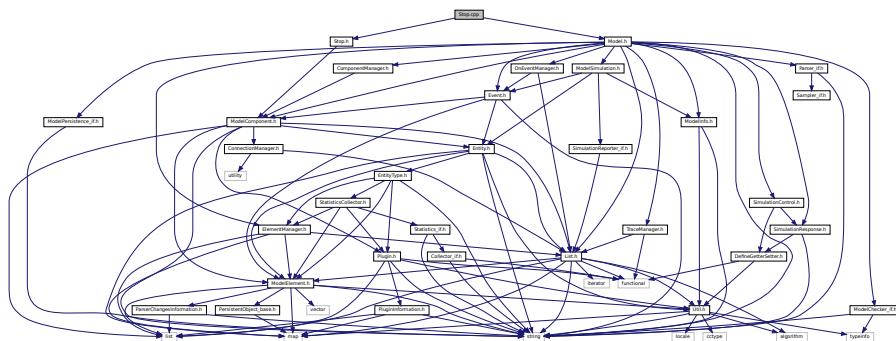
```

00025     virtual ~StatisticsDefaultImpl1() = default;
00026 public:
00027     virtual Collector_if* getCollector();
00028     virtual void setCollector(Collector_if* collector);
00029 public:
00030     virtual unsigned int numElements();
00031     virtual double min();
00032     virtual double max();
00033     virtual double average();
00034     virtual double variance();
00035     virtual double stdDeviation();
00036     virtual double variationCoef();
00037     virtual double halfWidthConfidenceInterval();
00038     virtual unsigned int newSampleSize(double halfWidth);
00039     virtual double getConfidenceLevel();
00040     virtual void setConfidenceLevel(double confidencelevel);
00041 private:
00042     void collectorAddHandler(double newValue);
00043     void collectorClearHandler();
00044     void initStatistics();
00045 private:
00046     Collector_if* _collector;
00047     unsigned long _elems;
00048     double _sum;
00049     double _sumSquare;
00050     double _min;
00051     double _max;
00052     double _average;
00053     double _variance;
00054     double _stdDeviation;
00055     double _variationCoef;
00056     double _confidenceLevel = 0.95;
00057     double _criticalTn_1 = 1.96;
00058     double _halfWidth;
00059 };
00060 //namespace\\}
00061 #endif /* STATISTICSDEFAULTIMPL1_H */
00062

```

9.466 Stop.cpp File Reference

```
#include "Stop.h"
#include "Model.h"
Include dependency graph for Stop.cpp:
```



9.467 Stop.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Stop.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15

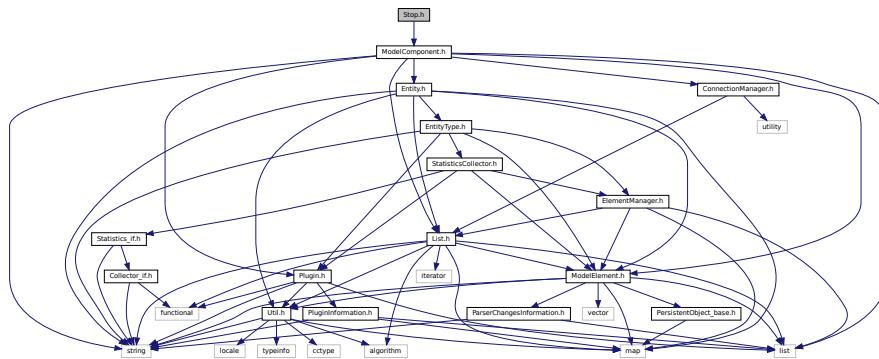
```

```
00012  */
00013
00014 #include "Stop.h"
00015
00016 #include "Model.h"
00017
00018 Stop::Stop(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Stop>(), name) {
00019 }
00020
00021 std::string Stop::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Stop::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026     Stop* newComponent = new Stop(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030
00031     }
00032     return newComponent;
00033 }
00034
00035 void Stop::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Stop::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Stop::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Stop::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Stop::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Stop::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Stop>(), &Stop::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069
```

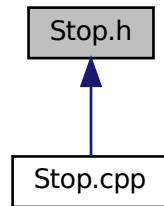
9.468 Stop.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for Stop.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `Stop`

9.469 Stop.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Stop.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef STOP_H
00015 #define STOP_H
00016
00017 #include "ModelComponent.h"
00018
00036 class Stop : public ModelComponent {
00037 public: // constructors
00038     Stop(Model* model, std::string name = "");
00039     virtual ~Stop() = default;
00040 public: // virtual

```

```

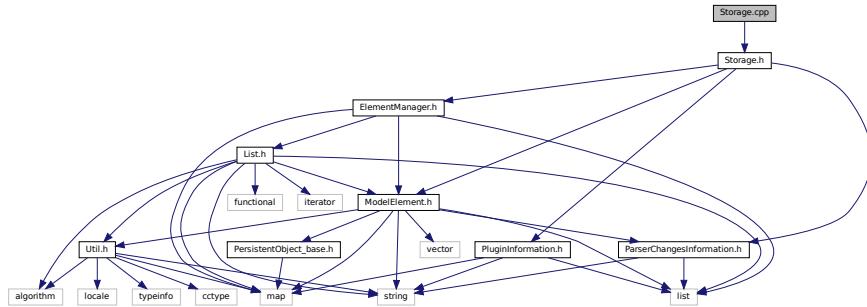
00041     virtual std::string show();
00042 public: // static
00043     static PluginInformation* GetPluginInformation();
00044     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00045 protected: // virtual
00046     virtual void _execute(Entity* entity);
00047     virtual void _initBetweenReplications();
00048     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00049     virtual std::map<std::string, std::string>* _saveInstance();
00050     virtual bool _check(std::string* errorMessage);
00051 private: // methods
00052 private: // attributes 1:1
00053 private: // attributes 1:n
00054 };
00055
00056
00057 #endif /* STOP_H */
00058

```

9.470 Storage.cpp File Reference

#include "Storage.h"

Include dependency graph for Storage.cpp:



9.471 Storage.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Storage.cpp
00009  * Author: rlcancian
00010  *
00011  * Created on 20 de Storageembro de 2019, 20:06
00012 */
00013
00014 #include "Storage.h"
00015
00016 Storage::Storage(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Storage>(), name)
00017 {
00018 }
00019 std::string Storage::show() {
00020     return ModelElement::show() +
00021         "";
00022 }
00023
00024 PluginInformation* Storage::GetPluginInformation() {
00025     PluginInformation* info = new PluginInformation(Util::TypeOf<Storage>(), &Storage::LoadInstance);
00026     return info;
00027 }
00028
00029 ModelElement* Storage::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00030     Storage* newElement = new Storage(model);
00031     try {

```

```

00032     newElement->_loadInstance(fields);
00033 } catch (const std::exception& e) {
00034 }
00035 }
00036 return newElement;
00037 }
00038
00039 bool Storage::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelElement::_loadInstance(fields);
00041     if (res) {
00042         try {
00043             //this->_attributeName = (*fields->find("attributeName")).second;
00044             //this->_orderRule = static_cast<OrderRule>
00045             (stoi((*fields->find("orderRule")).second));
00046         } catch (...) {
00047         }
00048     return res;
00049 }
00050
00051 std::map<std::string, std::string>* Storage::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00053     //Util::TypeOf<Storage>());
00054     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055     //fields->emplace("attributeName", "\"" + this->_attributeName + "\"");
00056 }
00057
00058 bool Storage::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     // resultAll |= ...
00061     return resultAll;
00062 }
00063
00064 ParserChangesInformation* Storage::_getParserChangesInformation() {
00065     ParserChangesInformation* changes = new ParserChangesInformation();
00066     //changes->getProductionToAdd()->insert(...);
00067     //changes->getTokensToAdd()->insert(...);
00068     return changes;
00069 }
00070

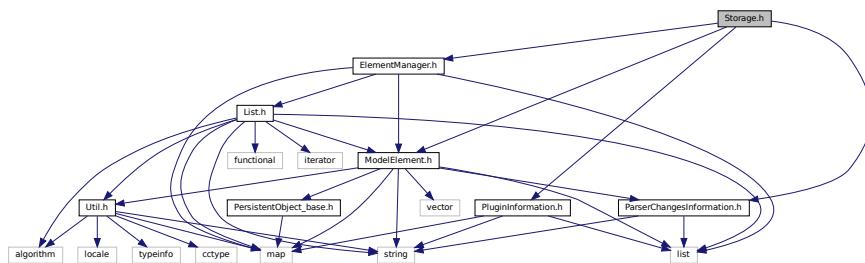
```

9.472 Storage.h File Reference

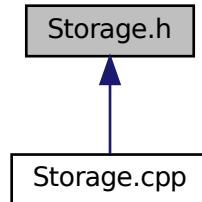
```

#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
Include dependency graph for Storage.h:

```



This graph shows which files directly or indirectly include this file:



Classes

- class [Storage](#)

9.473 Storage.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Storage.h
00009  * Author: rlcancian
00010 *
00011 * Created on 20 de Storageembro de 2019, 20:06
00012 */
00013
00014 #ifndef STORAGE_H
00015 #define STORAGE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00023 /*
00024 Storage module
00025 DESCRIPTION
00026 The Storage module defines the name of a storage. Storages are automatically created
00027 by any module that references the storage so that this module is seldom needed. The
00028 only time this module is needed is when a storage is defined as a member of a storage
00029 set or specified using an attribute or expression.
00030 TYPICAL USES
00031 Defining an animate storage for a set of storages
00032 PROMPTS
00033 Prompt Description
00034 Name The name of the storage set being defined. This name must be
00035 unique.
00036 */
00037 class Storage : public ModelElement {
00038 public:
00039     Storage(Model* model, std::string name = "");
00040     virtual ~Storage() = default;
00041 public: // static
00042     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00043     static PluginInformation* GetPluginInformation();
00044 public:
00045     virtual std::string show();
00046
00047 protected: // must be overriden by derived classes
00048     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00049     virtual std::map<std::string, std::string>* _saveInstance();
00050 protected: // could be overriden by derived classes

```

```

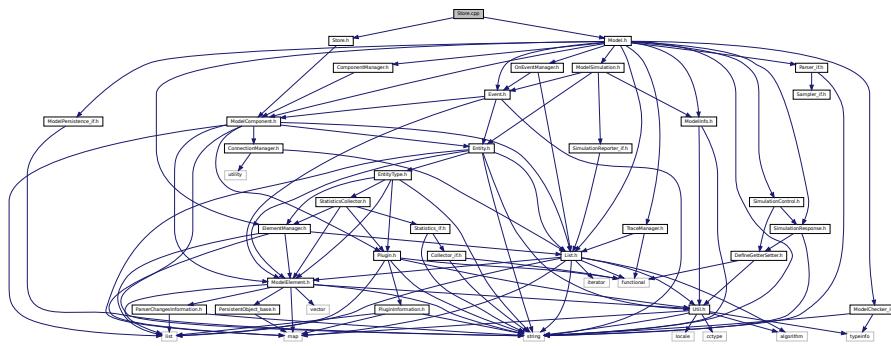
00051     virtual bool _check(std::string* errorMessage);
00052     virtual ParserChangesInformation* _getParserChangesInformation();
00053 };
00054 #endif /* STORAGE_H */
00055

```

9.474 Store.cpp File Reference

```
#include "Store.h"
#include "Model.h"

Include dependency graph for Store.cpp:
```



9.475 Store.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Store.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:07
00012 */
00013
00014 #include "Store.h"
00015 #include "Model.h"
00016
00017 Store::Store(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Store>(), name) {
00018 }
00019
00020 std::string Store::show() {
00021     return ModelComponent::show() + "";
00022 }
00023
00024 ModelComponent* Store::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025     Store* newComponent = new Store(model);
00026     try {
00027         newComponent->_loadInstance(fields);
00028     } catch (const std::exception& e) {
00029     }
00030 }
00031     return newComponent;
00032 }
00033
00034 void Store::_execute(Entity* entity) {
00035     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036     this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
00038
00039 bool Store::_loadInstance(std::map<std::string, std::string>* fields) {
00040     bool res = ModelComponent::_loadInstance(fields);
00041     if (res) {
00042         //...
00043     }
00044     return res;

```

```

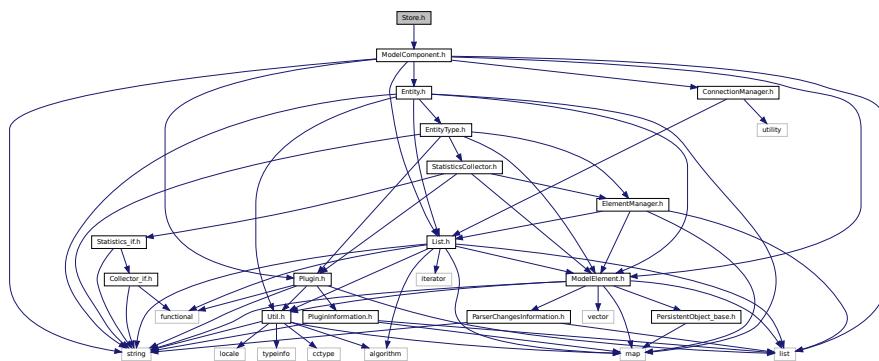
00045 }
00046
00047 void Store::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Store::_saveInstance() {
00051     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052     //...
00053     return fields;
00054 }
00055
00056 bool Store::_check(std::string* errorMessage) {
00057     bool resultAll = true;
00058     //...
00059     return resultAll;
00060 }
00061
00062 PluginInformation* Store::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Store>(), &Store::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067
00068

```

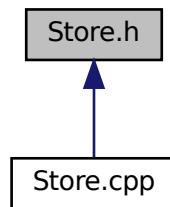
9.476 Store.h File Reference

#include "ModelComponent.h"

Include dependency graph for Store.h:



This graph shows which files directly or indirectly include this file:



Classes

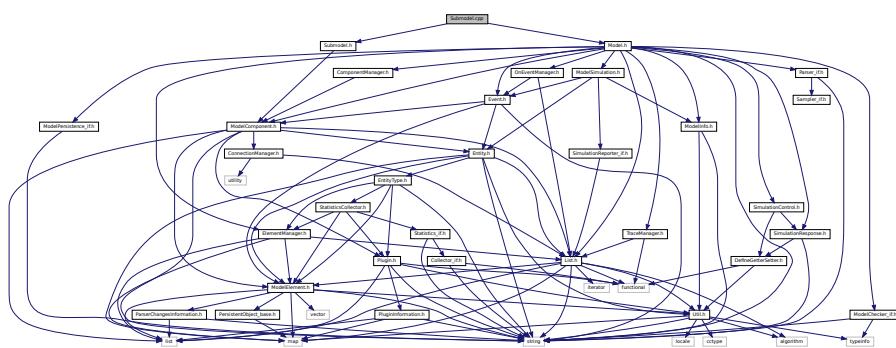
- class [Store](#)

9.477 Store.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: Store.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 11 de Setembro de 2019, 13:07  
00012 */  
00013  
00014 #ifndef STORE_H  
00015 #define STORE_H  
00016  
00017 #include "ModelComponent.h"  
00018  
00050 class Store : public ModelComponent {  
00051 public: // constructors  
00052     Store(Model* model, std::string name = "");  
00053     virtual ~Store() = default;  
00054 public: // virtual  
00055     virtual std::string show();  
00056 public: // static  
00057     static PluginInformation* GetPluginInformation();  
00058     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);  
00059 protected: // virtual  
00060     virtual void _execute(Entity* entity);  
00061     virtual void _initBetweenReplications();  
00062     virtual bool _loadInstance(std::map<std::string, std::string>* fields);  
00063     virtual std::map<std::string, std::string>* _saveInstance();  
00064     virtual bool _check(std::string* errorMessage);  
00065 private: // methods  
00066 private: // attributes 1:1  
00067 private: // attributes 1:n  
00068 };  
00069  
00070  
00071 #endif /* STORE_H */  
00072
```

9.478 Submodel.cpp File Reference

```
#include "Submodel.h"  
#include "Model.h"  
Include dependency graph for Submodel.cpp:
```



9.479 Submodel.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006
```

```

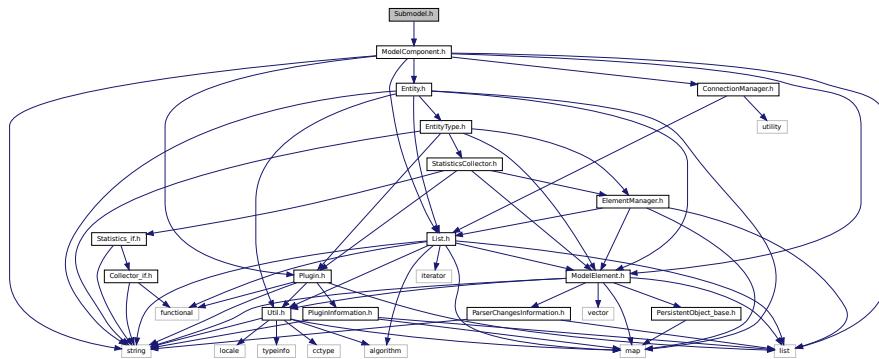
00007 /*
00008  * File: Submodel.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:04
00012 */
00013
00014 #include "Submodel.h"
00015
00016 #include "Model.h"
00017
00018 Submodel::Submodel(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Submodel>(),
00019     name) {
00020
00021     std::string Submodel::show() {
00022         return ModelComponent::show() + "";
00023     }
00024
00025     ModelComponent* Submodel::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00026         Submodel* newComponent = new Submodel(model);
00027         try {
00028             newComponent->_loadInstance(fields);
00029         } catch (const std::exception& e) {
00030
00031     }
00032         return newComponent;
00033     }
00034
00035     void Submodel::_execute(Entity* entity) {
00036         _parentModel->getTracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00037         this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00038             0.0);
00039
00040     bool Submodel::_loadInstance(std::map<std::string, std::string>* fields) {
00041         bool res = ModelComponent::_loadInstance(fields);
00042         if (res) {
00043             //...
00044         }
00045         return res;
00046     }
00047
00048     void Submodel::_initBetweenReplications() {
00049
00050
00051     std::map<std::string, std::string>* Submodel::_saveInstance() {
00052         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00053         //...
00054         return fields;
00055     }
00056
00057     bool Submodel::_check(std::string* errorMessage) {
00058         bool resultAll = true;
00059         //...
00060         return resultAll;
00061     }
00062
00063     PluginInformation* Submodel::GetPluginInformation() {
00064         PluginInformation* info = new PluginInformation(Util::TypeOf<Submodel>(),
00065             &Submodel::LoadInstance);
00066         // ...
00067         return info;
00068     }
00069

```

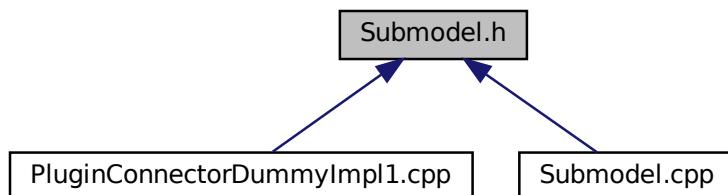
9.480 Submodel.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for Submodel.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Submodel](#)

9.481 Submodel.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Submodel.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:04
00012 */
00013
00014 #ifndef SUBMODEL_H
00015 #define SUBMODEL_H
00016
00017 #include "ModelComponent.h"
00018
00022 class Submodel : public ModelComponent {
00023 public: // constructors
00024     Submodel(Model* model, std::string name = "");
00025     virtual ~Submodel() = default;
00026 public: // virtual
00027     virtual std::string show();

```

```

00028 public: // static
00029     static PluginInformation* GetPluginInformation();
00030     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00031 protected: // virtual
00032     virtual void _execute(Entity* entity);
00033     virtual void _initBetweenReplications();
00034     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00035     virtual std::map<std::string, std::string>* _saveInstance();
00036     virtual bool _check(std::string* errorMessage);
00037 private: // methods
00038 private: // attributes 1:1
00039 private: // attributes 1:n
00040 };
00041
00042
00043 #endif /* SUBMODEL_H */
00044

```

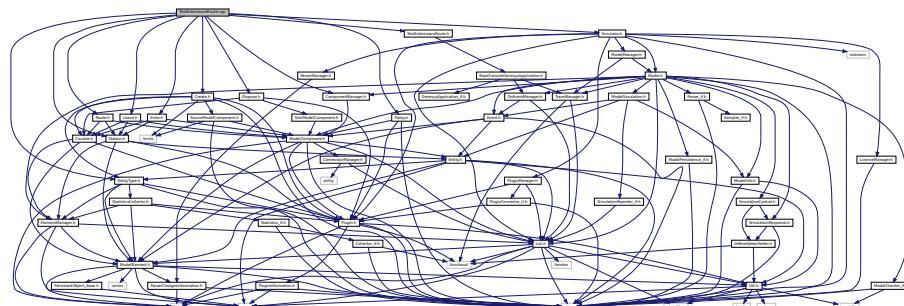
9.482 TestEnterLeaveRoute.cpp File Reference

```

#include "TestEnterLeaveRoute.h"
#include "ComponentManager.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Enter.h"
#include "Route.h"
#include "Leave.h"
#include "EntityType.h"
#include "Station.h"

```

Include dependency graph for TestEnterLeaveRoute.cpp:



9.483 TestEnterLeaveRoute.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestEnterLeaveRoute.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 18:04
00012 */
00013
00014 #include "TestEnterLeaveRoute.h"
00015 #include "ComponentManager.h"
00016
00017
00018 // you have to included need libs
00019
00020 // GEnSyS Simulator

```

```

00021 #include "Simulator.h"
00022
00023 // Model Components
00024 #include "Create.h"
00025 #include "Delay.h"
00026 #include "Dispose.h"
00027 #include "Enter.h"
00028 #include "Route.h"
00029 #include "Leave.h"
00030
00031 // Model elements
00032 #include "EntityType.h"
00033 #include "Station.h"
00034
00035 TestEnterLeaveRoute::TestEnterLeaveRoute() {
00036 }
00037
00038 int TestEnterLeaveRoute::main(int argc, char** argv) {
00039     Simulator* genesys = new Simulator();
00040     // creates an empty model
00041     Model* model = new Model(genesys);
00042     // Handle traces and simulation events to output them
00043     TraceManager* tm = model->getTracer();
00044     this->setDefaultTraceHandlers(tm);
00045     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of
00046     tracing
00047     tm->setTraceLevel(Util::TraceLevel::componentArrival);
00048     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00049     this->insertFakePluginsByHand(genesys);
00050     // get easy access to classes used to insert components and elements into a model
00051     ComponentManager* components = model->getComponents();
00052     ElementManager* elements = model->getElements();
00053     //
00054     // build the simulation model
00055     //
00056     // set general info about the model
00057     ModelSimulation* sim = model->getSimulation();
00058     sim->setReplicationLength(30);
00059     sim->setNumberOfReplications(3);
00060     // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00061     EntityType* entityType1 = new EntityType(model, "AnyEntityType");
00062     elements->insert(entityType1);
00063     // create a ModelComponent of type Create, used to insert entities into the model
00064     Create* create1 = new Create(model);
00065     create1->setEntityType(entityType1);
00066     create1->setTimeBetweenCreationsExpression("5.0");
00067     create1->setEntitiesPerCreation(1);
00068     components->insert(create1);
00069     // create stations to enter and route to
00070     Station* station1 = new Station(model, "Station 1");
00071     Station* station2 = new Station(model, "Station 2");
00072     Station* station3 = new Station(model, "Station 3");
00073     elements->insert(station1);
00074     elements->insert(station2);
00075     elements->insert(station3);
00076     // create components to Enter into Stations
00077     Enter* enter1 = new Enter(model);
00078     enter1->setStation(station1);
00079     components->insert(enter1);
00080     Enter* enter2 = new Enter(model);
00081     enter2->setStation(station2);
00082     components->insert(enter2);
00083     Enter* enter3 = new Enter(model);
00084     enter3->setStation(station3);
00085     components->insert(enter3);
00086     // create components to Leave stations
00087     Leave* leave1 = new Leave(model);
00088     leave1->setStation(station1);
00089     components->insert(leave1);
00090     Leave* leave2 = new Leave(model);
00091     leave2->setStation(station2);
00092     components->insert(leave2);
00093     Leave* leave3 = new Leave(model);
00094     leave3->setStation(station3);
00095     components->insert(leave3);
00096     // create route components
00097     Route* route0 = new Route(model);
00098     route0->setStation(station1);
00099     route0->setRouteTimeExpression("0.5");
00100    components->insert(route0);
00101    Route* routel = new Route(model);
00102    routel->setStation(station2);
00103    routel->setRouteTimeExpression("0.5");
00104    components->insert(routel);
00105    Route* route2 = new Route(model);
00106    route2->setStation(station3);
00107    route2->setRouteTimeExpression("0.5");

```

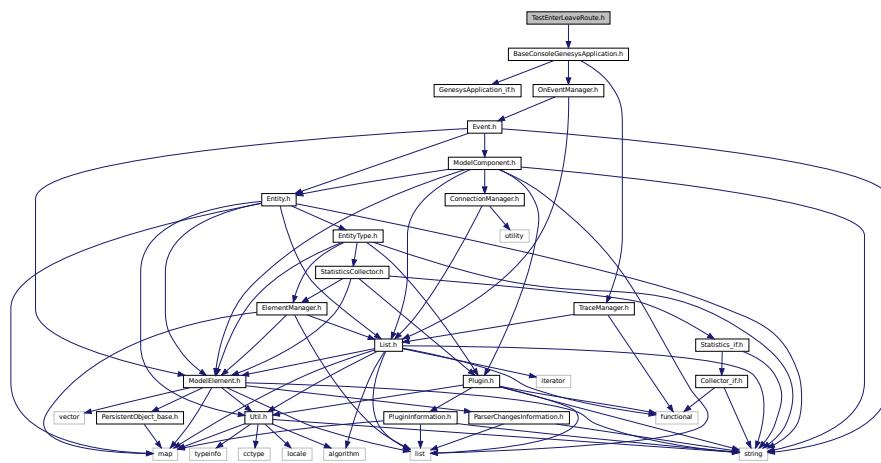
```

00107     components->insert(route2);
00108     // create delay components
00109     Delay* delay1 = new Delay(model);
00110     delay1->setDelayExpression("1.0");
00111     components->insert(delay1);
00112     Delay* delay2 = new Delay(model);
00113     delay2->setDelayExpression("1.0");
00114     components->insert(delay2);
00115     Delay* delay3 = new Delay(model);
00116     delay3->setDelayExpression("1.0");
00117     components->insert(delay3);
00118     // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00119     Dispose* dispose1 = new Dispose(model);
00120     components->insert(dispose1);
00121     // connect model components to create a "workflow"
00122     create1->getNextComponents()->insert(route0);
00123     //
00124     enter1->getNextComponents()->insert(delay1);
00125     delay1->getNextComponents()->insert(leave1);
00126     leave1->getNextComponents()->insert(route1);
00127     //
00128     enter2->getNextComponents()->insert(delay2);
00129     delay2->getNextComponents()->insert(leave2);
00130     leave2->getNextComponents()->insert(route2);
00131     //
00132     enter3->getNextComponents()->insert(delay3);
00133     delay3->getNextComponents()->insert(leave3);
00134     leave3->getNextComponents()->insert(dispose1);
00135     // insert the model into the simulator
00136     genesys->getModels()->insert(model);
00137     // check the model
00138     model->check();
00139     // save the model into a text file
00140     model->save("./temp/testEnterLeaveRoute.txt");
00141     // show the model
00142     model->show();
00143     // execute the simulation
00144     sim->start();
00145     return 0;
00146 }
00147

```

9.484 TestEnterLeaveRoute.h File Reference

#include "BaseConsoleGenesysApplication.h"
Include dependency graph for TestEnterLeaveRoute.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **TestEnterLeaveRoute**

9.485 TestEnterLeaveRoute.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestEnterLeaveRoute.h
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 18:04
00012 */
00013
00014 #ifndef TESTENTERLEAVEROUTE_H
00015 #define TESTENTERLEAVEROUTE_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestEnterLeaveRoute : public BaseConsoleGenesysApplication {
00020 public:
00021     TestEnterLeaveRoute();
00022 public:
00023     int main(int argc, char** argv);
00024
00025 };
00026
00027 #endif /* TESTENTERLEAVEROUTE_H */
00028

```

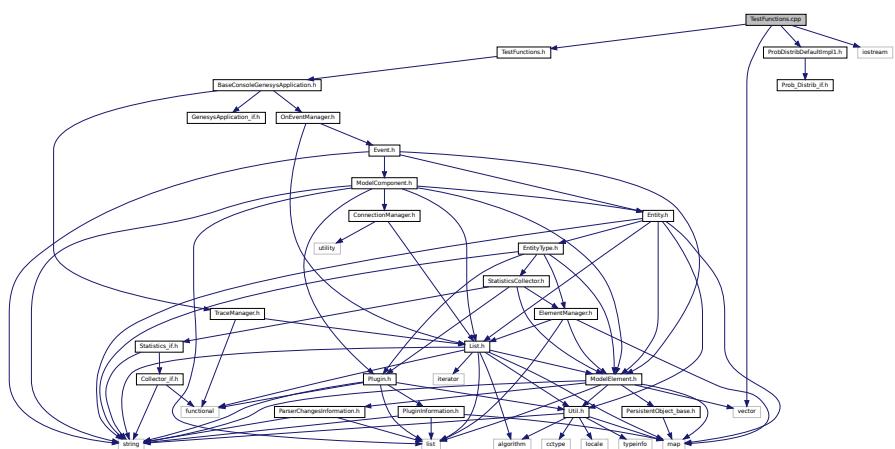
9.486 TestFunctions.cpp File Reference

```

#include "TestFunctions.h"
#include "ProbDistribDefaultImpl1.h"
#include <iostream>
#include <vector>

Include dependency graph for TestFunctions.cpp:

```



Typedefs

- `typedef std::vector< double > state_type`

Functions

- void `harmonic_oscillator` (const `state_type` &x, `state_type` &dxdt, const double t)

Variables

- const double `gam` = 0.15

9.486.1 Typedef Documentation

9.486.1.1 `state_type` `typedef std::vector< double > state_type`

Definition at line 23 of file [TestFunctions.cpp](#).

9.486.2 Function Documentation

9.486.2.1 `harmonic_oscillator()` `void harmonic_oscillator (` `const state_type & x,` `state_type & dxdt,` `const double t)`

Definition at line 26 of file [TestFunctions.cpp](#).

```
00026
00027     dxdt[0] = x[1];
00028     dxdt[1] = 0.0; //x[0] + exp(t);
00029 }
```

9.486.3 Variable Documentation

9.486.3.1 `gam` `const double gam = 0.15`

Definition at line 24 of file [TestFunctions.cpp](#).

9.487 TestFunctions.cpp

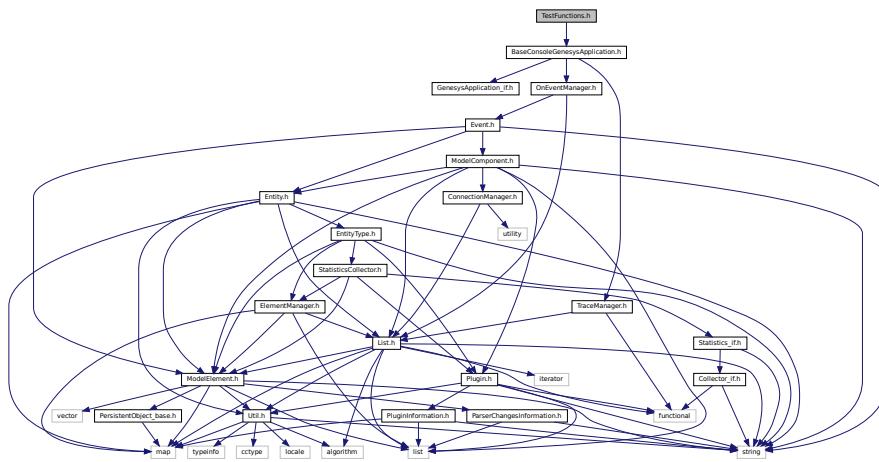
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 26 de Setembro de 2019, 13:00
00012 */
00013
00014 #include "TestFunctions.h"
00015 #include "ProbDistribDefaultImpl.h"
00016
00017 #include <iostream>
00018 #include <vector>
00019
00020 //##include <boost/numeric/odeint.hpp>
00021 //##include <boost/math/distributions.hpp>
00022
00023 typedef std::vector< double > state_type;
00024 const double gam = 0.15;
00025
00026 void harmonic_oscillator(const state_type &x, state_type &dxdt, const double t) {
00027     dxdt[0] = x[1];
00028     dxdt[1] = 0.0; //x[0] + exp(t);
00029 }
00030
00031 TestODE::TestODE() {
00032 }
00033
00034 int TestODE::main(int argc, char** argv) {
00035     using namespace std;
00036     //using namespace boost::numeric::odeint;
00037     //using namespace boost::math;
00038
00039     //beta_distribution<>* betadist = new beta_distribution<>(2.0, 2.0);
00040     //betadist->alpha();
00041     ProbDistribDefaultImpl pd;
00042     for (double x = -2.0; x <= 2.0; x += 0.1) {
00043         std::cout << x << ":" << pd.gamma(x, 2.0, 2.0) << std::endl;
00044     }
00045     //normdist->mean();
00046     //beta<double,double>* b = new beta<double,double>(2.0, 2.0);
00047     //b->
00048
00049     state_type x(2);
00050
00051 /*
00052 x[0] = 1.0; // start at x=1.0, p=0.0
00053 x[1] = 0.0;
00054 runge_kutta4< state_type > stepper;
00055 integrate_const(stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00056 std::cout << x[0] << " ; " << x[1] << std::endl;
00057 */
00058
00059 /*
00060 x[0] = 1.0; // start at x=1.0, p=0.0
00061 x[1] = 0.0;
00062 const double dt = 0.01;
00063 double t;
00064 for (t = 0.0; t < 0.2; t += dt) {
00065     std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00066     stepper.do_step(harmonic_oscillator, x, t, dt);
00067 }
00068 std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00069 */
00070
00071 /*
00072 x[0] = 1.0; // start at x=1.0, p=0.0
00073 x[1] = 0.0;
00074 typedef runge_kutta_cash_karp54< state_type > error stepper_type;
00075 typedef controlled_runge_kutta< error stepper_type > controlled stepper_type;
00076 controlled stepper_type controlled stepper;
00077 //integrate_adaptive(controlled stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00078 integrate_adaptive(make controlled< error stepper_type >(1.0e-20, 1.0e-16), harmonic_oscillator,
00079 x, 0.0, 0.2, 0.01);
00080 std::cout << x[0] << " ; " << x[1] << std::endl;
00081 */
00082 return 0;
00083 }

```

9.488 TestFunctions.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
Include dependency graph for TestFunctions.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `TestODE`

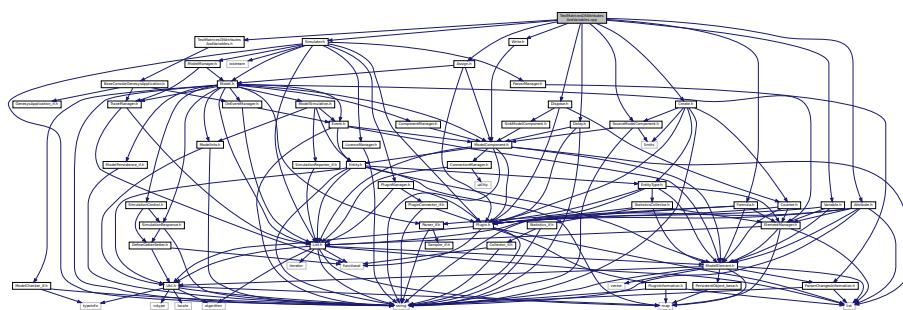
9.489 TestFunctions.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TestODE.h
00009  * Author:  rlcancian
00010  *
00011  * Created on 26 de Setembro de 2019, 13:00
00012 */
00013
00014 #ifndef TESTFUNCTIONS_H
00015 #define TESTFUNCTIONS_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestODE : public BaseConsoleGenesysApplication {
00020 public:
00021     TestODE();
00022 public:
00023     virtual int main(int argc, char** argv);
00024 };
00025
00026 #endif /* TESTFUNCTIONS_H */
00027
```

9.490 TestMatricesOfAttributesAndVariables.cpp File Reference

```
#include "TestMatricesOfAttributesAndVariables.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "Assign.h"
#include "Attribute.h"
#include "Variable.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Formula.h"
#include "Write.h"
```

Include dependency graph for TestMatricesOfAttributesAndVariables.cpp:



9.491 TestMatricesOfAttributesAndVariables.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestMatricesOfAttributesAndVariables.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 1 de Novembro de 2019, 18:10
00012 */
00013
00014 #include "TestMatricesOfAttributesAndVariables.h"
00015 #include "Simulator.h"
00016 #include "SourceModelComponent.h"
00017 #include "Assign.h"
00018 #include "Attribute.h"
00019 #include "Variable.h"
00020 #include "Create.h"
00021 #include "Delay.h"
00022 #include "Dispose.h"
00023 // #include "Separate.h"
00024 #include "Formula.h"
00025 #include "Write.h"
00026
00027 TestMatricesOfAttributesAndVariables::TestMatricesOfAttributes() {
00028 }
00029
00030 TestMatricesOfAttributesAndVariables::TestMatricesOfAttributes(const
00031     TestMatricesOfAttributesAndVariables& orig) {
00032 }
00033 TestMatricesOfAttributesAndVariables::~TestMatricesOfAttributes() {
00034 }
00035
00036 int TestMatricesOfAttributesAndVariables::main(int argc, char** argv) {
00037     Simulator* genesys = new Simulator();
00038     setDefaultTraceHandlers(genesys->getTracer());
00039     genesys->getTracer()->setTraceLevel(Util::TraceLevel::modelSimulationInternal);
00040     insertFakePluginsByHand(genesys);
```

```

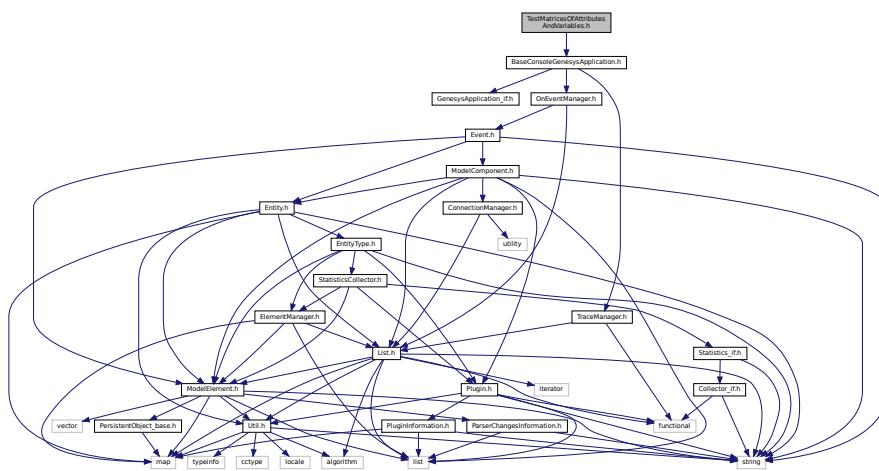
00041     Model* m = new Model(genesys);
00042     genesys->getModels()->insert(m);
00043     m->getInfos()->setProjectTitle("Stochastic Simulation of Chemical Reactions");
00044     m->getSimulation()->setReplicationLength(500);
00045     Create* cr1 = new Create(m);
00046     Write* wl = new Write(m);
00047     Assign* as1 = new Assign(m, "Define próxima reação a ocorrer");
00048     Delay* del = new Delay(m, "Aguarda tempo em que a reação ocorre");
00049     Dispose* dil = new Dispose(m);
00050     cr1->getNextComponents()->insert(wl);
00051     wl->getNextComponents()->insert(as1);
00052     as1->getNextComponents()->insert(del);
00053     del->getNextComponents()->insert(wl);
00054     del->getNextComponents()->insert(dil); // trick to be connected
00055     cr1->setEntityType(new EntityType(m));
00056     cr1->setMaxCreations("1");
00057     Variable* s = new Variable(m, "s");
00058     Variable* k = new Variable(m, "k");
00059     Variable* N = new Variable(m, "N");
00060     new Variable(m, "temp");
00061     Formula* prop = new Formula(m, "prop");
00062     s->setInitialValue("1,1", -1);
00063     s->setInitialValue("1,2", -1);
00064     s->setInitialValue("1,3", 1);
00065     s->setInitialValue("2,1", 1);
00066     s->setInitialValue("2,2", 1);
00067     s->setInitialValue("2,3", -1);
00068     k->setInitialValue("1", 0.1);
00069     k->setInitialValue("2", 0.2);
00070     N->setInitialValue("1", 100);
00071     N->setInitialValue("2", 100);
00072     N->setInitialValue("3", 0);
00073     prop->setExpression("1", "k[1]*N[1]*N[2]");
00074     prop->setExpression("2", "k[2]*N[3]");
00075     wl->setWriteToType(Write::WriteToType::FILE);
00076     wl->setFilename("./temp/molecules.txt");
00077     wl->writeElements()->insert(new WriteElement("tnow", true));
00078     wl->writeElements()->insert(new WriteElement(" "));
00079     wl->writeElements()->insert(new WriteElement("N[1]", true));
00080     wl->writeElements()->insert(new WriteElement(" "));
00081     wl->writeElements()->insert(new WriteElement("N[2]", true));
00082     wl->writeElements()->insert(new WriteElement(" "));
00083     wl->writeElements()->insert(new WriteElement("N[3]", true, true));
00084     //wl->writeElements()->insert(new WriteElement("temp[6]",true, true));
00085     //wl->writeElements()->insert(new WriteElement("tnow",true, true));
00086     as1->getAssignments()->insert(new Assign::Assignment("temp[1]", "k[1]*N[1]*N[2]"));
00087     as1->getAssignments()->insert(new Assign::Assignment("temp[2]", "k[2]*N[3]"));
00088     as1->getAssignments()->insert(new Assign::Assignment("temp[3]", "temp[1]+temp[2]"));
00089     as1->getAssignments()->insert(new Assign::Assignment("temp[4]", "if (temp[3]>0) temp[1]/temp[3] else 0"));
00090     as1->getAssignments()->insert(new Assign::Assignment("temp[5]", "if (temp[3]>0) (if (rnd<temp[4]) 1 else 2) else 0"));
00091     as1->getAssignments()->insert(new Assign::Assignment("N[1]", "N[1]+s[temp[5],1]"));
00092     as1->getAssignments()->insert(new Assign::Assignment("N[2]", "N[2]+s[temp[5],2]"));
00093     as1->getAssignments()->insert(new Assign::Assignment("N[3]", "N[3]+s[temp[5],3]"));
00094     as1->getAssignments()->insert(new Assign::Assignment("temp[6]", "1-exp(-temp[3]))"));
00095     as1->getAssignments()->insert(new Assign::Assignment("temp[7]", "expo(temp[6]))"));
00096     del->setDelayExpression("temp[7]");
00097     m->getSimulation()->setTerminatingCondition("(N[1]+N[2]+N[3]==0");
00098     m->getSimulation()->start();
00099     return 0;
00100
00101 /*
00102 Create* createl = new Create(m);
00103 Assign* assignl = new Assign(m);
00104 //Separate* sep1 = new Separate(m);
00105 Dispose* dispose1 = new Dispose(m);
00106 createl->nextComponents()->insert(assignl);
00107 assignl->nextComponents()->insert(assignl);
00108 assignl->nextComponents()->insert(dispose1);
00109 //sep1->nextComponents()->insert(dispose1);
00110 //sep1->nextComponents()->insert(assignl);
00111 createl->setEntityType(new EntityType(m));
00112 createl->setMaxCreations("1");
00113 new Attribute(m, "attr1");
00114 Variable* varl = new Variable(m, "varl");
00115 std::string expression, index;
00116 for (int i = 1; i < 3; i++) {
00117     for (int j = 1; j < 3; j++) {
00118         index = std::to_string(i) + "," + std::to_string(j);
00119         varl->setInitialValue(index, 1.0*i + j / 10.0);
00120         expression = "attr1[" + index + "] + varl[" + index + "]";
00121         assignl->assignments()->insert(new Assign::Assignment("attr1[" + index + "]", expression));
00122     }
00123 }
00124 m->simulation()->start();
00125 return 0;

```

00126 */
00127 }

9.492 TestMatricesOfAttributesAndVariables.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
Include dependency graph for TestMatricesOfAttributesAndVariables.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `TestMatricesOfAttributesAndVariables`

9.493 TestMatricesOfAttributesAndVariables.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006 /*  
00007 */  
00008 * File: TestMatricesOfAttributesAndVariables.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 1 de Novembro de 2019, 18:10  
00012 */  
00013  
00014 #ifndef TESTMATRICESOFATTRIBUTESANDVARIABLES_H  
00015 #define TESTMATRICESOFATTRIBUTESANDVARIABLES_H  
00016  
00017 #include "BaseConsoleGenesysApplication.h"  
00018  
00019 class TestMatricesOfAttributesAndVariables : public BaseConsoleGenesysApplication {  
00020 public:  
00021     TestMatricesOfAttributesAndVariables();  
00022     TestMatricesOfAttributesAndVariables(const TestMatricesOfAttributesAndVariables& orig);
```

```

00023     virtual ~TestMatricesOfAttributesAndVariables();
00024     virtual int main(int argc, char** argv);
00025 private:
00026 };
00028
00029 #endif /* TESTMATRICESOFATTRIBUTESANDVARIABLES_H */
00030

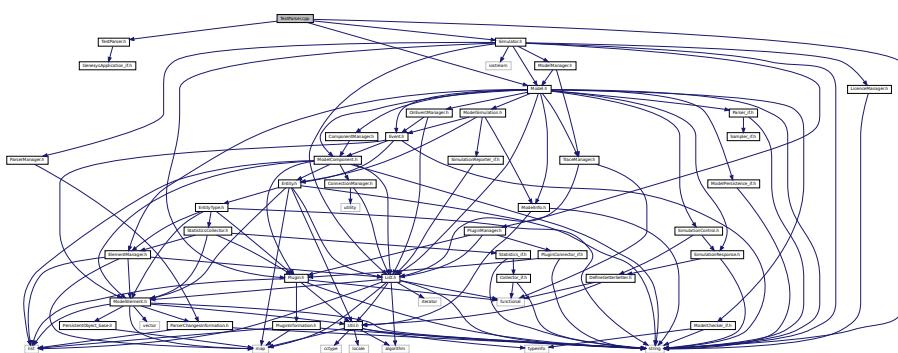
```

9.494 TestParser.cpp File Reference

```

#include "TestParser.h"
#include <string>
#include "Model.h"
#include "Simulator.h"
Include dependency graph for TestParser.cpp:

```



9.495 TestParser.cpp

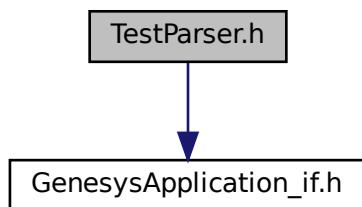
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestParser.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 20:17
00012 */
00013
00014 #include "TestParser.h"
00015 #include <string>
00016 #include "Model.h"
00017 #include "Simulator.h"
00018
00019 TestParser::TestParser() {
00020 }
00021
00022 int TestParser::main(int argc, char** argv) {
00023     Simulator* simulator = new Simulator();
00024     Model* model = new Model(simulator);
00025     simulator->getModels()->insert(model);
00026     double value;
00027     bool success;
00028     std::string errorMsg = "";
00029     value = model->parseExpression("NORM(20,10)", &success, &errorMsg);
00030     std::cout << value << std::endl;
00031     value = model->parseExpression("-10+1+2+3", &success, &errorMsg);
00032     std::cout << value << std::endl;
00033     value = model->parseExpression("1+2+3-10", &success, &errorMsg);
00034     std::cout << value << std::endl;
00035     return 0;
00036 }

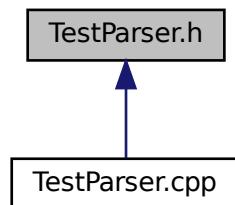
```

9.496 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](#)

9.497 TestParser.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestParser.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 20:17
00012 */
00013
00014 #ifndef TESTPARSER_H
00015 #define TESTPARSER_H
00016
00017 #include "GenesysApplication_if.h"
00018
  
```

```

00019 class TestParser : public GenesysApplication_if {
00020 public:
00021     TestParser();
00022     virtual ~TestParser() = default;
00023 public:
00024     virtual int main(int argc, char** argv);
00025 private:
00026
00027 };
00028
00029 #endif /* TESTPARSER_H */
00030

```

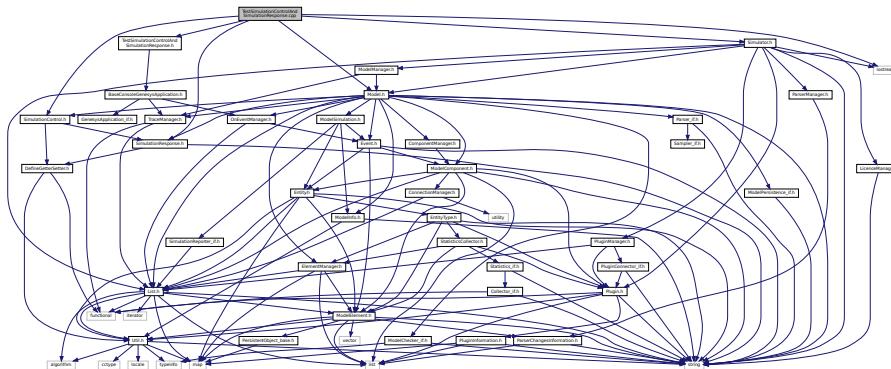
9.498 TestSimulationControlAndSimulationResponse.cpp File Reference

```

#include "TestSimulationControlAndSimulationResponse.h"
#include "Simulator.h"
#include "Model.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include <iostream>

```

Include dependency graph for TestSimulationControlAndSimulationResponse.cpp:



9.499 TestSimulationControlAndSimulationResponse.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: TestSimulationControlAndSimulationResponse.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 17:44
00012 */
00013
00014 #include "TestSimulationControlAndSimulationResponse.h"
00015 #include "Simulator.h"
00016 #include "Model.h"
00017 #include "SimulationControl.h"
00018 #include "SimulationResponse.h"
00019 #include <iostream>
00020
00021 TestSimulationControlAndSimulationResponse::TestSimulationControlAndSimulationResponse() {
00022 }
00023
00024 int TestSimulationControlAndSimulationResponse::main(int argc, char** argv) {
00025     Simulator* simulator = new Simulator();
00026     TraceManager* tm = simulator->getTracer();
00027     this->setDefaultTraceHandlers(tm);
00028     tm->setTraceLevel(Util::TraceLevel::everythingMostDetailed);
00029     this->insertFakePluginsByHand(simulator);

```

```

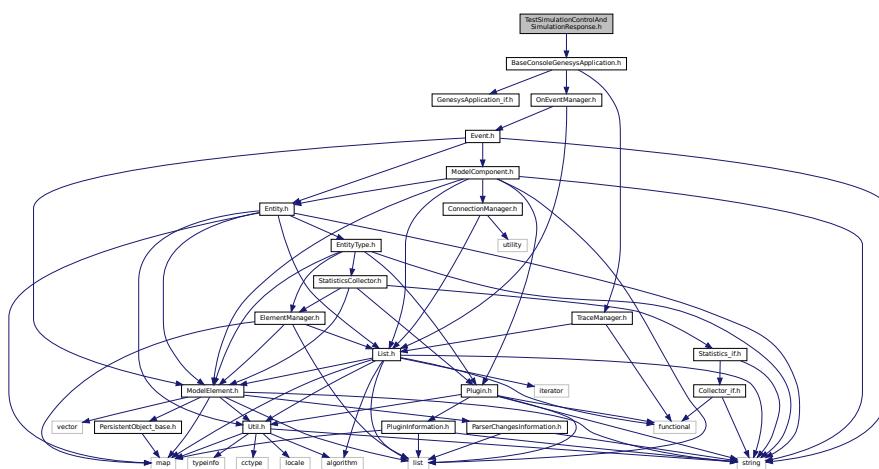
00030     simulator->getModels()->loadModel("./temp/forthExampleOfSimulation.txt");
00031     Model* model = simulator->getModels()->current();
00032     model->getSimulation()->start();
00033     tm->trace("\nResponses:");
00034     for (std::list<SimulationResponse*>::iterator it = model->getResponses()->list()->begin(); it != model->getResponses()->list()->end(); it++) {
00035         tm->trace((*it)->getName() + ": " + std::to_string((*it)->getValue()));
00036     }
00037 }
00038 */
00039 Model* model = new Model(simulator);
00040 model->show();
00041
00042 std::cout << "NumRepl antes: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00043 model->getInfos()->setNumberOfReplications(10);
00044 std::cout << "NumRepl depois: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00045 SimulationControl* control = model->getControls()->front();
00046 std::cout << control->getName() << " antes: " << control->getValue() << std::endl;
00047 control->setValue(20);
00048 std::cout << control->getName() << " depois: " << control->getValue() << std::endl;
00049 std::cout << "NumRepl depois: " << model->getSimulation()->getNumberOfReplications() << std::endl;
00050 */
00051
00052 return 0;
00053 }

```

9.500 TestSimulationControlAndSimulationResponse.h File Reference

#include "BaseConsoleGenesysApplication.h"

Include dependency graph for TestSimulationControlAndSimulationResponse.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TestSimulationControlAndSimulationResponse](#)

9.501 TestSimulationControlAndSimulationResponse.h

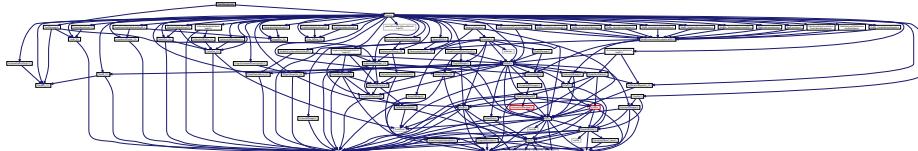
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: TestSimulationControlAndSimulationResponse.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 17:44
00012 */
00013
00014 #ifndef TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H
00015 #define TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestSimulationControlAndSimulationResponse : public BaseConsoleGenesysApplication {
00020 public:
00021     TestSimulationControlAndSimulationResponse();
00022     virtual ~TestSimulationControlAndSimulationResponse() = default;
00023 public:
00024     virtual int main(int argc, char** argv);
00025 private:
00026
00027 };
00028
00029 #endif /* TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H */
00030

```

9.502 ToolManager.cpp File Reference

```
#include "ToolManager.h"
#include "Traits.h"
Include dependency graph for ToolManager.cpp:
```



9.503 ToolManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ToolManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 09:04
00012 */
00013
00014 #include "ToolManager.h"
00015 #include "Traits.h"
00016
00017 ToolManager::ToolManager(Simulator* simulator) {
00018     _simulator = simulator;
00019     /**
00020     _fitter = new Traits<Fitter_if>::Implementation();
00021     _sampler = new Traits<Sampler_if>::Implementation();
00022     _processAnalyser = new Traits<ExperimentManager_if>::Implementation();
00023 }
00024
00025 Sampler_if* ToolManager::sampler() const {
00026     return _sampler;
00027 }

```

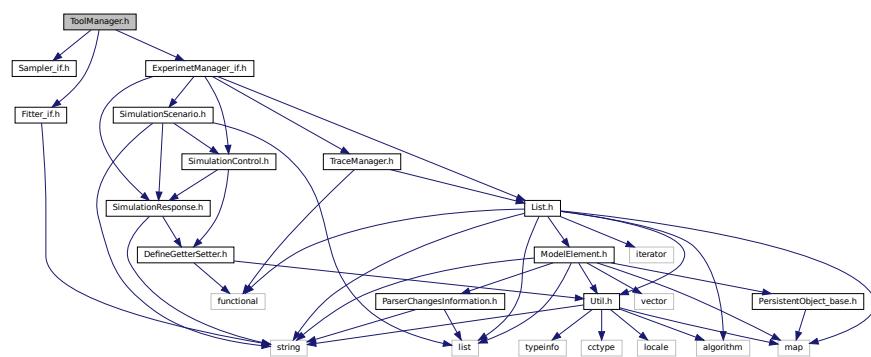
```

00027 }
00028
00029 Fitter_if* ToolManager::fitter() const {
00030     return _fitter;
00031 }
00032
00033 ExperimentManager_if* ToolManager::experimentDesigner() const {
00034     return _processAnalyser;
00035 }

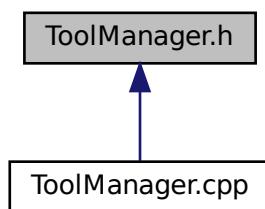
```

9.504 ToolManager.h File Reference

```
#include "Sampler_if.h"
#include "Fitter_if.h"
#include "ExperimentManager_if.h"
Include dependency graph for ToolManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ToolManager](#)

9.505 ToolManager.h

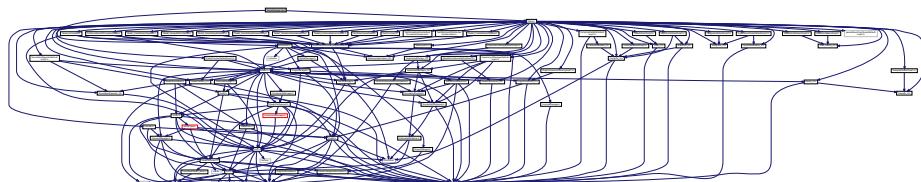
```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: ToolManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 09:04
00012 */
00013
00014 #ifndef TOOLMANAGER_H
00015 #define TOOLMANAGER_H
00016
00017 #include "Sampler_if.h"
00018 #include "Fitter_if.h"
00019 #include "ExperimentManager_if.h"
00020
00021 class Simulator;
00022
00023 class ToolManager {
00024 public:
00025     ToolManager(Simulator* _simulator);
00026     virtual ~ToolManager() = default;
00027 public:
00028     Sampler_if* sampler() const;
00029     Fitter_if* fitter() const;
00030     ExperimentManager_if* experimentDesigner() const;
00031 public: // event handlers
00032 private:
00033     Fitter_if* _fitter; // = new Traits<Fitter_if>::Implementation();
00034     Sampler_if* _sampler; // = new Traits<Sampler_if>::Implementation();
00035     ExperimentManager_if* _processAnalyser;
00036 private:
00037     Simulator* _simulator;
00038 };
00039
00040 #endif /* TOOLMANAGER_H */
00041

```

9.506 TraceManager.cpp File Reference

```
#include "TraceManager.h"
#include "Traits.h"
Include dependency graph for TraceManager.cpp:
```



9.507 TraceManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TraceManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 11:59
00012 */
00013
00014 #include "TraceManager.h"

```

```

00015 #include "Traits.h"
00016
00017 //using namespace GenesysKernel;
00018
00019 TraceManager::TraceManager(Simulator* simulator) { // (Model* model) {
00020     _simulator = simulator;
00021     _debugged = Traits<Model>::debugged;
00022     _traceLevel = Traits<Model>::traceLevel;
00023 }
00024
00025 void TraceManager::setTraceLevel(Util::TraceLevel _traceLevel) {
00026     this->_traceLevel = _traceLevel;
00027 }
00028
00029 Util::TraceLevel TraceManager::getTraceLevel() const {
00030     return _traceLevel;
00031 }
00032
00033 Simulator* TraceManager::getParentSimulator() const {
00034     return _simulator;
00035 }
00036
00037 void TraceManager::addTraceHandler(traceListener traceListener) {
00038     this->_traceHandlers->insert(traceListener);
00039 }
00040
00041 void TraceManager::addTraceSimulationHandler(traceSimulationListener traceSimulationListener) {
00042     this->_traceSimulationHandlers->insert(traceSimulationListener);
00043 }
00044
00045 void TraceManager::addTraceErrorHandler(traceErrorListener traceErrorListener) {
00046     this->_traceErrorHandlers->insert(traceErrorListener);
00047 }
00048
00049 void TraceManager::addTraceReportHandler(traceListener traceReportListener) {
00050     this->_traceReportHandlers->insert(traceReportListener);
00051 }
00052
00053 void TraceManager::trace(Util::TraceLevel level, std::string text) {
00054     trace(text, level);
00055 }
00056
00057 void TraceManager::trace(std::string text, Util::TraceLevel level) {
00058     if (_traceConditionPassed(level)) {
00059         //text = std::to_string(static_cast<int> (level)) + ". " + Util::Indent() + text;
00060         text = Util::Indent() + text;
00061         TraceEvent e = TraceEvent(text, level);
00062         /* \todo---: somewhere in future it should be interesting to use "auto" and c++17 at least */
00063         for (std::list<traceListener>::iterator it = this->_traceHandlers->list()->begin(); it != _traceHandlers->list()->end(); it++) {
00064             (*it)(e);
00065         }
00066         for (std::list<traceListenerMethod>::iterator it =
00067             this->_traceHandlersMethod->list()->begin(); it != _traceHandlersMethod->list()->end(); it++) {
00068             (*it)(e);
00069         }
00070     }
00071 }
00072
00073 void TraceManager::traceError(std::exception e, std::string text) {
00074     traceError(text, e);
00075 }
00076
00077 void TraceManager::traceError(std::string text, std::exception e) {
00078     text = Util::Indent() + text;
00079     TraceErrorEvent exceptEvent = TraceErrorEvent(text, e);
00080     /* \todo---: somewhere in future it should be interesting to use "auto" and c++17 at least */
00081     for (std::list<traceErrorListener>::iterator it = this->_traceErrorHandlers->list()->begin(); it != _traceErrorHandlers->list()->end(); it++) {
00082         (*it)(exceptEvent);
00083     }
00084     for (std::list<traceErrorListenerMethod>::iterator it =
00085         this->_traceErrorHandlersMethod->list()->begin(); it != _traceErrorHandlersMethod->list()->end(); it++) {
00086         (*it)(exceptEvent);
00087     }
00088
00089 void TraceManager::traceSimulation(Util::TraceLevel level, double time, Entity* entity,
00090     ModelComponent* component, std::string text) {
00091     traceSimulation(time, entity, component, text, level);
00092 }
00093 void TraceManager::traceSimulation(double time, Entity* entity, ModelComponent* component, std::string
00094     text, Util::TraceLevel level) {
00095     if (_traceConditionPassed(level)) {

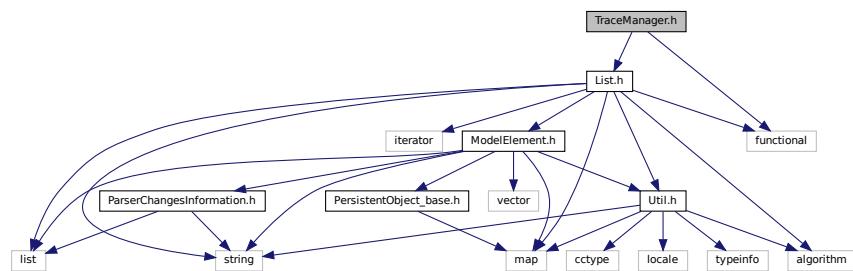
```

```

00095     text = Util::Indent() + text;
00096     TraceSimulationEvent e = TraceSimulationEvent(level, time, entity, component, text);
00097     for (std::list<traceSimulationListener>::iterator it =
00098         this->_traceSimulationHandlers->list()->begin(); it != _traceSimulationHandlers->list()->end(); it++)
00099     {
00100         (*it)(e);
00101     }
00102     for (std::list<traceSimulationListenerMethod>::iterator it =
00103         this->_traceSimulationHandlersMethod->list()->begin(); it != _traceSimulationHandlersMethod->list()->end(); it++)
00104     {
00105         (*it)(e);
00106     }
00107 }
00108
00109 void TraceManager::traceReport(Util::TraceLevel level, std::string text) {
00110     traceReport(text, level);
00111 }
00112
00113 void TraceManager::traceReport(std::string text, Util::TraceLevel level) {
00114     if (_traceConditionPassed(level)) {
00115         text = Util::Indent() + text;
00116         TraceEvent e = TraceEvent(text, level);
00117         for (std::list<traceListener>::iterator it = this->_traceReportHandlers->list()->begin(); it !=
00118             _traceReportHandlers->list()->end(); it++) {
00119             (*it)(e);
00120         }
00121     }
00122 }
00123 List<std::string>* TraceManager::errorMessages() const {
00124     return _errorMessages;
00125 }
00126
00127 bool TraceManager::_traceConditionPassed(Util::TraceLevel level) {
00128     return this->_debugged && static_cast<int> (this->_traceLevel) >= static_cast<int> (level);
00129 }
```

9.508 TraceManager.h File Reference

```
#include "List.h"
#include <functional>
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)`
- `typedef std::function< void(TraceEvent) > traceListenerMethod`
- `typedef std::function< void(TraceErrorEvent) > traceErrorListenerMethod`
- `typedef std::function< void(TraceSimulationEvent) > traceSimulationListenerMethod`
- `typedef std::function< void(TraceSimulationProcess) > traceSimulationProcessListenerMethod`

9.508.1 Typedef Documentation

9.508.1.1 **traceErrorListener** `typedef void(* traceErrorListener) (TraceErrorEvent)`

Definition at line 104 of file [TraceManager.h](#).

9.508.1.2 **traceErrorListenerMethod** `typedef std::function<void(TraceErrorEvent) > traceErrorListenerMethod`

Definition at line 109 of file [TraceManager.h](#).

9.508.1.3 **traceListener** `typedef void(* traceListener) (TraceEvent)`

Definition at line 103 of file [TraceManager.h](#).

9.508.1.4 **traceListenerMethod** `typedef std::function<void(TraceEvent) > traceListenerMethod`

Definition at line 108 of file [TraceManager.h](#).

9.508.1.5 traceSimulationListener `typedef void(* traceSimulationListener) (TraceSimulationEvent)`

Definition at line 105 of file [TraceManager.h](#).

9.508.1.6 traceSimulationListenerMethod `typedef std::function<void(TraceSimulationEvent)> traceSimulationListenerMethod`

Definition at line 110 of file [TraceManager.h](#).

9.508.1.7 traceSimulationProcessListener `typedef void(* traceSimulationProcessListener) (TraceSimulationProcess)`

Definition at line 106 of file [TraceManager.h](#).

9.508.1.8 traceSimulationProcessListenerMethod `typedef std::function<void(TraceSimulationProcess)> traceSimulationProcessListenerMethod`

Definition at line 111 of file [TraceManager.h](#).

9.509 TraceManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TraceManager.h
00009  * Author:  rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 11:59
00012 */
00013
00014 #ifndef TRACEMANAGER_H
00015 #define TRACEMANAGER_H
00016
00017 #include "List.h"
00018 #include <functional>
00019 //namespace GenesysKernel {
00020     class Model;
00021     class Entity;
00022     class ModelComponent;
00023     class Simulator;
00024
00025     class TraceEvent {
00026     public:
00027
00028         TraceEvent(Util::TraceLevel level, std::string text) {
00029             _tracelevel = level;
00030             _text = text;
00031         }
00032
00033         TraceEvent(std::string text, Util::TraceLevel level = Util::TraceLevel::componentInternal) {
00034             _tracelevel = level;
00035             _text = text;
00036         }
00037
00038         Util::TraceLevel getTracelevel() const {
00039             return _tracelevel;
00040         }
00041
00042         std::string getText() const {
```

```

00043         return _text;
00044     }
00045     private:
00046     Util::TraceLevel _tracelevel;
00047     std::string _text;
00048 };
00049
00050 class TraceErrorEvent : public TraceEvent {
00051 public:
00052
00053     TraceErrorEvent(std::string text, std::exception e) : TraceEvent(text,
00054         Util::TraceLevel::errorFatal) {
00055         _e = e;
00056     }
00057     std::exception getException() const {
00058         return _e;
00059     }
00060     private:
00061     std::exception _e;
00062 };
00063
00064 class TraceSimulationEvent : public TraceEvent {
00065 public:
00066
00067     TraceSimulationEvent(Util::TraceLevel level, double time, Entity* entity, ModelComponent*
00068     component, std::string text) : TraceEvent(text, level) {
00069         _time = time;
00070         _entity = entity;
00071         _component = component;
00072     }
00073     ModelComponent* getComponent() const {
00074         return _component;
00075     }
00076
00077     Entity* getEntity() const {
00078         return _entity;
00079     }
00080
00081     double getTime() const {
00082         return _time;
00083     }
00084
00085     private:
00086     double _time;
00087     Entity* _entity;
00088     ModelComponent* _component;
00089 };
00090
00091 class TraceSimulationProcess : public TraceEvent {
00092 public:
00093
00094     TraceSimulationProcess(std::string text, Util::TraceLevel
00095     level=Util::TraceLevel::modelSimulationEvent) : TraceEvent(text, level) {
00096
00097 };
00098
00099 // for handlers that are simple functions
00100 typedef void (*traceListener)(TraceEvent);
00101 typedef void (*traceErrorListener)(TraceErrorEvent);
00102 typedef void (*traceSimulationListener)(TraceSimulationEvent);
00103 typedef void (*traceSimulationProcessListener)(TraceSimulationProcess);
00104 // for handlers that are class members (methods)
00105 typedef std::function<void(TraceEvent) > traceListenerMethod;
00106 typedef std::function<void(TraceErrorEvent) > traceErrorListenerMethod;
00107 typedef std::function<void(TraceSimulationEvent) > traceSimulationListenerMethod;
00108 typedef std::function<void(TraceSimulationProcess) > traceSimulationProcessListenerMethod;
00109
00110 class TraceManager {
00111 public:
00112     TraceManager(Simulator* simulator); // (Model* model);
00113     virtual ~TraceManager() = default;
00114     public: // add trace handlers
00115     // for handlers that are simple functions
00116     void addTraceHandler(traceListener traceListener);
00117     void addTraceReportHandler(traceListener traceReportListener);
00118     void addTraceSimulationHandler(traceSimulationListener traceSimulationListener);
00119     void addTraceErrorHandler(traceErrorListener traceErrorListener);
00120     // for handlers that are class members (methods)
00121     template<typename Class> void addTraceHandler(Class * object, void
00122     (Class::*function)(TraceEvent));
00123     template<typename Class> void addTraceErrorHandler(Class * object, void
00124     (Class::*function)(TraceErrorEvent));
00125     template<typename Class> void addTraceReportHandler(Class * object, void
00126     (Class::*function)(TraceEvent));
00127     template<typename Class> void addTraceSimulationHandler(Class * object, void
00128     (Class::*function)(TraceSimulationEvent));
00129
00130
00131

```

```

        (Class::*function) (TraceSimulationEvent));
00132     public: // traces (invoke trace handlers)
00133         void trace(Util::TraceLevel level, std::string text);
00134         void traceError(std::exception e, std::string text);
00135         void traceReport(Util::TraceLevel level, std::string text);
00136         void traceSimulation(Util::TraceLevel level, double time, Entity* entity, ModelComponent*
component, std::string text);
00137     public: // traces (invoke trace handlers) SINCE 20191025 NEW TRACES JUST INVERTED THE PARAMETERS,
MAKING TRACELEVEL OPTIONAL
00138         void trace(std::string text, Util::TraceLevel level = Util::TraceLevel::componentInternal);
00139         void traceError(std::string text, std::exception e);
00140         void traceReport(std::string text, Util::TraceLevel level = Util::TraceLevel::report);
00141         void traceSimulation(double time, Entity* entity, ModelComponent* component, std::string text,
Util::TraceLevel level = Util::TraceLevel::componentInternal);
00142     public:
00143         List<std::string>* errorMessages() const;
00144         void setTraceLevel(Util::TraceLevel _traceLevel);
00145         Util::TraceLevel getTraceLevel() const;
00146         Simulator* getParentSimulator() const;
00147     private:
00148         //void _addHandler(List<traceListener>* list, )
00149         bool _traceConditionPassed(Util::TraceLevel level);
00150     private: // trace listener
00151         // for handlers that are simple functions
00152         List<traceListener>* _traceHandlers = new List<traceListener>();
00153         List<traceErrorListener>* _traceErrorHandlers = new List<traceErrorListener>();
00154         List<traceListener>* _traceReportHandlers = new List<traceListener>();
00155         List<traceSimulationListener>* _traceSimulationHandlers = new List<traceSimulationListener>();
00156         // for handlers that are class members (methods)
00157         List<traceListenerMethod>* _traceHandlersMethod = new List<traceListenerMethod>();
00158         List<traceErrorListenerMethod>* _traceErrorHandlersMethod = new
List<traceErrorListenerMethod>();
00159         List<traceListenerMethod>* _traceReportHandlersMethod = new List<traceListenerMethod>();
00160         List<traceSimulationListenerMethod>* _traceSimulationHandlersMethod = new
List<traceSimulationListenerMethod>();
00161     private:
00162         //Model* _model;
00163         Simulator* _simulator;
00164     private:
00165         Util::TraceLevel _traceLevel; // = Util::TraceLevel::mostDetailed;
00166         bool _debugged;
00167         double _lastTimeTraceSimulation = -1.0;
00168         Util::identification _lastEntityTraceSimulation = 0;
00169         Util::identification _lastModuleTraceSimulation = 0;
00170         List<std::string>* _errorMessages; /* \todo: 18/08/24 this is a new one. several methods
should use it */
00171     };
00172
00173     // implementation for template methods
00174
00175     template<typename Class> void TraceManager::addTraceHandler(Class * object, void
(Class::*function) (TraceEvent)) {
00176         this->_traceHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00177     }
00178
00179     template<typename Class> void TraceManager::addTraceErrorHandler(Class * object, void
(Class::*function) (TraceErrorEvent)) {
00180         this->_traceErrorHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00181     }
00182
00183     template<typename Class> void TraceManager::addTraceReportHandler(Class * object, void
(Class::*function) (TraceEvent)) {
00184         this->_traceReportHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00185     }
00186
00187     template<typename Class> void TraceManager::addTraceSimulationHandler(Class * object, void
(Class::*function) (TraceSimulationEvent)) {
00188         this->_traceSimulationHandlersMethod->insert(std::bind(function, object,
std::placeholders::_1));
00189     }
00190
00191 //namespace\\}
00192 #endif /* TRACEMANAGER_H */
00193

```

9.510 Traits.h File Reference

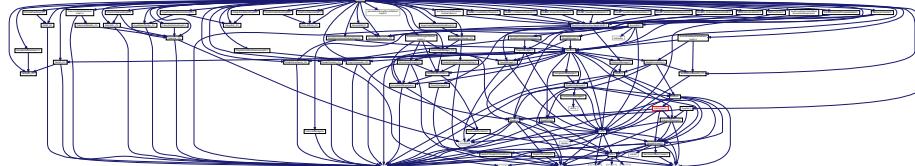
```

#include "Collector_if.h"
#include "ExperimentDesign_if.h"
#include "Fitter_if.h"
#include "GenesysApplication_if.h"

```

```
#include "HypothesisTester_if.h"
#include "Integrator_if.h"
#include "Model.h"
#include "ModelChecker_if.h"
#include "ModelPersistence_if.h"
#include "Parser_if.h"
#include "PluginConnector_if.h"
#include "Prob_Distrib_if.h"
#include "ExperimentManager_if.h"
#include "Sampler_if.h"
#include "Statistics_if.h"
#include "SimulationReporter_if.h"
#include "CollectorDefaultImpl1.h"
#include "CollectorDatafileDefaultImpl1.h"
#include "HypothesisTesterDefaultImpl1.h"
#include "IntegratorDefaultImpl1.h"
#include "ProbDistribDefaultImpl1.h"
#include "ProbDistribBoostImpl.h"
#include "SamplerDefaultImpl1.h"
#include "StatisticsDefaultImpl1.h"
#include "parserBisonFlex/ParserDefaultImpl2.h"
#include "PluginConnectorDummyImpl1.h"
#include "ModelCheckerDefaultImpl1.h"
#include "ModelPersistenceDefaultImpl1.h"
#include "SimulationReporterDefaultImpl1.h"
#include "ExperimentDesignDefaultImpl1.h"
#include "FitterDefaultImpl1.h"
#include "ExperimentManagerDefaultImpl1.h"
#include "Counter.h"
#include "FullSimulationOfComplexModel.h"
#include "Model_CreateDelayDispose.h"
#include "Model_CreteDelayDispose2.h"
#include "Model_AssignWrite3Seizes.h"
#include "GenesysGUI.h"
#include "GenesysConsole.h"
#include "Model_SeizeDelayRelease1.h"
#include "Model_SeizeDelayReleaseMany.h"
#include "Model_StatationRouteSequence.h"
#include "Modelo_SistemaOperacional02.h"
#include "TestEnterLeaveRoute.h"
#include "TestFunctions.h"
#include "TestSimulationControlAndSimulationResponse.h"
#include "TestMatricesOfAttributesAndVariables.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< PluginConnector_if >`
- struct `Traits< Parser_if >`
- struct `Traits< Model >`
- struct `Traits< ModelPersistence_if >`
- struct `Traits< SimulationReporter_if >`
- struct `Traits< ModelChecker_if >`
- struct `Traits< ModelComponent >`
- struct `Traits< ModelElement >`
- struct `Traits< Collector_if >`
- struct `Traits< Statistics_if >`
- struct `Traits< Integrator_if >`
- struct `Traits< Sampler_if >`
- struct `Traits< ProbDistrib_if >`
- struct `Traits< Fitter_if >`
- struct `Traits< HypothesisTester_if >`
- struct `Traits< ExperimentDesign_if >`
- struct `Traits< ExperimentManager_if >`

9.511 Traits.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Traits.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 19:36
00012 */
00013
00014 #ifndef TRAITS_H
00015 #define TRAITS_H
00016
00017 // \todo: Separate into Traits.h and KernelTraits.h
00018
00019 // interfaces
00020 #include "Collector_if.h"
00021 #include "ExperimentDesign_if.h"
00022 #include "Fitter_if.h"
00023 #include "GenesysApplication_if.h"
00024 #include "HypothesisTester_if.h"
00025 #include "Integrator_if.h"
00026 #include "Model.h"
00027 #include "ModelChecker_if.h"
00028 #include "ModelPersistence_if.h"
00029 #include "Parser_if.h"
00030 #include "PluginConnector_if.h"
00031 #include "Prob_Distrib_if.h"
00032 #include "ExperimentManager_if.h"
00033 #include "Sampler_if.h"
00034 #include "Statistics_if.h"
00035 #include "SimulationReporter_if.h"
00036
00037 // Default implementations
00038 //statistics
00039 #include "CollectorDefaultImpl1.h"
00040 #include "CollectorDatafileDefaultImpl1.h"
00041 #include "HypothesisTesterDefaultImpl1.h"
00042 #include "IntegratorDefaultImpl1.h"
00043 #include "ProbDistribDefaultImpl1.h"
00044 #include "ProbDistribBoostImpl.h"
00045 #include "SamplerDefaultImpl1.h"
00046 #include "StatisticsDefaultImpl1.h"
00047 //simulator and parts
00048 #include "parserBisonFlex/ParserDefaultImpl2.h"

```

```

00049 #include "PluginConnectorDummyImpl1.h"
00050 //model and parts
00051 #include "ModelCheckerDefaultImpl1.h"
00052 #include "ModelPersistenceDefaultImpl1.h"
00053 #include "SimulationReporterDefaultImpl1.h"
00054 //tools
00055 #include "ExperimentDesignDefaultImpl1.h"
00056 #include "FitterDefaultImpl1.h"
00057 #include "ExperimentManagerDefaultImpl1.h"
00058
00059
00060 // simulationReporter issues
00061 #include "Counter.h"
00062
00063 // genesys applications
00064 #include "FullSimulationOfComplexModel.h"
00065 //##include "BuildSimulationModel03.h"
00066 #include "Model_CreateDelayDispose.h"
00067 #include "Model_CreteDelayDispose2.h"
00068 #include "Model_AssignWrite3Seizes.h"
00069 #include "GenesysGUI.h"
00070 #include "GenesysConsole.h"
00071 #include "Model_SeizeDelayRelease1.h"
00072 #include "Model_SeizeDelayReleaseMany.h"
00073 #include "Model_StatationRouteSequence.h"
00074 #include "Modelo_SistemaOperacional02.h"
00075 #include "TestEnterLeaveRoute.h"
00076 #include "TestFunctions.h"
00077 #include "TestSimulationControlAndSimulationResponse.h"
00078 //##include "TestLSODE.h"
00079 //##include "TestMarkovChain.h"
00080 #include "TestMatricesOfAttributesAndVariables.h"
00081
00082 template <typename T>
00083 struct Traits {
00084 };
00085
00086 /*
00087  * KERNEL STUFF
00088 */
00089
00090 /*
00091  * Genesys Application
00092 */
00093
00094 template <> struct Traits<GenesysApplication_if> {
00095     //typedef Model_CreateDelayDispose Application;
00096     //typedef Model_SeizeDelayRelease1 Application;
00097     //typedef Model_SeizeDelayReleaseMany Application;
00098     typedef Modelo_SistemaOperacional02 Application;
00099 };
00100
00101 /*
00102  * Simulator and Simulator Parts
00103 */
00104
00105
00106 template <> struct Traits<PluginConnector_if> {
00107     typedef PluginConnectorDummyImpl1 Implementation;
00108 };
00109
00110 template <> struct Traits<Parser_if> {
00111     typedef ParserDefaultImpl2 Implementation;
00112 };
00113
00114 /*
00115  * Model and Model Parts
00116 */
00117
00118 template <> struct Traits<Model> {
00119     static const bool debugged = true;
00120     static const Util::TraceLevel traceLevel = Util::TraceLevel::modelSimulationEvent;
00121 };
00122
00123 template <> struct Traits<ModelPersistence_if> {
00124     typedef ModelPersistenceDefaultImpl1 Implementation;
00125 };
00126
00127 template <> struct Traits<SimulationReporter_if> {
00128     typedef SimulationReporterDefaultImpl1 Implementation;
00129     typedef Counter CounterImplementation;
00130 };
00131
00132 template <> struct Traits<ModelChecker_if> {
00133     typedef ModelCheckerDefaultImpl1 Implementation;
00134 };
00135

```

```

00136 template <> struct Traits<ModelComponent> {
00137     typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation;
00138     typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation;
00139     static constexpr bool reportStatistics = true;
00140 };
00141 template <> struct Traits<ModelElement> {
00142     static constexpr bool reportStatistics = true;
00143 };
00144 /*
00145  * Statistics
00146 */
00147 */
00148
00149 template <> struct Traits<Collector_if> {
00150     typedef CollectorDatafileDefaultImpl1 Implementation;
00151 };
00152
00153 template <> struct Traits<Statistics_if> {
00154     typedef StatisticsDefaultImpl1 Implementation;
00155     typedef CollectorDefaultImpl1 CollectorImplementation;
00156     static constexpr double SignificanceLevel = 0.05;
00157 };
00158
00159 template <> struct Traits<Integrator_if> {
00160     typedef IntegratorDefaultImpl1 Implementation;
00161     static constexpr unsigned int MaxIterations = 1e3;
00162     static constexpr double Precision = 1e-9;
00163 };
00164
00165 template <> struct Traits<Sampler_if> {
00166     typedef SamplerDefaultImpl1 Implementation;
00167     typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters;
00168 };
00169
00170 template <> struct Traits<ProbDistrib_if> {
00171     typedef ProbDistribDefaultImpl1 Implementation;
00172 };
00173
00174 template <> struct Traits<Fitter_if> {
00175     typedef FitterDefaultImpl1 Implementation;
00176 };
00177
00178 template <> struct Traits<HypothesisTester_if> {
00179     typedef IntegratorDefaultImpl1 IntegratorImplementation;
00180     typedef HypothesisTesterDefaultImpl1 Implementation;
00181 };
00182
00183 /*
00184 * Tools
00185 */
00186
00187 template <> struct Traits<ExperimentDesign_if> {
00188     typedef ExperimentDesignDefaultImpl1 Implementation;
00189 };
00190
00191 template <> struct Traits<ExperimentManager_if> {
00192     typedef ExperimentManagerDefaultImpl1 Implementation;
00193 };
00194
00195 #endif /* TRAITS_H */
00196

```

9.512 TraitsKernel.h File Reference

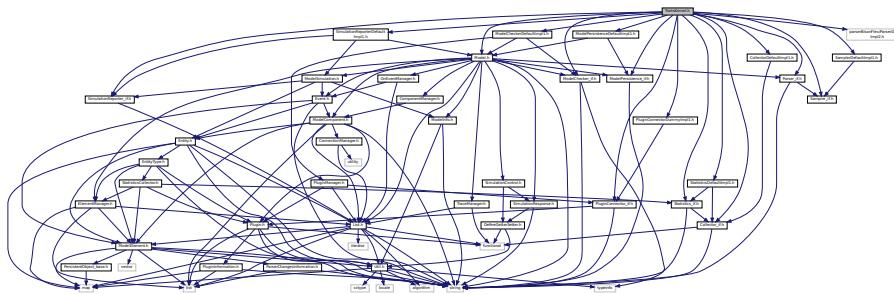
```

#include "Model.h"
#include "Collector_if.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "Statistics_if.h"
#include "ModelPersistence_if.h"
#include "SimulationReporter_if.h"
#include "PluginConnector_if.h"
#include "Sampler_if.h"
#include "PluginConnectorDummyImpl1.h"
#include "SimulationReporterDefaultImpl1.h"
#include "ModelCheckerDefaultImpl1.h"

```

```
#include "ModelPersistenceDefaultImpl1.h"
#include "parserBisonFlex/ParserDefaultImpl2.h"
#include "StatisticsDefaultImpl1.h"
#include "CollectorDefaultImpl1.h"
#include "SamplerDefaultImpl1.h"
```

Include dependency graph for TraitsKernel.h:



Classes

- struct [GenesysKernel::Traits< T >](#)
- struct [GenesysKernel::Traits< Model >](#)
- struct [GenesysKernel::Traits< ModelPersistence_if >](#)
- struct [GenesysKernel::Traits< SimulationReporter_if >](#)
- struct [GenesysKernel::Traits< ModelChecker_if >](#)
- struct [GenesysKernel::Traits< ModelComponent >](#)
- struct [GenesysKernel::Traits< Sampler_if >](#)
- struct [GenesysKernel::Traits< Parser_if >](#)
- struct [GenesysKernel::Traits< PluginConnector_if >](#)

Namespaces

- [GenesysKernel](#)

9.513 TraitsKernel.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Traits.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 19:36
00012 */
00013
00014
00015 #ifndef TRAITS_H
00016 #define TRAITS_H
00017
00018
00019 // interfaces
00020 #include "Model.h"
00021 #include "Collector_if.h"
00022 #include "ModelChecker_if.h"
00023 #include "Parser_if.h"
00024 #include "Statistics_if.h"
00025 #include "ModelPersistence_if.h"
00026 #include "SimulationReporter_if.h"
```

```

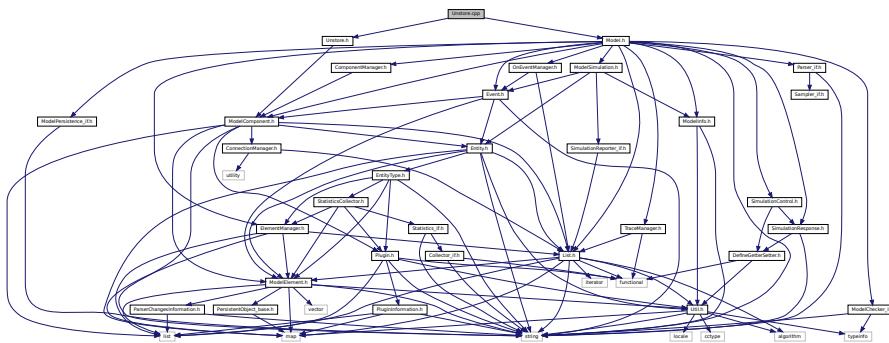
00027 #include "PluginConnector_if.h"
00028 #include "Sampler_if.h"
00029
00030 //model and parts
00031 #include "PluginConnectorDummyImpl1.h"
00032 #include "SimulationReporterDefaultImpl1.h"
00033 #include "ModelCheckerDefaultImpl1.h"
00034 #include "ModelPersistenceDefaultImpl1.h"
00035 #include "parserBisonFlex/ParserDefaultImpl2.h"
00036 #include "StatisticsDefaultImpl1.h"
00037 #include "CollectorDefaultImpl1.h"
00038 #include "SamplerDefaultImpl1.h"
00039
00040 namespace GenesysKernel {
00041
00042     template <typename T>
00043     struct Traits {
00044         };
00045
00046     /*
00047      * Model and Model Parts
00048     */
00049
00050     template <> struct Traits<Model> {
00051         static const bool debugged = true;
00052         static const Util::TraceLevel traceLevel = Util::TraceLevel::modelSimulationEvent;
00053     };
00054
00055     template <> struct Traits<ModelPersistence_if> {
00056         typedef ModelPersistenceDefaultImpl1 Implementation;
00057     };
00058
00059     template <> struct Traits<SimulationReporter_if> {
00060         typedef SimulationReporterDefaultImpl1 Implementation;
00061     };
00062
00063     template <> struct Traits<ModelChecker_if> {
00064         typedef ModelCheckerDefaultImpl1 Implementation;
00065     };
00066
00067     template <> struct Traits<ModelComponent> {
00068         typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation;
00069         typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation;
00070     };
00071
00072     template <> struct Traits<Sampler_if> {
00073         typedef SamplerDefaultImpl1 Implementation;
00074         typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters;
00075     };
00076
00077     template <> struct Traits<Parser_if> {
00078         typedef ParserDefaultImpl2 Implementation;
00079     };
00080
00081     template <> struct Traits<PluginConnector_if> {
00082         typedef PluginConnectorDummyImpl1 Implementation;
00083     };
00084 }
00085
00086 #endif /* TRAITS_H */
00087

```

9.514 Unstore.cpp File Reference

```
#include "Unstore.h"
#include "Model.h"
```

Include dependency graph for Unstore.cpp:



9.515 Unstore.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Unstore.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #include "Unstore.h"
00015 #include "Model.h"
00016
00017 Unstore::Unstore(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Unstore>(),
00018   name) {
00019
00020   std::string Unstore::show() {
00021     return ModelComponent::show() + "";
00022   }
00023
00024 ModelComponent* Unstore::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00025   Unstore* newComponent = new Unstore(model);
00026   try {
00027     newComponent->_loadInstance(fields);
00028   } catch (const std::exception& e) {
00029   }
00030   return newComponent;
00031 }
00032
00033
00034 void Unstore::_execute(Entity* entity) {
00035   _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00036   this->_parentModel->sendEntityToComponent(entity, this->nextComponents()->frontConnection(), 0.0);
00037 }
00038
00039 bool Unstore::_loadInstance(std::map<std::string, std::string>* fields) {
00040   bool res = ModelComponent::_loadInstance(fields);
00041   if (res) {
00042     //...
00043   }
00044   return res;
00045 }
00046
00047 void Unstore::_initBetweenReplications() {
00048 }
00049
00050 std::map<std::string, std::string>* Unstore::_saveInstance() {
00051   std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00052   //...
00053   return fields;
00054 }
00055
00056 bool Unstore::_check(std::string* errorMessage) {
00057   bool resultAll = true;
00058   //...
00059   return resultAll;

```

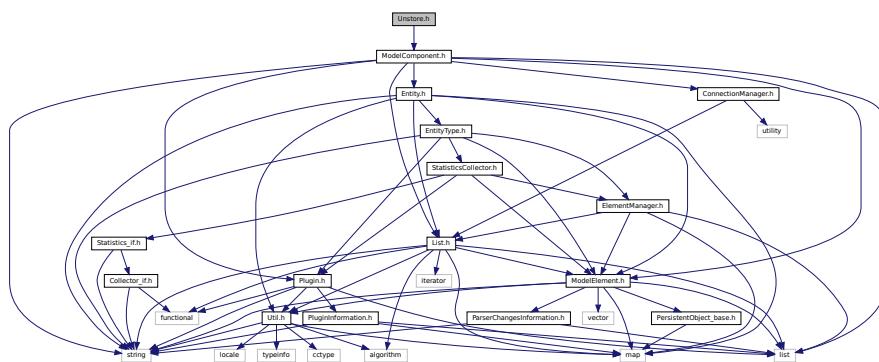
```

00060 }
00061
00062 PluginInformation* Unstore::GetPluginInformation() {
00063     PluginInformation* info = new PluginInformation(Util::TypeOf<Unstore>(), &Unstore::LoadInstance);
00064     // ...
00065     return info;
00066 }
00067
00068

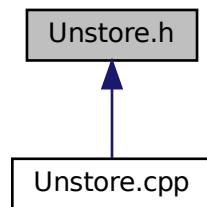
```

9.516 Unstore.h File Reference

#include "ModelComponent.h"
Include dependency graph for Unstore.h:



This graph shows which files directly or indirectly include this file:



Classes

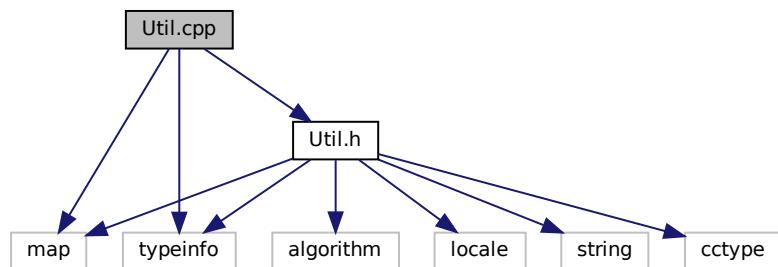
- class [Unstore](#)

9.517 Unstore.h

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 /*  
00008 * File: Unstore.h  
00009 * Author: rlcancian  
00010 *  
00011 * Created on 11 de Setembro de 2019, 13:08  
00012 */  
00013  
00014 #ifndef UNSTORE_H  
00015 #define UNSTORE_H  
00016  
00017 #include "ModelComponent.h"  
00018  
00019 class Unstore : public ModelComponent {  
00020 public: // constructors  
00021     Unstore(Model* model, std::string name = "");  
00022     virtual ~Unstore() = default;  
00023 public: // virtual  
00024     virtual std::string show();  
00025 public: // static  
00026     static PluginInformation* GetPluginInformation();  
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);  
00028 protected: // virtual  
00029     virtual void _execute(Entity* entity);  
00030     virtual void _initBetweenReplications();  
00031     virtual bool _loadInstance(std::map<std::string, std::string>* fields);  
00032     virtual std::map<std::string, std::string>* _saveInstance();  
00033     virtual bool _check(std::string* errorMessage);  
00034 private: // methods  
00035 private: // attributes 1:1  
00036 private: // attributes 1:n  
00037 };  
00038  
00039  
00040  
00041  
00042  
00043  
00044  
00045  
00046  
00047  
00048  
00049  
00050  
00051  
00052  
00053  
00054  
00055  
00056  
00057  
00058  
00059  
00060  
00061  
00062  
00063  
00064  
00065  
00066 #endif /* UNSTORE_H */  
00067
```

9.518 Util.cpp File Reference

```
#include "Util.h"
#include <typeinfo>
#include <map>
Include dependency graph for Util.cpp:
```



9.519 Util.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.
```

```
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Util.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 13:02
00012 */
00013
00014 #include "Util.h"
00015 #include <typeinfo>
00016 #include <map>
00017
00018 //using namespace GenesysKernel;
00019
00020 Util::identification Util::_S_lastId = 0;
00021 unsigned int Util::_S_indentation;
00022 std::map<std::string, Util::identification> Util::_S_lastIdOfType = std::map<std::string,
    Util::identification>();
00023 std::map<std::string, std::string> Util::_S_TypeOf = std::map<std::string, std::string>();
00024
00025 /*
00026 std::string Util::ToStrTimeUnit(TimeUnit tu) {
00027     switch (tu) {
00028         case TimeUnit::picosecond: return "picosecond";
00029         case TimeUnit::nanosecond: return "nanosecond";
00030         case TimeUnit::microsecond: return "microsecond";
00031         case TimeUnit::millisecond: return "millisecond";
00032         case TimeUnit::second: return "second";
00033         case TimeUnit::minute: return "minute";
00034         case TimeUnit::hour: return "hour";
00035         case TimeUnit::day: return "day";
00036         case TimeUnit::week: return "week";
00037         default: return "";
00038     }
00039 }
00040 */
00041
00042 void Util::IncIndent() {
00043     Util::_S_indentation++;
00044 }
00045
00046 void Util::SetIndent(const unsigned short indent) {
00047     Util::_S_indentation = indent;
00048 }
00049
00050 void Util::DecIndent() {
00051     Util::_S_indentation--;
00052 }
00053
00054 void Util::SepKeyVal(std::string str, std::string *key, std::string *value) {
00055     // \todo: Check pointers when splitting string. There is an error
00056     //char *c;
00057     bool settingKey = true;
00058     //key = new std::string();
00059     //value = new std::string();
00060     key->clear();
00061     value->clear();
00062     for (std::string::iterator it = str.begin(); it != str.end(); it++) {
00063         if (settingKey) {
00064             if ((*it) != '=') {
00065                 key->append(new char((*it)));
00066             } else {
00067                 settingKey = false;
00068             }
00069         } else {
00070             value->append(new char((*it)));
00071         }
00072     }
00073 }
00074
00075 std::string Util::Indent() {
00076     std::string spaces = "";
00077     for (unsigned int i = 0; i < Util::_S_indentation; i++) {
00078         spaces += "| ";
00079     }
00080     return spaces;
00081 }
00082
00083 std::string Util::SetW(std::string text, unsigned short width) {
00084     std::string spaces(width, ' ');
00085     std::string result = text + spaces;
00086     return result.substr(0, width);
00087 }
00088
```

```

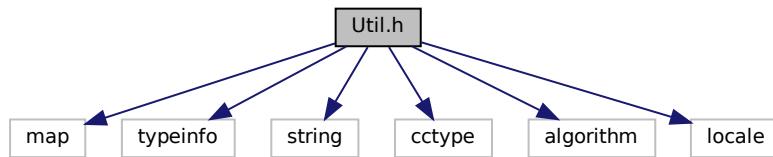
00089 std::string Util::StrTimeUnit(Util::TimeUnit timeUnit) {
00090     switch (static_cast<int> (timeUnit)) {
00091         case 1: return "picosecond";
00092         case 2: return "nanosecond";
00093         case 3: return "microsecond";
00094         case 4: return "millisecond";
00095         case 5: return "second";
00096         case 6: return "minute";
00097         case 7: return "hour";
00098         case 8: return "day";
00099         case 9: return "week";
00100    }
00101    return "";
00102 }
00103
00104 Util::identification Util::GenerateNewId() {
00105     Util::_S_lastId++;
00106     return Util::_S_lastId;
00107 }
00108
00109 Util::identification Util::GenerateNewIdOfType(std::string objtype) {
00110     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00111     if (it == Util::_S_lastIdOfType.end()) {
00112         // a new one. create the pair
00113         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00114         it = Util::_S_lastIdOfType.find(objtype);
00115     }
00116     Util::identification next = (*it).second;
00117     next++;
00118     (*it).second = next;
00119     return (*it).second;
00120 }
00121
00122 Util::identification Util::GetLastIdOfType(std::string objtype) {
00123     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00124     if (it == Util::_S_lastIdOfType.end()) {
00125         // a new one. create the pair
00126         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00127         it = Util::_S_lastIdOfType.find(objtype);
00128     }
00129     //Util::identification next = (*it).second;
00130     //next++;
00131     //(*it).second = next;
00132     return (*it).second;
00133 }
00134
00135
00136 void Util::ResetIdOfType(std::string objtype) {
00137     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00138     if (it == Util::_S_lastIdOfType.end()) {
00139         // a new one. create the pair
00140         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00141         it = Util::_S_lastIdOfType.find(objtype);
00142     }
00143     (*it).second = 0;
00144 }
00145
00146 double Util::TimeUnitConvert(Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2) {
00147     double scaleValues[] = {1.0, 1000.0, 1000.0, 1000.0, 1000.0, 60.0, 60.0, 24.0, 7.0};
00148     // TU_picosecond = 1, TU_microsecond = 3, TU_millisecond = 4, TU_second = 5, TU_minute = 6, TU_hour
00149     = 7, TU_day = 8, TU_week = 9
00150     double res = 1.0;
00151     int intTimeUnit1, intTimeUnit2;
00152     intTimeUnit1 = static_cast<int> (timeUnit1);
00153     intTimeUnit2 = static_cast<int> (timeUnit2);
00154     if (intTimeUnit1 <= intTimeUnit2) {
00155         for (int i = intTimeUnit1 + 1; i <= intTimeUnit2; i++) {
00156             res /= scaleValues[i];
00157         }
00158     } else {
00159         for (int i = intTimeUnit2 + 1; i <= intTimeUnit1; i++) {
00160             res *= scaleValues[i];
00161         }
00162     }
00163 }

```

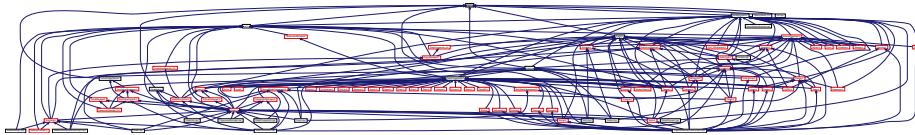
9.520 Util.h File Reference

```
#include <map>
#include <typeinfo>
```

```
#include <string>
#include <cctype>
#include <algorithm>
#include <locale>
Include dependency graph for Util.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Util](#)

Functions

- static std::string [map2str](#) (std::map< std::string, std::string > *mapss)
- static void [ltrim](#) (std::string &s)
- static void [rtrim](#) (std::string &s)
- static void [trim](#) (std::string &s)
- static void [trimwithin](#) (std::string &str)
- static std::string [getFileName](#) (const std::string &s)
- static std::string [loadField](#) (std::map< std::string, std::string > *fields, std::string fieldName, std::string defaultValue)

9.520.1 Function Documentation

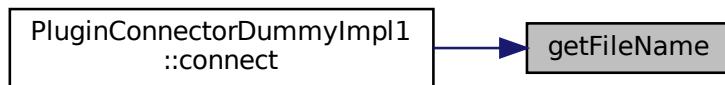
9.520.1.1 getFileName() static std::string getFileName (const std::string & s) [inline], [static]

Definition at line 67 of file Util.h.

```
00067 {  
00068     char sep = '/';  
00069     size_t i = s.rfind(sep, s.length());  
00070     if (i != std::string::npos) {  
00071         return (s.substr(i + 1, s.length() - i));  
00072     }  
00073     return s;  
00074 }
```

Referenced by [PluginConnectorDummyImpl1::connect\(\)](#).

Here is the caller graph for this function:



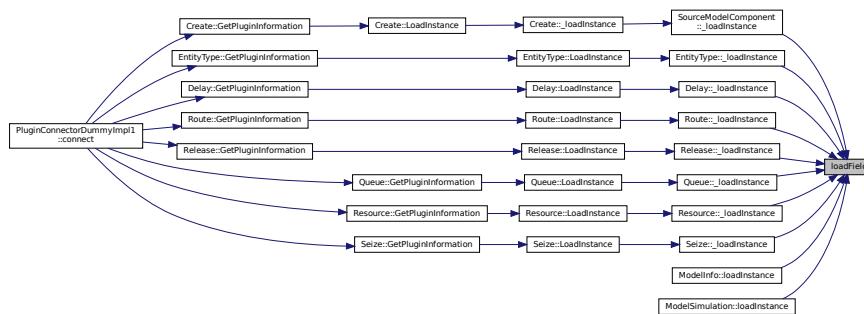
9.520.1.2 loadField() static std::string loadField (std::map< std::string, std::string > * fields, std::string fieldName, std::string defaultValue) [inline], [static]

Definition at line 76 of file Util.h.

```
00076 {  
00077     return fields->find(fieldName) != fields->end() ? ((*fields->find(fieldName))).second :  
defaultValue;  
00078 }
```

Referenced by [SourceModelComponent::_loadInstance\(\)](#), [EntityType::_loadInstance\(\)](#), [Delay::_loadInstance\(\)](#), [Route::_loadInstance\(\)](#), [Release::_loadInstance\(\)](#), [Queue::_loadInstance\(\)](#), [Resource::_loadInstance\(\)](#), [Seize::_loadInstance\(\)](#), [ModellInfo::loadInstance\(\)](#), and [ModelSimulation::loadInstance\(\)](#).

Here is the caller graph for this function:



9.520.1.3 ltrim() static void ltrim (
 std::string & s) [inline], [static]

Definition at line 37 of file Util.h.

```
00037     {
00038         s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](int ch) {
00039             return !std::isspace(ch);
00040         }));
00041 }
```

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

Here is the caller graph for this function:



9.520.1.4 map2str() static std::string map2str (
 std::map< std::string, std::string > * mapss) [inline], [static]

Definition at line 28 of file Util.h.

```
00028
00029     std::string res = "";
00030     for (std::map<std::string, std::string>::iterator it = mapss->begin(); it != mapss->end(); it++) {
00031         res += (*it).first + "=" + (*it).second + " ";
00032     }
00033     res = res.substr(0, res.length() - 1);
00034     return res;
00035 }
```

9.520.1.5 rtrim() static void rtrim (
 std::string & s) [inline], [static]

Definition at line 45 of file Util.h.

```
00045
00046     {
00047         s.erase(std::find_if(s.rbegin(), s.rend(), [](int ch) {
00048             return !std::isspace(ch);
00049         }).base(), s.end());
00050 }
```

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

Here is the caller graph for this function:



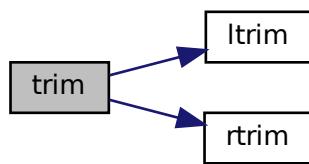
9.520.1.6 trim() static void trim (
 std::string & s) [inline], [static]

Definition at line 53 of file [Util.h](#).

```
00053
00054     ltrim(s);
00055     rtrim(s);
00056 }
```

References [ltrim\(\)](#), and [rtrim\(\)](#).

Here is the call graph for this function:



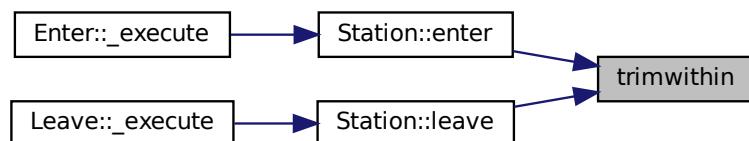
9.520.1.7 trimwithin() static void trimwithin (
 std::string & str) [inline], [static]

Definition at line 60 of file [Util.h](#).

```
00060
00061     //ltrim(s);
00062     //rtrim(s);
00063     //s.erase(std::remove_if(s.begin(), s.end(), std::isspace), s.end());
00064     str.erase(remove(str.begin(), str.end(), ','), str.end());
00065 }
```

Referenced by [Station::enter\(\)](#), and [Station::leave\(\)](#).

Here is the caller graph for this function:



9.521 Util.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Util.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 13:02
00012 */
00013
00014 #ifndef UTIL_H
00015 #define UTIL_H
00016
00017 #include <map>
00018 #include <typeinfo>
00019 #include <string>
00020 #include <cctype>
00021 #include <algorithm>
00022 #include <cctype>
00023 #include <locale>
00024
00025 //namespace GenesysKernel {
00026 // trim from start (in place)
00027
00028 static inline std::string map2str(std::map<std::string, std::string>* mapss) {
00029     std::string res = "";
00030     for (std::map<std::string, std::string>::iterator it = mapss->begin(); it != mapss->end(); it++) {
00031         res += (*it).first + "=" + (*it).second + " ";
00032     }
00033     res = res.substr(0, res.length() - 1);
00034     return res;
00035 }
00036
00037 static inline void ltrim(std::string &s) {
00038     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](int ch) {
00039         return !std::isspace(ch);
00040     }));
00041 }
00042
00043 // trim from end (in place)
00044
00045 static inline void rtrim(std::string &s) {
00046     s.erase(std::find_if(s.rbegin(), s.rend(), [](int ch) {
00047         return !std::isspace(ch);
00048     }).base(), s.end());
00049 }
00050
00051 // trim from both ends (in place)
00052
00053 static inline void trim(std::string &s) {
00054     ltrim(s);
00055     rtrim(s);
00056 }
00057
00058 // trim all spaces within the string (in place) -- used to transform general names into valid literals
00059
00060 static inline void trimwithin(std::string &str) {
00061     //ltrim(s);
00062     //rtrim(s);
00063     //s.erase(std::remove_if(s.begin(), s.end(), std::isspace), s.end());
00064     str.erase(remove(str.begin(), str.end(), ' '), str.end());
00065 }
00066
00067 static inline std::string getFileName(const std::string& s) {
00068     char sep = '/';
00069     size_t i = s.rfind(sep, s.length());
00070     if (i != std::string::npos) {
00071         return (s.substr(i + 1, s.length() - i));
00072     }
00073     return s;
00074 }
00075
00076 static inline std::string loadField(std::map<std::string, std::string>* fields, std::string fieldName,
00077                                     std::string defaultValue) {
00078     if (fields->find(fieldName) != fields->end() ? ((*fields->find(fieldName)).second) :
00079         defaultValue;
00080 }
00081
00082 class Util {
00083 public:
00084     typedef unsigned long identification;
00085     typedef unsigned int rank;

```

```

00084
00085     enum class TimeUnit : int {
00086         picosecond = 1,
00087         nanosecond = 2,
00088         microsecond = 3,
00089         millisecond = 4,
00090         second = 5,
00091         minute = 6,
00092         hour = 7,
00093         day = 8,
00094         week = 9
00095     };
00096     //static std::stringToStrTimeUnit(TimeUnit tu);
00097
00098     enum class TraceLevel : int {
00099         noTraces = 0,
00100         errorFatal = 1,
00101         errorRecover = 2,
00102         warning = 3,
00103         report = 4,
00104
00105         simulatorResult = 10,
00106         toolResult = 11,
00107         modelResult = 12,
00108         componentResult = 13,
00109         elementResult = 14,
00110
00111         modelSimulationEvent = 15,
00112
00113         componentArrival = 20,
00114         simulatorInternal = 21,
00115         toolInternal = 22,
00116         modelSimulationInternal = 23,
00117         modelInternal = 24,
00118         componentInternal = 25,
00119         elementInternal = 26,
00120
00121         simulatorDetailed = 30,
00122         toolDetailed = 31,
00123         modelSimulationDetailed = 32,
00124         modelDetailed = 33,
00125         componentDetailed = 34,
00126         elementDetailed = 35,
00127
00128         everythingMostDetailed = 99
00129     };
00130 private:
00131     static Util::identification _S_lastId;
00132     static std::map<std::string, Util::identification> _S_lastIdOfType;
00133     static std::map<std::string, std::string> _S_TypeOf;
00134
00135 public: // indentation and string
00136     static unsigned int _S_indentation; // \todo: IS PRIVATE. ITS HERE JUST TO INCLUDE IT AS A WATCH
00137     static void SetIndent(const unsigned short indent);
00138     static void IncIndent();
00139     static void DecIndent();
00140     static void SepKeyVal(std::string str, std::string *key, std::string *value);
00141     static std::string Indent();
00142     static std::string SetW(std::string text, unsigned short width);
00143     static std::string StrTimeUnit(Util::TimeUnit timeUnit);
00144 public: // identification
00145     static Util::identification GenerateNewId();
00146     static Util::identification GenerateNewIdOfType(std::string objtype);
00147     static Util::identification GetLastIdOfType(std::string objtype);
00148     static void ResetIdOfType(std::string objtype);
00149
00150 public: // simulation support
00151     static double TimeUnitConvert(Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2);
00152
00153 public: // template implementations
00154
00155     template<class T> static std::string TypeOf() {
00156         std::string name = typeid (T).name();
00157         std::map<std::string, std::string>::iterator it = _S_TypeOf.find(name);
00158         if (it != _S_TypeOf.end()) {
00159             return (*it).second;
00160         } else {
00161             std::string newname(name);
00162             while (std::isdigit(newname[0])) {
00163                 newname.erase(0, 1);
00164             }
00165             _S_TypeOf.insert(std::pair<std::string, std::string>(name, newname));
00166             return newname;
00167         }
00168     }
00169
00170     template<class T> static Util::identification GenerateNewIdOfType() {
00171
00172 }
```

```

00177     std::string objtype = Util::TypeOf<T>();
00178     std::map<std::string, Util::identification>::iterator it =
00179         Util::_S_lastIdOfType.find(objtype);
00180     if (it == Util::_S_lastIdOfType.end()) {
00181         // a new one. create the pair
00182         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00183         it = Util::_S_lastIdOfType.find(objtype);
00184     }
00185     Util::identification next = (*it).second;
00186     next++;
00187     (*it).second = next;
00188     return (*it).second;
00189 }
00190 private:
00191     Util();
00192     virtual ~Util() = default;
00193 };
00194 //namespace\\\
00195 #endif /* UTIL_H */
00196

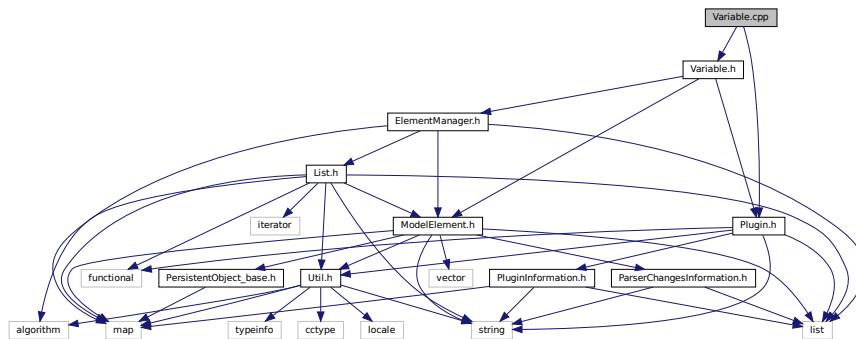
```

9.522 Variable.cpp File Reference

#include "Variable.h"

#include "Plugin.h"

Include dependency graph for Variable.cpp:



9.523 Variable.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Variable.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 4 de Setembro de 2018, 18:28
00012 */
00013
00014 #include "Variable.h"
00015 #include "Plugin.h"
00016
00017 Variable::Variable(Model* model, std::string name) : ModelElement(model, Util::TypeOf<Variable>(),
00018     name) {
00019     _name = name;
00020 }
00021 std::string Variable::show() {
00022     return ModelElement::show(); // \todo: include values
00023 }
00024
00025 PluginInformation* Variable::GetPluginInformation() {

```

```

00026     PluginInformation* info = new PluginInformation(Util::TypeOf<Variable>(),
00027         &Variable::LoadInstance);
00028     return info;
00029 }
00030 double Variable::value() {
00031     return value("");
00032 }
00033
00034 double Variable::value(std::string index) {
00035     std::map<std::string, double>::iterator it = _values->find(index);
00036     if (it == _values->end()) {
00037         return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00038     } else {
00039         return it->second;
00040     }
00041 }
00042
00043 void Variable::setValue(double value) {
00044     setValue("", value);
00045 }
00046
00047 void Variable::setValue(std::string index, double value) {
00048     std::map<std::string, double>::iterator it = _values->find(index);
00049     if (it == _values->end()) {
00050         // index does not exist. Create it.
00051         _values->insert({index, value}); //((std::pair<std::string, double>(index, value)));
00052     } else {
00053         it->second = value;
00054     }
00055 }
00056
00057 double Variable::initialValue() {
00058     return initialValue("");
00059 }
00060
00061 void Variable::setInitialValue(double value) {
00062     setValue("", value);
00063 }
00064
00065 double Variable::initialValue(std::string index) {
00066     std::map<std::string, double>::iterator it = _initialValues->find(index);
00067     if (it == _initialValues->end()) {
00068         return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00069     } else {
00070         return it->second;
00071     }
00072 }
00073
00074 void Variable::setInitialValue(std::string index, double value) {
00075     std::map<std::string, double>::iterator it = _initialValues->find(index);
00076     if (it == _initialValues->end()) {
00077         // index does not exist. Create it.
00078         _initialValues->insert(std::pair<std::string, double>(index, value));
00079     } else {
00080         it->second = value;
00081     }
00082 }
00083
00084 List<unsigned int>* Variable::dimensionSizes() const {
00085     return _dimensionSizes;
00086 }
00087
00088 ModelElement* Variable::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00089     Variable* newElement = new Variable(model);
00090     try {
00091         newElement->_loadInstance(fields);
00092     } catch (const std::exception& e) {
00093     }
00094 }
00095     return newElement;
00096 }
00097
00098 bool Variable::_loadInstance(std::map<std::string, std::string>* fields) {
00099     bool res = ModelElement::_loadInstance(fields);
00100     if (res) {
00101         unsigned int nv = std::stoi((*(fields->find("numValues"))).second);
00102         std::string pos;
00103         double value;
00104         for (unsigned int i = 0; i < nv; i++) {
00105             pos = (*(fields->find("pos" + std::to_string(i)))).second;
00106             value = std::stod((*(fields->find("value" + std::to_string(i)))).second);
00107             this->_initialValues->emplace(pos, value);
00108         }
00109     }
00110 }
00111
00112     return res;
00113 }
```

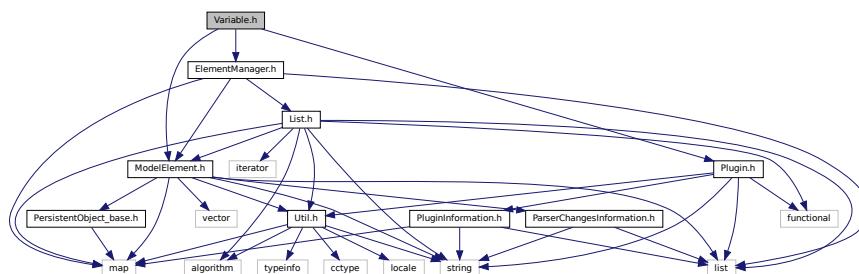
```

00114
00115 std::map<std::string, std::string>* Variable::_saveInstance() {
00116     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00117     //Util::TypeOf<Variable>());
00118     fields->emplace("numValues", std::to_string(this->_initialValues->size()));
00119     unsigned int i = 0;
00120     for (std::map<std::string, double>::iterator it = this->_initialValues->begin(); it != _initialValues->end(); it++) {
00121         fields->emplace("pos" + std::to_string(i), (*it).first);
00122         fields->emplace("value" + std::to_string(i), std::to_string((*it).second));
00123         i++;
00124     }
00125 }
00126 return fields;
00127 }
00128
00129 bool Variable::_check(std::string* errorMessage) {
00130     return true;
00131 }
00132
00133 void Variable::_initBetweenReplications() {
00134     this->_values->clear();
00135     this->_values = this->_initialValues;
00136 }

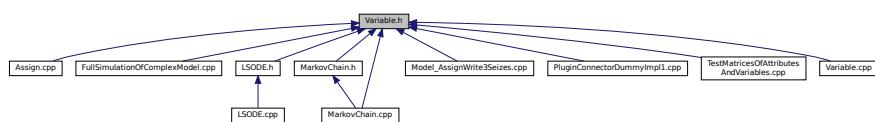
```

9.524 Variable.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
Include dependency graph for Variable.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Variable](#)

9.525 Variable.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Variable.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 4 de Setembro de 2018, 18:28
00012 */
00013
00014 #ifndef VARIABLE_H
00015 #define VARIABLE_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020
00021 class Variable : public ModelElement {
00022 public:
00023     Variable(Model* model, std::string name = "");
00024     virtual ~Variable() = default;
00025 public:
00026     virtual std::string show();
00027 public: //static
00028     static PluginInformation* GetPluginInformation();
00029     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00030 public:
00031     double value();
00032     void setValue(double value);
00033     double value(std::string index);
00034     void setValue(std::string index, double value);
00035     double initialValue();
00036     void setValue(double value);
00037     double initialValue(std::string index);
00038     void setValue(std::string index, double value);
00039     List<unsigned int>* dimensionSizes() const;
00040
00041 protected:
00042     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00043     virtual std::map<std::string, std::string>* _saveInstance();
00044     virtual bool _check(std::string* errorMessage);
00045     virtual void _initBetweenReplications();
00046
00047 private:
00048     List<unsigned int>* _dimensionSizes = new List<unsigned int>();
00049     std::map<std::string, double>* _values = new std::map<std::string, double>();
00050     std::map<std::string, double>* _initialValues = new std::map<std::string, double>();
00051 };
00052 #endif /* VARIABLE_H */
00053

```

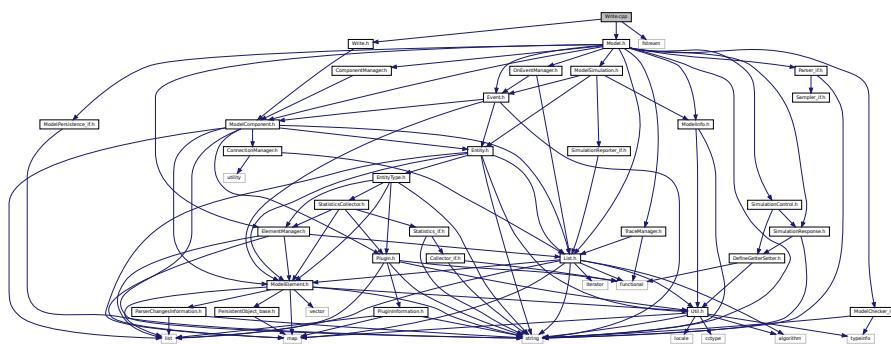
9.526 Write.cpp File Reference

```

#include "Write.h"
#include "Model.h"
#include <fstream>

```

Include dependency graph for Write.cpp:



9.527 Write.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Write.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:06
00012 */
00013
00014 #include "Write.h"
00015 #include "Model.h"
00016
00017 #include <fstream>
00018
00019 Write::Write(Model* model, std::string name) : ModelComponent(model, Util::TypeOf<Write>(), name) {
00020 }
00021
00022 std::string Write::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Write::LoadInstance(Model* model, std::map<std::string, std::string>* fields) {
00027     Write* newComponent = new Write(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032     return newComponent;
00033 }
00034 }
00035
00036 List<WriteElement*>* Write::writeElements() const {
00037     return _writeElements;
00038 }
00039
00040 void Write::setFilename(std::string _filename) {
00041     this->_filename = _filename;
00042 }
00043
00044 std::string Write::filename() const {
00045     return _filename;
00046 }
00047
00048 void Write::setWriteToType(WriteToType _writeToType) {
00049     this->_writeToType = _writeToType;
00050 }
00051
00052 Write::WriteToType Write::writeToType() const {
00053     return _writeToType;
00054 }
00055
00056 void Write::_execute(Entity* entity) {
00057     WriteElement* msgElem;
00058     std::list<WriteElement*>* msgs = this->_writeElements->list();
00059     std::string message = "";
00060     for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
```

```

00061     msgElem = (*it);
00062     if (msgElem->isExpression) {
00063         message += std::to_string(_parentModel->parseExpression(msgElem->text));
00064     } else {
00065         message += msgElem->text;
00066     }
00067     if (msgElem->newline) {
00068         if (this->_writeToType == Write::WriteToType::SCREEN) { //\todo: Write To FILE not
00069             implemented
00070             _parentModel->getTracer()->trace(Util::TraceLevel::report, message);
00071         } else if (this->_writeToType == Write::WriteToType::FILE) {
00072             // open file
00073             std::ofstream savefile;
00074             savefile.open(_filename, std::ofstream::app);
00075             savefile << message << std::endl;
00076             savefile.close();
00077             message = "";
00078         }
00079     }
00080 }
00081     this->_parentModel->sendEntityToComponent(entity, this->getNextComponents()->getFrontConnection(),
00082     0.0);
00083 }
00084 void Write::_initBetweenReplications() {
00085     try {
00086         std::ofstream savefile;
00087         savefile.open(_filename, std::ofstream::app);
00088         savefile << "# Replication number " <<
00089         _parentModel->getSimulation()->getCurrentReplicationNumber() << "/" <<
00090         _parentModel->getSimulation()->getNumberOfReplications() << std::endl;
00091         savefile.close();
00092     } catch (...) {
00093 }
00094 }
00095 bool Write::_loadInstance(std::map<std::string, std::string>* fields) {
00096     bool res = ModelComponent::_loadInstance(fields);
00097     if (res) {
00098         this->_writeToType = static_cast<WriteToType>
00099             (std::stoi((*fields->find("writeToType")).second));
00100         unsigned short writesSize = std::stoi((*fields->find("writesSize")).second);
00101         for (unsigned short i = 0; i < writesSize; i++) {
00102             std::string text = (*fields->find(text)).second;
00103             bool isExpression = static_cast<bool> (std::stoi((*fields->find("isExpression")).second));
00104             bool newline = static_cast<bool> (std::stoi((*fields->find("newline")).second));
00105             this->_writeElements->insert(new WriteElement(text, isExpression, newline));
00106         }
00107     }
00108     return res;
00109 }
00110 std::map<std::string, std::string>* Write::_saveInstance() {
00111     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance();
00112     fields->emplace("writeToType", std::to_string(static_cast<int> (_writeToType)));
00113     fields->emplace("writesSize", std::to_string(_writeElements->size()));
00114     unsigned short i = 0;
00115     WriteElement* writeElem;
00116     for (std::list<WriteElement*>::iterator it = _writeElements->list()->begin(); it !=
00117         _writeElements->list()->end(); it++) {
00118         writeElem = (*it);
00119         fields->emplace("isExpression" + std::to_string(i), std::to_string(writeElem->isExpression));
00120         fields->emplace("newline" + std::to_string(i), std::to_string(writeElem->newline));
00121         fields->emplace("text" + std::to_string(i), writeElem->text);
00122         i++;
00123     }
00124     //this->_writeElements
00125     return fields;
00126 }
00127
00128 bool Write::_check(std::string* errorMessage) {
00129     bool resultAll = true;
00130     WriteElement* msgElem;
00131     unsigned short i = 0;
00132     std::list<WriteElement*> msgs = this->_writeElements->list();
00133     for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
00134         msgElem = (*it);
00135         i++;
00136         if (msgElem->isExpression) {
00137             resultAll &= _parentModel->checkExpression(msgElem->text, "writeExpression" +
00138                 std::to_string(i), errorMessage);
00139         }
00140     }
00141     // when cheking the model (before simulating it), remove the file if exists

```

```

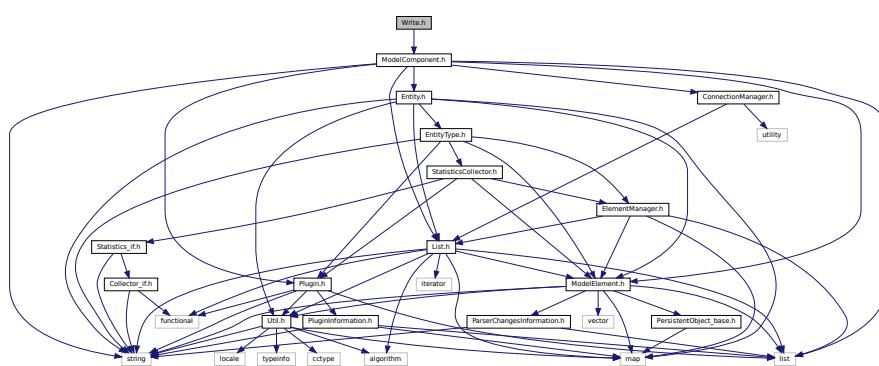
00141     std::remove(_filename.c_str());
00142
00143     return resultAll;
00144 }
00145
00146 PluginInformation* Write::GetPluginInformation() {
00147     PluginInformation* info = new PluginInformation(Util::TypeOf<Write>(), &Write::LoadInstance);
00148     // ...
00149     return info;
00150 }
00151
00152

```

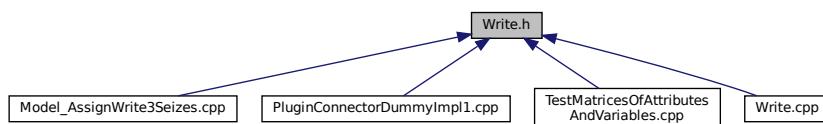
9.528 Write.h File Reference

#include "ModelComponent.h"

Include dependency graph for Write.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [WriteElement](#)
- class [Write](#)

9.529 Write.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Write.h
00009  * Author: rlcancian

```

```

00010  *
00011  * Created on 11 de Setembro de 2019, 13:06
00012  */
00013
00014 #ifndef WRITE_H
00015 #define WRITE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class WriteElement {
00020 public:
00021
00022     WriteElement(std::string text, bool isExpression = false, bool newline = false) {
00023         this->text = text;
00024         this->isExpression = isExpression;
00025         this->newline = newline;
00026     }
00027     std::string text;
00028     bool isExpression;
00029     bool newline;
00030 };
00031
00035 class Write : public ModelComponent {
00036 public:
00037
00038     enum class WriteToType : int {
00039         SCREEN = 1, FILE = 2
00040     };
00041
00042
00043
00044 public: // constructors
00045     Write(Model* model, std::string name = "");
00046     virtual ~Write() = default;
00047 public: // virtual
00048     virtual std::string show();
00049 public: // static
00050     static PluginInformation* GetPluginInformation();
00051     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>* fields);
00052 public:
00053     List<WriteElement*>* writeElements() const;
00054     void setFilename(std::string _filename);
00055     std::string filename() const;
00056     void setWriteToType(WriteToType _writeToType);
00057     Write::WriteToType writeToType() const;
00058
00059 protected: // virtual
00060     virtual void _execute(Entity* entity);
00061     virtual void _initBetweenReplications();
00062     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00063     virtual std::map<std::string, std::string>* _saveInstance();
00064     virtual bool _check(std::string* errorMessage);
00065 private: // methods
00066     //std::string _buildText();
00067 private: // attributes 1:1
00068     WriteToType _writeToType = Write::WriteToType::SCREEN;
00069     std::string _filename = "";
00070 private: // attributes 1:n
00071     List<WriteElement*>* _writeElements = new List<WriteElement*>();
00072 };
00073
00074
00075 #endif /* WRITE_H */
00076

```