GenESyS - GENeric and Expansible SYstem Simulator

Generated by Doxygen 1.9.4

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 7

2.1 Class List . . . 7

3 Class Documentation 14

3.1 Access Class Reference . . . 14

3.1.1 Detailed Description . . . 16

3.1.2 of Cells Number of contiguous conveyor cells the entity requires for . . . 16

3.1.3 Constructor & Destructor Documentation . . . 16

3.1.4 Member Function Documentation . . . 17

3.2 Assign Class Reference . . . 19

3.2.1 Detailed Description . . . 21

3.2.2 Constructor & Destructor Documentation . . . 21

3.2.3 Member Function Documentation . . . 22

3.3 Assignment Class Reference . . . 25

3.3.1 Detailed Description . . . 26

3.3.2 Constructor & Destructor Documentation . . . 26

3.3.3 Member Function Documentation . . . 27

3.4 Attribute Class Reference . . . 29

3.4.1 Detailed Description . . . 30

3.4.2 Constructor & Destructor Documentation . . . 30

3.4.3 Member Function Documentation . . . 31

3.5 BaseGenesysTerminalApplication Class Reference . . . 33

3.5.1 Constructor & Destructor Documentation . . . 35

3.5.2 Member Function Documentation . . . 35

3.6 Batch Class Reference . . . 43

3.6.1 Detailed Description . . . 44

3.6.2 Member Enumeration Documentation . . . 45

3.6.3 Constructor & Destructor Documentation . . . 45

3.6.4 Member Function Documentation . . . 45

3.7 SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference . . . 52

3.7.1 Member Data Documentation . . . 52

3.8 Buffer Class Reference . . . 52

3.8.1 Detailed Description . . . 54

3.8.2 Member Enumeration Documentation . . . 54

3.8.3 Constructor & Destructor Documentation . . . 55

3.8.4 Member Function Documentation . . . 55

3.9 CellularAutomata Class Reference . . . 60

3.9.1 Constructor & Destructor Documentation . . . 61

3.9.2 Member Function Documentation . . . 61

3.10 Clone Class Reference . . . 64

3.10.1 Detailed Description . . . 66
3.10.2 Constructor & Destructor Documentation . . . 66
3.10.3 Member Function Documentation . . . 66
3.11 Collector_if Class Reference . . . 71
3.11.1 Detailed Description . . . 72
3.11.2 Member Function Documentation . . . 72
3.12 CollectorDatafile_if Class Reference . . . 73
3.12.1 Detailed Description . . . 74
3.12.2 Member Function Documentation . . . 74
3.13 CollectorDatafileDefaultImpl1 Class Reference . . . 75
3.13.1 Constructor & Destructor Documentation . . . 76
3.13.2 Member Function Documentation . . . 76
3.14 CollectorDefaultImpl1 Class Reference . . . 78
3.14.1 Constructor & Destructor Documentation . . . 79
3.14.2 Member Function Documentation . . . 79
3.15 CppCompiler::CompilationResult Class Reference . . . 80
3.15.1 Constructor & Destructor Documentation . . . 80
3.15.2 Member Data Documentation . . . 80
3.16 CompilerThread Class Reference . . . 81
3.16.1 Member Function Documentation . . . 81
3.17 ComponentManager Class Reference . . . 81
3.17.1 Detailed Description . . . 82
3.17.2 Constructor & Destructor Documentation . . . 82
3.17.3 Member Function Documentation . . . 82
3.18 Connection Struct Reference . . . 85
3.18.1 Detailed Description . . . 85
3.18.2 Member Data Documentation . . . 85
3.19 ConnectionChannel Struct Reference . . . 86
3.19.1 Member Data Documentation . . . 86
3.20 ConnectionManager Class Reference . . . 86
3.20.1 Detailed Description . . . 87
3.20.2 Constructor & Destructor Documentation . . . 87
3.20.3 Member Function Documentation . . . 87
3.21 Counter Class Reference . . . 90
3.21.1 Detailed Description . . . 92
3.21.2 Constructor & Destructor Documentation . . . 92
3.21.3 Member Function Documentation . . . 93
3.22 CppCompiler Class Reference . . . 96
3.22.1 Constructor & Destructor Documentation . . . 98
3.22.2 Member Function Documentation . . . 98
3.23 CppForG Class Reference . . . 105
3.23.1 Detailed Description . . . 106

3.23.2 Constructor & Destructor Documentation . . . 106
3.23.3 Member Function Documentation . . . 107
3.24 CppSerializer Class Reference . . . 112
3.24.1 Constructor & Destructor Documentation . . . 113
3.24.2 Member Function Documentation . . . 113
3.25 Create Class Reference . . . 117
3.25.1 Detailed Description . . . 118
3.25.2 Constructor & Destructor Documentation . . . 119
3.25.3 Member Function Documentation . . . 119
3.26 DataAnalyser_if Class Reference . . . 124
3.26.1 Member Function Documentation . . . 124
3.27 Decide Class Reference . . . 125
3.27.1 Detailed Description . . . 127
3.27.2 Constructor & Destructor Documentation . . . 127
3.27.3 Member Function Documentation . . . 128
3.28 Delay::DEFAULT_VALUES Struct Reference . . . 131
3.28.1 Member Data Documentation . . . 131
3.29 Seize::DEFAULT_VALUES Struct Reference . . . 132
3.29.1 Member Data Documentation . . . 132
3.30 SourceModelComponent::DEFAULT_VALUES Struct Reference . . . 132
3.30.1 Member Data Documentation . . . 133
3.31 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference . . . 134
3.31.1 Constructor & Destructor Documentation . . . 134
3.31.2 Member Data Documentation . . . 135
3.32 Delay Class Reference . . . 135
3.32.1 Detailed Description . . . 137
3.32.2 Constructor & Destructor Documentation . . . 137
3.32.3 Member Function Documentation . . . 138
3.32.4 Member Data Documentation . . . 143
3.33 DiffEquations Class Reference . . . 144
3.33.1 Detailed Description . . . 145
3.33.2 Constructor & Destructor Documentation . . . 145
3.33.3 Member Function Documentation . . . 146
3.34 Dispose Class Reference . . . 151
3.34.1 Detailed Description . . . 152
3.34.2 Constructor & Destructor Documentation . . . 152
3.34.3 Member Function Documentation . . . 153
3.35 DropOff Class Reference . . . 156
3.35.1 Detailed Description . . . 158
3.35.2 Constructor & Destructor Documentation . . . 158
3.35.3 Member Function Documentation . . . 158
3.36 DummyComponent Class Reference . . . 161

3.36.1 Detailed Description . . . 163
3.36.2 Constructor & Destructor Documentation . . . 163
3.36.3 Member Function Documentation . . . 163
3.37 DummyElement Class Reference . . . 166
3.37.1 Constructor & Destructor Documentation . . . 168
3.37.2 Member Function Documentation . . . 168
3.38 Enter Class Reference . . . 170
3.38.1 Detailed Description . . . 172
3.38.2 Constructor & Destructor Documentation . . . 173
3.38.3 Member Function Documentation . . . 173
3.39 Entity Class Reference . . . 178
3.39.1 Detailed Description . . . 180
3.39.2 Member Function Documentation . . . 181
3.39.3 Friends And Related Function Documentation . . . 185
3.40 EntityGroup Class Reference . . . 186
3.40.1 Constructor & Destructor Documentation . . . 187
3.40.2 Member Function Documentation . . . 187
3.41 EntityType Class Reference . . . 191
3.41.1 Constructor & Destructor Documentation . . . 193
3.41.2 Member Function Documentation . . . 194
3.42 PersistenceRecord::Entry Struct Reference . . . 198
3.42.1 Detailed Description . . . 198
3.42.2 Member Enumeration Documentation . . . 198
3.42.3 Member Data Documentation . . . 199
3.43 Event Class Reference . . . 199
3.43.1 Detailed Description . . . 201
3.43.2 Constructor & Destructor Documentation . . . 201
3.43.3 Member Function Documentation . . . 201
3.43.4 Member Data Documentation . . . 202
3.44 Exact Class Reference . . . 202
3.44.1 Constructor & Destructor Documentation . . . 203
3.44.2 Member Function Documentation . . . 203
3.45 Exit Class Reference . . . 205
3.45.1 Detailed Description . . . 207
3.45.2 of Cells Number of contiguous conveyor cells the entity will relinquish. . . . 207
3.45.3 Constructor & Destructor Documentation . . . 207
3.45.4 Member Function Documentation . . . 207
3.46 ExperimentManager Class Reference . . . 210
3.46.1 Constructor & Destructor Documentation . . . 211
3.46.2 Member Function Documentation . . . 211
3.47 ExperimentManager_if Class Reference . . . 213
3.47.1 Detailed Description . . . 214

3.47.2 Member Function Documentation . . . 214
3.48 ExperimentManagerDefaultImpl1 Class Reference . . . 215
3.48.1 Constructor & Destructor Documentation . . . 215
3.48.2 Member Function Documentation . . . 216
3.49 ExtendedFSM Class Reference . . . 217
3.49.1 Constructor & Destructor Documentation . . . 218
3.49.2 Member Function Documentation . . . 218
3.50 Failure Class Reference . . . 221
3.50.1 Detailed Description . . . 223
3.50.2 Member Enumeration Documentation . . . 224
3.50.3 Constructor & Destructor Documentation . . . 224
3.50.4 Member Function Documentation . . . 224
3.51 File Class Reference . . . 229
3.51.1 Detailed Description . . . 231
3.51.2 Constructor & Destructor Documentation . . . 231
3.51.3 Member Function Documentation . . . 232
3.52 Fitter_if Class Reference . . . 234
3.52.1 Member Function Documentation . . . 236
3.53 FitterDummyImpl Class Reference . . . 238
3.53.1 Constructor & Destructor Documentation . . . 239
3.53.2 Member Function Documentation . . . 239
3.54 Formula Class Reference . . . 241
3.54.1 Constructor & Destructor Documentation . . . 243
3.54.2 Member Function Documentation . . . 243
3.55 FSM_State Class Reference . . . 247
3.55.1 Constructor & Destructor Documentation . . . 248
3.55.2 Member Function Documentation . . . 248
3.56 FSM_Transition Class Reference . . . 249
3.56.1 Constructor & Destructor Documentation . . . 250
3.57 ParserManager::GenerateNewParserResult Struct Reference . . . 250
3.57.1 Member Data Documentation . . . 250
3.58 GenesysApplication_if Class Reference . . . 251
3.58.1 Member Function Documentation . . . 252
3.59 genesyspp_driver Class Reference . . . 252
3.59.1 Constructor & Destructor Documentation . . . 253
3.59.2 Member Function Documentation . . . 253
3.60 GenesysShell Class Reference . . . 257
3.60.1 Constructor & Destructor Documentation . . . 258
3.60.2 Member Function Documentation . . . 259
3.61 GenesysShell_if Class Reference . . . 263
3.61.1 Member Function Documentation . . . 264
3.62 GenSerializer Class Reference . . . 267

3.62.1 Constructor & Destructor Documentation . . . 268
3.62.2 Member Function Documentation . . . 268
3.62.3 Friends And Related Function Documentation . . . 272
3.63 InternalEvent Class Reference . . . 272
3.63.1 Constructor & Destructor Documentation . . . 273
3.63.2 Member Function Documentation . . . 273
3.64 Json Struct Reference . . . 274
3.64.1 Member Enumeration Documentation . . . 275
3.64.2 Member Data Documentation . . . 275
3.65 JsonSerializer Class Reference . . . 276
3.65.1 Constructor & Destructor Documentation . . . 277
3.65.2 Member Function Documentation . . . 277
3.66 Label Class Reference . . . 280
3.66.1 Constructor & Destructor Documentation . . . 282
3.66.2 Member Function Documentation . . . 282
3.67 Leave Class Reference . . . 285
3.67.1 Detailed Description . . . 287
3.67.2 of Cells Number of contiguous cells the entity requires. . . . 288
3.67.3 Constructor & Destructor Documentation . . . 288
3.67.4 Member Function Documentation . . . 288
3.68 LicenceManager Class Reference . . . 292
3.68.1 Detailed Description . . . 293
3.68.2 Constructor & Destructor Documentation . . . 293
3.68.3 Member Function Documentation . . . 293
3.69 List< T > Class Template Reference . . . 294
3.69.1 Detailed Description . . . 295
3.69.2 Member Typedef Documentation . . . 295
3.69.3 Constructor & Destructor Documentation . . . 295
3.69.4 Member Function Documentation . . . 295
3.69.5 Member Data Documentation . . . 298
3.70 ListObservable< T > Class Template Reference . . . 298
3.70.1 Detailed Description . . . 300
3.70.2 Member Typedef Documentation . . . 300
3.70.3 Constructor & Destructor Documentation . . . 300
3.70.4 Member Function Documentation . . . 300
3.71 LSODE Class Reference . . . 301
3.71.1 Detailed Description . . . 303
3.71.2 Constructor & Destructor Documentation . . . 303
3.71.3 Member Function Documentation . . . 303
3.72 MarkovChain Class Reference . . . 308
3.72.1 Constructor & Destructor Documentation . . . 309
3.72.2 Member Function Documentation . . . 309

3.73 Match Class Reference . . . 313
3.73.1 Detailed Description . . . 315
3.73.2 Member Enumeration Documentation . . . 315
3.73.3 Constructor & Destructor Documentation . . . 316
3.73.4 Member Function Documentation . . . 316
3.74 Model Class Reference . . . 320
3.74.1 Detailed Description . . . 321
3.74.2 Constructor & Destructor Documentation . . . 321
3.74.3 Member Function Documentation . . . 322
3.75 ModelChecker_if Class Reference . . . 332
3.75.1 Detailed Description . . . 332
3.75.2 Member Function Documentation . . . 333
3.76 ModelCheckerDefaultImpl1 Class Reference . . . 334
3.76.1 Constructor & Destructor Documentation . . . 334
3.76.2 Member Function Documentation . . . 335
3.77 ModelComponent Class Reference . . . 339
3.77.1 Detailed Description . . . 340
3.77.2 Constructor & Destructor Documentation . . . 340
3.77.3 Member Function Documentation . . . 341
3.77.4 Member Data Documentation . . . 345
3.78 ModelDataDefinition Class Reference . . . 346
3.78.1 Detailed Description . . . 348
3.78.2 Constructor & Destructor Documentation . . . 348
3.78.3 Member Function Documentation . . . 348
3.78.4 Member Data Documentation . . . 359
3.79 ModelDataManager Class Reference . . . 360
3.79.1 Detailed Description . . . 360
3.79.2 Constructor & Destructor Documentation . . . 360
3.79.3 Member Function Documentation . . . 361
3.80 ModelEvent Class Reference . . . 365
3.80.1 Detailed Description . . . 365
3.80.2 Constructor & Destructor Documentation . . . 365
3.80.3 Member Function Documentation . . . 366
3.81 ModelInfo Class Reference . . . 366
3.81.1 Detailed Description . . . 366
3.81.2 Constructor & Destructor Documentation . . . 366
3.81.3 Member Function Documentation . . . 366
3.82 ModelManager Class Reference . . . 368
3.82.1 Constructor & Destructor Documentation . . . 369
3.82.2 Member Function Documentation . . . 369
3.83 ModelPersistence_if Class Reference . . . 372
3.83.1 Member Enumeration Documentation . . . 373

3.83.2 Member Function Documentation . . . 373
3.84 ModelPersistenceDefaultImpl2 Class Reference . . . 374
3.84.1 Constructor & Destructor Documentation . . . 375
3.84.2 Member Function Documentation . . . 375
3.85 ModelSerializer Class Reference . . . 377
3.85.1 Detailed Description . . . 378
3.85.2 Constructor & Destructor Documentation . . . 378
3.85.3 Member Function Documentation . . . 379
3.86 ModelSimulation Class Reference . . . 381
3.86.1 Detailed Description . . . 382
3.86.2 Constructor & Destructor Documentation . . . 382
3.86.3 Member Function Documentation . . . 383
3.86.4 Friends And Related Function Documentation . . . 390
3.87 ParserManager::NewParser Struct Reference . . . 390
3.87.1 Member Data Documentation . . . 391
3.88 obj_t Class Reference . . . 391
3.88.1 Constructor & Destructor Documentation . . . 391
3.88.2 Member Data Documentation . . . 392
3.89 ODEfunction Class Reference . . . 392
3.89.1 Constructor & Destructor Documentation . . . 392
3.89.2 Member Data Documentation . . . 393
3.90 OLD_ODEelement Class Reference . . . 393
3.90.1 Constructor & Destructor Documentation . . . 395
3.90.2 Member Function Documentation . . . 395
3.91 OnEventManager Class Reference . . . 398
3.91.1 Detailed Description . . . 400
3.91.2 Constructor & Destructor Documentation . . . 400
3.91.3 Member Function Documentation . . . 400
3.92 Parser_if Class Reference . . . 406
3.92.1 Member Function Documentation . . . 407
3.93 ParserChangesInformation Class Reference . . . 408
3.93.1 Constructor & Destructor Documentation . . . 409
3.93.2 Member Function Documentation . . . 409
3.94 ParserDefaultImpl1 Class Reference . . . 410
3.94.1 Constructor & Destructor Documentation . . . 411
3.94.2 Member Function Documentation . . . 412
3.95 ParserDefaultImpl2 Class Reference . . . 412
3.95.1 Constructor & Destructor Documentation . . . 413
3.95.2 Member Function Documentation . . . 414
3.96 ParserManager Class Reference . . . 416
3.96.1 Constructor & Destructor Documentation . . . 416
3.96.2 Member Function Documentation . . . 416

3.97 PersistenceRecord Class Reference . . . 416
3.97.1 Detailed Description . . . 418
3.97.2 Member Typedef Documentation . . . 418
3.97.3 Constructor & Destructor Documentation . . . 418
3.97.4 Member Function Documentation . . . 418
3.98 PersistentObject_base Class Reference . . . 422
3.98.1 Detailed Description . . . 422
3.98.2 Constructor & Destructor Documentation . . . 422
3.98.3 Member Function Documentation . . . 423
3.99 PickableStationItem Class Reference . . . 424
3.99.1 Constructor & Destructor Documentation . . . 424
3.99.2 Member Function Documentation . . . 425
3.100 PickStation Class Reference . . . 427
3.100.1 Detailed Description . . . 428
3.100.2 Member Enumeration Documentation . . . 429
3.100.3 Constructor & Destructor Documentation . . . 429
3.100.4 Member Function Documentation . . . 429
3.101 PickUp Class Reference . . . 434
3.101.1 Detailed Description . . . 436
3.101.2 Constructor & Destructor Documentation . . . 436
3.101.3 Member Function Documentation . . . 436
3.102 Plugin Class Reference . . . 437
3.102.1 Detailed Description . . . 437
3.102.2 Constructor & Destructor Documentation . . . 437
3.102.3 Member Function Documentation . . . 438
3.103 PluginConnector_if Class Reference . . . 440
3.103.1 Member Function Documentation . . . 441
3.104 PluginConnectorDummyImpl1 Class Reference . . . 442
3.104.1 Constructor & Destructor Documentation . . . 443
3.104.2 Member Function Documentation . . . 443
3.105 PluginInformation Class Reference . . . 444
3.105.1 Constructor & Destructor Documentation . . . 445
3.105.2 Member Function Documentation . . . 445
3.106 PluginManager Class Reference . . . 449
3.106.1 Constructor & Destructor Documentation . . . 450
3.106.2 Member Function Documentation . . . 450
3.107 ProbabilityDistribution Class Reference . . . 454
3.107.1 Member Function Documentation . . . 455
3.108 ProbabilityDistributionBase Class Reference . . . 456
3.108.1 Member Function Documentation . . . 457
3.109 Process Class Reference . . . 461
3.109.1 Detailed Description . . . 463

3.109.2 Constructor & Destructor Documentation . . . 463
3.109.3 Member Function Documentation . . . 464
3.110 PropertyGenesys Class Reference . . . 473
3.110.1 Constructor & Destructor Documentation . . . 474
3.110.2 Member Function Documentation . . . 475
3.110.3 Member Data Documentation . . . 477
3.111 PropertyGenesysBase Class Reference . . . 478
3.111.1 Constructor & Destructor Documentation . . . 479
3.111.2 Member Function Documentation . . . 479
3.111.3 Member Data Documentation . . . 480
3.112 Queue Class Reference . . . 481
3.112.1 Detailed Description . . . 482
3.112.2 Member Enumeration Documentation . . . 483
3.112.3 Constructor & Destructor Documentation . . . 483
3.112.4 Member Function Documentation . . . 483
3.113 QueueableItem Class Reference . . . 489
3.113.1 Member Enumeration Documentation . . . 490
3.113.2 Constructor & Destructor Documentation . . . 490
3.113.3 Member Function Documentation . . . 491
3.114 Record Class Reference . . . 493
3.114.1 Detailed Description . . . 495
3.114.2 Constructor & Destructor Documentation . . . 496
3.114.3 Member Function Documentation . . . 496
3.115 Release Class Reference . . . 501
3.115.1 Detailed Description . . . 503
3.115.2 Constructor & Destructor Documentation . . . 503
3.115.3 Member Function Documentation . . . 504
3.116 Remove Class Reference . . . 509
3.116.1 Detailed Description . . . 510
3.116.2 Member Enumeration Documentation . . . 510
3.116.3 Constructor & Destructor Documentation . . . 511
3.116.4 Member Function Documentation . . . 511
3.117 Resource Class Reference . . . 516
3.117.1 Detailed Description . . . 519
3.117.2 Member Typedef Documentation . . . 519
3.117.3 Member Enumeration Documentation . . . 519
3.117.4 Constructor & Destructor Documentation . . . 520
3.117.5 Member Function Documentation . . . 520
3.117.6 Friends And Related Function Documentation . . . 527
3.118 Sampler_if::RNG_Parameters Struct Reference . . . 527
3.118.1 Detailed Description . . . 528
3.118.2 Constructor & Destructor Documentation . . . 528

3.119 Route Class Reference . . . 528
3.119.1 Detailed Description . . . 530
3.119.2 Member Enumeration Documentation . . . 531
3.119.3 Constructor & Destructor Documentation . . . 531
3.119.4 Member Function Documentation . . . 531
3.120 Sampler_if Class Reference . . . 537
3.120.1 Detailed Description . . . 539
3.120.2 Member Function Documentation . . . 539
3.121 SamplerBoostImpl Class Reference . . . 542
3.121.1 Constructor & Destructor Documentation . . . 544
3.121.2 Member Function Documentation . . . 544
3.122 SamplerDefaultImpl1 Class Reference . . . 546
3.122.1 Constructor & Destructor Documentation . . . 548
3.122.2 Member Function Documentation . . . 548
3.123 ScenarioExperiment_if Class Reference . . . 554
3.124 SchedulableItem Class Reference . . . 554
3.124.1 Member Enumeration Documentation . . . 555
3.124.2 Constructor & Destructor Documentation . . . 555
3.124.3 Member Function Documentation . . . 555
3.125 Schedule Class Reference . . . 556
3.125.1 Constructor & Destructor Documentation . . . 558
3.125.2 Member Function Documentation . . . 558
3.126 Search Class Reference . . . 562
3.126.1 Detailed Description . . . 564
3.126.2 Member Enumeration Documentation . . . 565
3.126.3 Constructor & Destructor Documentation . . . 565
3.126.4 Member Function Documentation . . . 565
3.127 SeizableItem Class Reference . . . 571
3.127.1 Member Enumeration Documentation . . . 572
3.127.2 Constructor & Destructor Documentation . . . 572
3.127.3 Member Function Documentation . . . 573
3.128 Seize Class Reference . . . 578
3.128.1 Detailed Description . . . 580
3.128.2 Constructor & Destructor Documentation . . . 581
3.128.3 Member Function Documentation . . . 582
3.128.4 Member Data Documentation . . . 588
3.129 Separate Class Reference . . . 588
3.129.1 Detailed Description . . . 590
3.129.2 of Duplicates Number of outgoing entities that will leave the module, in . . . 590
3.129.3 Constructor & Destructor Documentation . . . 590
3.129.4 Member Function Documentation . . . 591
3.130 Sequence Class Reference . . . 594

3.130.1 Detailed Description . . . 596
3.130.2 Constructor & Destructor Documentation . . . 596
3.130.3 Member Function Documentation . . . 596
3.131 SequenceStep Class Reference . . . 599
3.131.1 Constructor & Destructor Documentation . . . 601
3.131.2 Member Function Documentation . . . 601
3.132 Set Class Reference . . . 604
3.132.1 Detailed Description . . . 605
3.132.2 Constructor & Destructor Documentation . . . 605
3.132.3 Member Function Documentation . . . 606
3.133 Signal Class Reference . . . 609
3.133.1 Detailed Description . . . 611
3.133.2 Constructor & Destructor Documentation . . . 611
3.133.3 Member Function Documentation . . . 612
3.134 SignalData Class Reference . . . 616
3.134.1 Member Typedef Documentation . . . 617
3.134.2 Constructor & Destructor Documentation . . . 617
3.134.3 Member Function Documentation . . . 618
3.135 SimulationControl Class Reference . . . 621
3.135.1 Constructor & Destructor Documentation . . . 621
3.135.2 Member Function Documentation . . . 621
3.135.3 Member Data Documentation . . . 622
3.136 SimulationControlBool Class Reference . . . 623
3.136.1 Constructor & Destructor Documentation . . . 624
3.136.2 Member Function Documentation . . . 624
3.137 SimulationControlDouble Class Reference . . . 624
3.137.1 Constructor & Destructor Documentation . . . 626
3.137.2 Member Function Documentation . . . 626
3.138 SimulationControlInt Class Reference . . . 626
3.138.1 Constructor & Destructor Documentation . . . 628
3.138.2 Member Function Documentation . . . 628
3.139 SimulationControlString Class Reference . . . 628
3.139.1 Constructor & Destructor Documentation . . . 630
3.139.2 Member Function Documentation . . . 630
3.140 SimulationControlTimeUnit Class Reference . . . 630
3.140.1 Constructor & Destructor Documentation . . . 632
3.140.2 Member Function Documentation . . . 632
3.141 SimulationControlUInt Class Reference . . . 632
3.141.1 Constructor & Destructor Documentation . . . 634
3.141.2 Member Function Documentation . . . 634
3.142 SimulationControlUShort Class Reference . . . 634
3.142.1 Constructor & Destructor Documentation . . . 636

3.142.2 Member Function Documentation . . . 636
3.143 SimulationEvent Class Reference . . . 636
3.143.1 Detailed Description . . . 637
3.143.2 Member Function Documentation . . . 637
3.143.3 Friends And Related Function Documentation . . . 639
3.144 SimulationExperiment Class Reference . . . 639
3.144.1 Constructor & Destructor Documentation . . . 639
3.145 SimulationReporter_if Class Reference . . . 640
3.145.1 Member Function Documentation . . . 640
3.146 SimulationReporterDefaultImpl1 Class Reference . . . 641
3.146.1 Detailed Description . . . 642
3.146.2 Constructor & Destructor Documentation . . . 642
3.146.3 Member Function Documentation . . . 643
3.147 SimulationScenario Class Reference . . . 646
3.147.1 Detailed Description . . . 647
3.147.2 Constructor & Destructor Documentation . . . 647
3.147.3 Member Function Documentation . . . 647
3.148 Simulator Class Reference . . . 649
3.148.1 Detailed Description . . . 649
3.148.2 Constructor & Destructor Documentation . . . 649
3.148.3 Member Function Documentation . . . 650
3.148.4 Friends And Related Function Documentation . . . 651
3.149 SinkModelComponent Class Reference . . . 651
3.149.1 Detailed Description . . . 653
3.149.2 Constructor & Destructor Documentation . . . 653
3.149.3 Member Function Documentation . . . 653
3.150 Solver_if Class Reference . . . 654
3.150.1 Detailed Description . . . 656
3.150.2 Member Function Documentation . . . 656
3.151 SolverDefaultImpl1 Class Reference . . . 658
3.151.1 Constructor & Destructor Documentation . . . 660
3.151.2 Member Function Documentation . . . 660
3.152 SortFile Class Reference . . . 662
3.152.1 Constructor & Destructor Documentation . . . 662
3.152.2 Member Function Documentation . . . 663
3.153 SourceModelComponent Class Reference . . . 663
3.153.1 Detailed Description . . . 665
3.153.2 Constructor & Destructor Documentation . . . 666
3.153.3 Member Function Documentation . . . 666
3.153.4 Member Data Documentation . . . 671
3.154 Start Class Reference . . . 672
3.154.1 Detailed Description . . . 674

3.154.2 Constructor & Destructor Documentation . . . 674
3.154.3 Member Function Documentation . . . 674
3.155 Station Class Reference . . . 677
3.155.1 Constructor & Destructor Documentation . . . 679
3.155.2 Member Function Documentation . . . 679
3.156 Statistics_if Class Reference . . . 684
3.156.1 Detailed Description . . . 686
3.156.2 Member Function Documentation . . . 686
3.157 StatisticsCollector Class Reference . . . 688
3.157.1 Detailed Description . . . 689
3.157.2 Constructor & Destructor Documentation . . . 689
3.157.3 Member Function Documentation . . . 690
3.158 StatisticsDatafile_if Class Reference . . . 692
3.158.1 Member Function Documentation . . . 694
3.159 StatisticsDatafileDefaultImpl1 Class Reference . . . 695
3.159.1 Constructor & Destructor Documentation . . . 697
3.159.2 Member Function Documentation . . . 697
3.160 StatisticsDefaultImpl1 Class Reference . . . 704
3.160.1 Constructor & Destructor Documentation . . . 705
3.160.2 Member Function Documentation . . . 706
3.161 Stop Class Reference . . . 708
3.161.1 Detailed Description . . . 709
3.161.2 Constructor & Destructor Documentation . . . 709
3.161.3 Member Function Documentation . . . 710
3.162 Storage Class Reference . . . 712
3.162.1 Constructor & Destructor Documentation . . . 714
3.162.2 Member Function Documentation . . . 715
3.163 Store Class Reference . . . 719
3.163.1 Detailed Description . . . 720
3.163.2 Constructor & Destructor Documentation . . . 720
3.163.3 Member Function Documentation . . . 721
3.164 Submodel Class Reference . . . 723
3.164.1 Detailed Description . . . 725
3.164.2 Constructor & Destructor Documentation . . . 725
3.164.3 Member Function Documentation . . . 725
3.165 Token Struct Reference . . . 728
3.165.1 Member Enumeration Documentation . . . 728
3.165.2 Member Data Documentation . . . 729
3.166 TraceErrorEvent Class Reference . . . 729
3.166.1 Constructor & Destructor Documentation . . . 730
3.166.2 Member Function Documentation . . . 730
3.167 TraceEvent Class Reference . . . 731

3.167.1 Constructor & Destructor Documentation . . . 731
3.167.2 Member Function Documentation . . . 731
3.168 TraceManager Class Reference . . . 732
3.168.1 Detailed Description . . . 733
3.168.2 Member Enumeration Documentation . . . 733
3.168.3 Constructor & Destructor Documentation . . . 733
3.168.4 Member Function Documentation . . . 733
3.169 TraceSimulationEvent Class Reference . . . 738
3.169.1 Constructor & Destructor Documentation . . . 739
3.169.2 Member Function Documentation . . . 739
3.170 TraceSimulationProcess Class Reference . . . 740
3.170.1 Detailed Description . . . 740
3.170.2 Constructor & Destructor Documentation . . . 741
3.171 TraitsApp< T > Struct Template Reference . . . 741
3.172 TraitsApp< GenesysApplication_if > Struct Reference . . . 741
3.172.1 Detailed Description . . . 741
3.172.2 Member Data Documentation . . . 741
3.173 TraitsKernel< T > Struct Template Reference . . . 742
3.174 TraitsKernel< Collector_if > Struct Reference . . . 742
3.174.1 Member Typedef Documentation . . . 742
3.175 TraitsKernel< Model > Struct Reference . . . 742
3.175.1 Member Typedef Documentation . . . 742
3.175.2 Member Data Documentation . . . 743
3.176 TraitsKernel< ModelChecker_if > Struct Reference . . . 743
3.176.1 Member Typedef Documentation . . . 743
3.176.2 Member Data Documentation . . . 743
3.177 TraitsKernel< ModelComponent > Struct Reference . . . 744
3.177.1 Member Data Documentation . . . 744
3.178 TraitsKernel< ModelDataDefinition > Struct Reference . . . 744
3.178.1 Member Data Documentation . . . 744
3.179 TraitsKernel< ModelPersistence_if > Struct Reference . . . 745
3.179.1 Member Typedef Documentation . . . 745
3.179.2 Member Data Documentation . . . 745
3.180 TraitsKernel< Parser_if > Struct Reference . . . 745
3.180.1 Member Typedef Documentation . . . 745
3.181 TraitsKernel< PluginConnector_if > Struct Reference . . . 746
3.181.1 Member Typedef Documentation . . . 746
3.181.2 Member Data Documentation . . . 746
3.182 TraitsKernel< Sampler_if > Struct Reference . . . 746
3.182.1 Member Typedef Documentation . . . 746
3.183 TraitsKernel< SimulationReporter_if > Struct Reference . . . 747
3.183.1 Member Typedef Documentation . . . 747

3.183.2 Member Data Documentation . . . 747
3.184 TraitsKernel< Simulator > Struct Reference . . . 747
3.184.1 Member Data Documentation . . . 748
3.185 TraitsKernel< Statistics_if > Struct Reference . . . 748
3.185.1 Member Typedef Documentation . . . 748
3.185.2 Member Data Documentation . . . 748
3.186 TraitsKernel< StatisticsDatafile_if > Struct Reference . . . 748
3.186.1 Member Typedef Documentation . . . 749
3.186.2 Member Data Documentation . . . 749
3.187 TraitsTerminalApp< T > Struct Template Reference . . . 749
3.188 TraitsTerminalApp< GenesysApplication_if > Struct Reference . . . 749
3.188.1 Detailed Description . . . 749
3.188.2 Member Typedef Documentation . . . 749
3.189 TraitsTools< T > Struct Template Reference . . . 750
3.190 TraitsTools< Fitter_if > Struct Reference . . . 750
3.190.1 Detailed Description . . . 750
3.190.2 Member Typedef Documentation . . . 750
3.191 TraitsTools< HypothesisTester_if > Struct Reference . . . 750
3.191.1 Detailed Description . . . 750
3.191.2 Member Typedef Documentation . . . 751
3.191.3 Member Data Documentation . . . 751
3.192 TraitsTools< Solver_if > Struct Reference . . . 751
3.192.1 Detailed Description . . . 751
3.192.2 Member Typedef Documentation . . . 751
3.192.3 Member Data Documentation . . . 751
3.193 FSM_Transition::TransitionType Struct Reference . . . 752
3.193.1 Member Data Documentation . . . 752
3.194 Unstore Class Reference . . . 753
3.194.1 Detailed Description . . . 754
3.194.2 Constructor & Destructor Documentation . . . 754
3.194.3 Member Function Documentation . . . 755
3.195 Util Class Reference . . . 757
3.195.1 Member Typedef Documentation . . . 758
3.195.2 Member Enumeration Documentation . . . 758
3.195.3 Member Function Documentation . . . 759
3.196 Variable Class Reference . . . 764
3.196.1 Detailed Description . . . 766
3.196.2 Constructor & Destructor Documentation . . . 767
3.196.3 Member Function Documentation . . . 767
3.197 Wait Class Reference . . . 771
3.197.1 Detailed Description . . . 773
3.197.2 Member Enumeration Documentation . . . 774

3.197.3 Constructor & Destructor Documentation . . . 774
3.197.4 Member Function Documentation . . . 774
3.198 Waiting Class Reference . . . 779
3.198.1 Constructor & Destructor Documentation . . . 780
3.198.2 Member Function Documentation . . . 781
3.199 WaitingResource Class Reference . . . 782
3.199.1 Constructor & Destructor Documentation . . . 782
3.199.2 Member Function Documentation . . . 783
3.200 Write Class Reference . . . 784
3.200.1 Detailed Description . . . 785
3.200.2 Member Enumeration Documentation . . . 785
3.200.3 Constructor & Destructor Documentation . . . 786
3.200.4 Member Function Documentation . . . 786
3.201 XmlSerializer Class Reference . . . 790
3.201.1 Constructor & Destructor Documentation . . . 791
3.201.2 Member Function Documentation . . . 791

1 Hierarchical Index

1.1 Class Hierarchy

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

Assignment 25

Collector_if 71

- **CollectorDatafile_if** 73
  - **CollectorDatafileDefaultImpl1** 75
- **CollectorDefaultImpl1** 78

CppCompiler::CompilationResult 80

CompilerThread 81

ComponentManager 81

Connection 85

ConnectionChannel 86

ConnectionManager 86

DataAnalyser_if 124

Delay::DEFAULT_VALUES 131

Seize::DEFAULT_VALUES 132

SourceModelComponent::DEFAULT_VALUES 132

PersistenceRecord::Entry 198

Event 199

InternalEvent 272

Exact 202

ExperimentManager 210

ExperimentManager_if 213

ExperimentManagerDefaultImpl1 215

Fitter_if 234

FitterDummyImpl 238

ParserManager::GenerateNewParserResult 250

GenesysApplication_if 251

BaseGenesysTerminalApplication 33

GenesysShell 257

GenesysShell_if 263

genesyspp_driver 252

Json 274

LicenceManager 292

List< T > 294

ListObservable< T > 298

Model 320

ModelChecker_if 332

ModelCheckerDefaultImpl1 334

ModelDataManager 360

ModelEvent 365

ModelInfo 366

ModelManager 368

ModelPersistence_if 372

ModelPersistenceDefaultImpl2 374

ModelSerializer 377

CppSerializer 112

GenSerializer 267

JsonSerializer 276

XmlSerializer 790

ModelSimulation 381

ParserManager::NewParser 390

obj_t 391

ODEfunction 392

OnEventManager 398

Parser_if 406

ParserDefaultImpl1 410

ParserDefaultImpl2 412

ParserChangesInformation 408

ParserManager 416

PersistenceRecord 416

PersistentObject_base 422

FSM_State 247

FSM_Transition 249

ModelDataDefinition 346

Attribute 29

Counter 90

CppCompiler 96

DummyElement 166

Entity 178

EntityGroup 186

EntityType 191

ExtendedFSM 217

Failure 221

File 229

Formula 241

Label 280

ModelComponent 339

Access 14

Assign 19

Batch 43

Buffer 52

CellularAutomata 60

Clone 64

CppForG 105

Decide 125

Delay 135

DiffEquations 144

DropOff 156

DummyComponent 161

Enter 170

Exit 205

LSODE 301

Leave 285

MarkovChain 308

Match 313

PickStation 427

Process 461

Record 493

Release 501

Remove 509

PickUp 434

Route 528

Search 562

Seize 578

Separate 588

Signal 609

SinkModelComponent 651

Dispose 151

SourceModelComponent 663

Create 117

Start 672

Stop 708

Store 719

Submodel 723

Unstore 753

Wait 771

Write 784

OLD_ODEelement 393

Queue 481

Resource 516

Schedule 556

Sequence 594

Set 604

SignalData 616

Station 677

StatisticsCollector 688

Storage 712

Variable 764

SequenceStep 599

PickableStationItem 424

Plugin 437

PluginConnector_if 440

PluginConnectorDummyImpl1 442

PluginInformation 444

PluginManager 449

ProbabilityDistributionBase 456

ProbabilityDistribution 454

PropertyGenesysBase 478

PropertyGenesys 473

SimulationControl 621

SimulationControlBool 623

SimulationControlDouble 624

SimulationControlInt 626

SimulationControlString 628

SimulationControlTimeUnit 630

SimulationControlUInt 632

SimulationControlUShort 634

QueueableItem 489

Sampler_if::RNG_Parameters 527

SamplerBoostImpl::BoostImplRNG_Parameters 52

SamplerDefaultImpl1::DefaultImpl1RNG_Parameters 134

Sampler_if 537

SamplerBoostImpl 542

SamplerDefaultImpl1 546

ScenarioExperiment_if 554

SchedulableItem 554

SeizableItem 571

SimulationEvent 636

SimulationExperiment 639

SimulationReporter_if 640

SimulationReporterDefaultImpl1 641

SimulationScenario 646

Simulator 649

Solver_if 654

SolverDefaultImpl1 658

SortFile 662

Statistics_if 684

StatisticsDatafile_if 692

StatisticsDatafileDefaultImpl1 695

StatisticsDefaultImpl1 704

Token 728

TraceEvent 731

TraceErrorEvent 729

TraceSimulationEvent 738

TraceSimulationProcess 740

TraceManager 732

TraitsApp< T > 741

TraitsApp< GenesysApplication_if > 741

TraitsKernel< T > 742

TraitsKernel< Collector_if > 742

TraitsKernel< Model > 742

TraitsKernel< ModelChecker_if > 743

TraitsKernel< ModelComponent > 744

TraitsKernel< ModelDataDefinition > 744

TraitsKernel< ModelPersistence_if > 745

TraitsKernel< Parser_if > 745

TraitsKernel< PluginConnector_if > 746

TraitsKernel< Sampler_if > 746

TraitsKernel< SimulationReporter_if > 747

TraitsKernel< Simulator > 747

TraitsKernel< Statistics_if > 748

TraitsKernel< StatisticsDatafile_if > 748

TraitsTerminalApp< T > 749

TraitsTerminalApp< GenesysApplication_if > 749

TraitsTools< T > 750

TraitsTools< Fitter_if > 750

TraitsTools< HypothesisTester_if > 750

TraitsTools< Solver_if > 751

FSM_Transition::TransitionType 752

Util 757

Waiting 779

WaitingResource 782

2 Class Index

2.1 Class List

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

Access 14

Assign 19

Assignment 25

Attribute 29

BaseGenesysTerminalApplication 33

Batch 43

SamplerBoostImpl::BoostImplRNG_Parameters 52

Buffer 52

CellularAutomata 60

Clone 64

Collector_if 71

CollectorDatafile_if 73

CollectorDatafileDefaultImpl1 75

CollectorDefaultImpl1 78

CppCompiler::CompilationResult 80

CompilerThread 81

ComponentManager 81

Connection 85

ConnectionChannel 86

ConnectionManager 86

Counter 90

CppCompiler 96

CppForG 105

CppSerializer 112

Create 117

DataAnalyser_if 124

Decide 125

Delay::DEFAULT_VALUES 131

Seize::DEFAULT_VALUES 132

SourceModelComponent::DEFAULT_VALUES 132

SamplerDefaultImpl1::DefaultImpl1RNG_Parameters 134

Delay 135

DiffEquations 144

Dispose 151

DropOff 156

DummyComponent 161

DummyElement 166

Enter 170

Entity 178

EntityGroup 186

EntityType 191

PersistenceRecord::Entry
"Raw" entry, users should treat this as an opaque type since payloads could be in any format 198

Event 199

Exact 202

Exit 205

ExperimentManager 210

ExperimentManager_if 213

ExperimentManagerDefaultImpl1 215

ExtendedFSM 217

Failure 221

File 229

Fitter_if 234

FitterDummyImpl 238

Formula 241

FSM_State 247

FSM_Transition 249

ParserManager::GenerateNewParserResult 250

GenesysApplication_if 251

genesyspp_driver 252

GenesysShell 257

GenesysShell_if 263

GenSerializer 267

InternalEvent 272

Json 274

JsonSerializer 276

Label 280

Leave 285

LicenceManager 292

List< T > 294

ListObservable< T > 298

LSODE 301

MarkovChain 308

Match 313

Model 320

ModelChecker_if 332

ModelCheckerDefaultImpl1 334

ModelComponent 339

ModelDataDefinition 346

ModelDataManager 360

ModelEvent 365

ModelInfo 366

ModelManager 368

ModelPersistence_if 372

ModelPersistenceDefaultImpl2 374

ModelSerializer
Serialization/Deserialization interface for simulation models 377

ModelSimulation 381

ParserManager::NewParser 390

obj_t 391

ODEfunction 392

OLD_ODEelement 393

OnEventManager 398

Parser_if 406

ParserChangesInformation 408

ParserDefaultImpl1 410

ParserDefaultImpl2 412

ParserManager 416

PersistenceRecord
Serialization/Deserialization structure which behaves like an associative array 416

PersistentObject_base
Dynamic serializable interface 422

PickableStationItem 424

PickStation 427

PickUp 434

Plugin 437

PluginConnector_if 440

PluginConnectorDummyImpl1 442

PluginInformation 444

PluginManager 449

ProbabilityDistribution 454

ProbabilityDistributionBase 456

Process 461

PropertyGenesys 473

PropertyGenesysBase 478

Queue 481

QueueableItem 489

Record 493

Release 501

Remove 509

Resource 516

Sampler_if::RNG_Parameters 527

Route 528

Sampler_if 537

SamplerBoostImpl 542

SamplerDefaultImpl1 546

ScenarioExperiment_if 554

SchedulableItem 554

Schedule 556

Search 562

SeizableItem 571

Seize 578

Separate 588

Sequence 594

SequenceStep 599

Set 604

Signal 609

SignalData 616

SimulationControl 621

SimulationControlBool 623

SimulationControlDouble 624

SimulationControlInt 626

SimulationControlString 628

SimulationControlTimeUnit 630

SimulationControlUInt 632

SimulationControlUShort 634

SimulationEvent 636

SimulationExperiment 639

SimulationReporter_if 640

SimulationReporterDefaultImpl1 641

SimulationScenario 646

Simulator 649

SinkModelComponent 651

Solver_if 654

SolverDefaultImpl1 658

SortFile 662

SourceModelComponent 663

Start 672

Station 677

Statistics_if 684

StatisticsCollector 688

StatisticsDatafile_if 692

StatisticsDatafileDefaultImpl1 695

StatisticsDefaultImpl1 704

Stop 708

Storage 712

Store 719

Submodel 723

Token 728

TraceErrorEvent 729

TraceEvent 731

TraceManager 732

TraceSimulationEvent 738

TraceSimulationProcess 740

TraitsApp< T > 741

TraitsApp< GenesysApplication_if > 741

TraitsKernel< T > 742

TraitsKernel< Collector_if > 742

TraitsKernel< Model > 742

TraitsKernel< ModelChecker_if > 743

TraitsKernel< ModelComponent > 744

TraitsKernel< ModelDataDefinition > 744

TraitsKernel< ModelPersistence_if > 745

TraitsKernel< Parser_if > 745

TraitsKernel< PluginConnector_if > 746

TraitsKernel< Sampler_if > 746

TraitsKernel< SimulationReporter_if > 747

TraitsKernel< Simulator > 747

TraitsKernel< Statistics_if > 748

TraitsKernel< StatisticsDatafile_if > 748

TraitsTerminalApp< T > 749

TraitsTerminalApp< GenesysApplication_if > 749

TraitsTools< T > 750

TraitsTools< Fitter_if > 750

TraitsTools< HypothesisTester_if > 750

TraitsTools< Solver_if > 751

FSM_Transition::TransitionType 752

Unstore 753

Util 757

Variable 764

Wait 771

Waiting 779

WaitingResource 782

Write 784

XmlSerializer 790

3 Class Documentation

3.1 Access Class Reference

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

Inheritance diagram for Access:

Public Member Functions

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

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.1.1 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.

3.1.2 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.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 Access()

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

3.1.3.2 ~Access() `virtual Access::~Access ( ) [virtual], [default]`

3.1.4 Member Function Documentation

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.1.4.2 _loadInstance() `bool Access::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.1.4.3 _onDispatchEvent() `void Access::_onDispatchEvent ( Entity * entity, unsigned int inputPortNumber ) [protected], [virtual]`

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.1.4.4 _saveInstance()

```
void Access::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.1.4.5 GetPluginInformation()

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

Here is the call graph for this function:

3.1.4.6 LoadInstance()

```
ModelComponent * Access::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.1.4.7 NewInstance()

```
ModelDataDefinition * Access::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.1.4.8 show()

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

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.2 Assign Class Reference

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

Inheritance diagram for Assign:

Public Member Functions

- Assign (Model ∗model, std::string name="")
- virtual ∼Assign ()=default
- virtual std::string show ()

  virtual public methods
- List< Assignment ∗ > ∗ getAssignments () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.2.1 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.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Assign()

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

3.2.2.2 ~Assign() `virtual Assign::~Assign ( ) [virtual], [default]`

3.2.3 Member Function Documentation

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.2.3.2 _createInternalAndAttachedData() `void Assign::_createInternalAndAttachedData ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.2.3.3 _loadInstance()

```
bool Assign::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.2.3.4 _onDispatchEvent()

```
void Assign::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.2.3.5 _saveInstance()

```
void Assign::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

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

3.2.3.7 GetPluginInformation() `PluginInformation * Assign::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.2.3.8 LoadInstance()

```
ModelComponent * Assign::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.2.3.9 NewInstance()

```
ModelDataDefinition * Assign::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.2.3.10 show()

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

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.3 Assignment Class Reference

```
#include <AssignmentItem.h>
```

Public Member Functions

- Assignment (Model *model, std::string destination, std::string expression, bool isAttributeNotVariable=true)
- Assignment (std::string destination, std::string expression, bool isAttributeNotVariable=true)
- void setDestination (std::string _destination)
- std::string getDestination () const
- void setExpression (std::string _expression)
- std::string getExpression () const
- void setAttributeNotVariable (bool isAttributeNotVariable)
- bool isAttributeNotVariable () const
- bool loadInstance (PersistenceRecord *fields, unsigned int parentIndex)
- void saveInstance (PersistenceRecord *fields, unsigned int parentIndex, bool saveDefault)

3.3.1 Detailed Description

While the assign ans sequence classes allows you to perform multiple assignments, the assignmentItem class defines an assignment itself.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Assignment() [1/2]

```
Assignment::Assignment (
    Model * model,
    std::string destination,
    std::string expression,
    bool isAttributeNotVariable = true )
```

Here is the call graph for this function:

3.3.2.2 Assignment() [2/2]

```
Assignment::Assignment (
    std::string destination,
    std::string expression,
    bool isAttributeNotVariable = true )
```

Here is the call graph for this function:

3.3.3 Member Function Documentation

3.3.3.1 getDestination() `std::string Assignment::getDestination ( ) const`

3.3.3.2 getExpression() `std::string Assignment::getExpression ( ) const`

3.3.3.3 isAttributeNotVariable() `bool Assignment::isAttributeNotVariable ( ) const`

3.3.3.4 loadInstance()

```
bool Assignment::loadInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex )
```

Here is the call graph for this function:

3.3.3.5 saveInstance()

```
void Assignment::saveInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex,
    bool saveDefault )
```

Here is the call graph for this function:

3.3.3.6 setAttributeNotVariable() `void Assignment::setAttributeNotVariable ( bool isAttributeNotVariable )`

Here is the call graph for this function:

3.3.3.7 setDestination() `void Assignment::setDestination ( std::string _destination )`

3.3.3.8 setExpression() `void Assignment::setExpression ( std::string _expression )`

3.4 Attribute Class Reference

`#include <Attribute.h>`

Inheritance 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 ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.4.1 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.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Attribute()

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

3.4.2.2 ~Attribute()

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

3.4.3 Member Function Documentation

3.4.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.4.3.2 _loadInstance()

```
bool Attribute::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.4.3.3 _saveInstance()

```
void Attribute::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

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

Here is the call graph for this function:

3.4.3.5 LoadInstance() `ModelDataDefinition * Attribute::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.4.3.6 NewInstance() `ModelDataDefinition * Attribute::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

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

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.5 BaseGenesysTerminalApplication Class Reference

```
#include <BaseGenesysTerminalApplication.h>
```

Inheritance diagram for BaseGenesysTerminalApplication:

Public Member Functions

- BaseGenesysTerminalApplication ()
- virtual ~BaseGenesysTerminalApplication ()=default
- virtual int main (int argc, char **argv)
- virtual void Simulate (Simulator *genesys, Model *model, PluginManager *plugins)
- void setDefaultTraceHandlers (TraceManager *tm)
- void setDefaultEventHandlers (OnEventManager *oem)

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 onAfterProcessEventHandler (SimulationEvent *re)
- virtual void onBreakpointHandler (SimulationEvent *re)
- virtual void onEntityCreateHandler (SimulationEvent *re)
- virtual void onEntityMoveHandler (SimulationEvent *re)
- virtual void onSimulationStartHandler (SimulationEvent *re)
- virtual void onReplicationStepHandler (SimulationEvent *re)
- virtual void onReplicationStartHandler (SimulationEvent *re)
- virtual void onProcessEventHandler (SimulationEvent *re)
- virtual void onReplicationEndHandler (SimulationEvent *re)
- virtual void onSimulationEndHandler (SimulationEvent *re)
- virtual void onSimulationPausedHandler (SimulationEvent *re)
- virtual void onSimulationResumeHandler (SimulationEvent *re)
- virtual void onEntityRemoveHandler (SimulationEvent *re)

3.5.1 Constructor & Destructor Documentation

3.5.1.1 BaseGenesysTerminalApplication()

```
BaseGenesysTerminalApplication::BaseGenesysTerminal↩
Application ( )
```

3.5.1.2 ~BaseGenesysTerminalApplication()

```
virtual BaseGenesysTerminalApplication::~Base↩
GenesysTerminalApplication ( ) [virtual], [default]
```

3.5.2 Member Function Documentation

3.5.2.1 main()

```
int BaseGenesysTerminalApplication::main (
    int argc,
    char ** argv ) [virtual]
```

Implements GenesysApplication_if.

Reimplemented in GenesysShell.

Here is the call graph for this function:

3.5.2.2 onAfterProcessEventHandler()

```
void BaseGenesysTerminalApplication::onAfterProcessEvent↩
Handler (
            SimulationEvent * re )  [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.3 onBreakpointHandler()

```
void BaseGenesysTerminalApplication::onBreakpointHandler (
            SimulationEvent * re )  [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.4 onEntityCreateHandler()

```
void BaseGenesysTerminalApplication::onEntityCreateHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.5 onEntityMoveHandler()

```
void BaseGenesysTerminalApplication::onEntityMoveHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.6 onEntityRemoveHandler()

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

Here is the call graph for this function:

3.5.2.7 onProcessEventHandler()

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

Here is the call graph for this function:

3.5.2.8 onReplicationEndHandler()

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

Here is the call graph for this function:

3.5.2.9 onReplicationStartHandler()

```
void BaseGenesysTerminalApplication::onReplicationStart↩
Handler (
        SimulationEvent * re ) [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.10 onReplicationStepHandler()

```
void BaseGenesysTerminalApplication::onReplicationStepHandler (
        SimulationEvent * re ) [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.11 onSimulationEndHandler()

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

Here is the call graph for this function:

3.5.2.12 onSimulationPausedHandler()

```
void BaseGenesysTerminalApplication::onSimulationPausedHandler (
        SimulationEvent * re ) [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.13 onSimulationResumeHandler()

```
void BaseGenesysTerminalApplication::onSimulationResumeHandler (
        SimulationEvent * re ) [protected], [virtual]
```

3.5.2.14 onSimulationStartHandler()

```
void BaseGenesysTerminalApplication::onSimulationStart↩
Handler (
            SimulationEvent * re )  [protected], [virtual]
```

3.5.2.15 setDefaultEventHandlers()

```
void BaseGenesysTerminalApplication::setDefaultEventHandlers
(
            OnEventManager * oem )
```

Add default method to handle simulator events so information is plotted to output screen Here is the call graph for this function:

3.5.2.16 setDefaultTraceHandlers()

```
void BaseGenesysTerminalApplication::setDefaultTraceHandlers (
        TraceManager * tm )
```

Here is the call graph for this function:

3.5.2.17 Simulate()

```
void BaseGenesysTerminalApplication::Simulate (
        Simulator * genesys,
        Model * model,
        PluginManager * plugins )  [virtual]
```

3.5.2.18 traceErrorHandler()

```
void BaseGenesysTerminalApplication::traceErrorHandler (
        TraceErrorEvent e )  [protected], [virtual]
```

Here is the call graph for this function:

3.5.2.19 traceHandler() `void BaseGenesysTerminalApplication::traceHandler ( TraceEvent e ) [protected], [virtual]`

Here is the call graph for this function:

3.5.2.20 traceReportHandler() `void BaseGenesysTerminalApplication::traceReportHandler ( TraceEvent e ) [protected], [virtual]`

Here is the call graph for this function:

3.5.2.21 traceSimulationHandler() `void BaseGenesysTerminalApplication::traceSimulationHandler ( TraceSimulationEvent e ) [protected], [virtual]`

Here is the call graph for this function:

3.6 Batch Class Reference

`#include <Batch.h>`

Inheritance diagram for Batch:

Public Types

- enum class BatchType : int { Temporary = 0 , Permanent = 1 }
- enum class Rule : int { Any = 0 , ByAttribute = 1 }
- enum class GroupedAttribs : int { FirstEntity = 0 , LastEntity = 1 , SumAttributes = 2 }

Public Member Functions

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

  virtual public methods
- void setGroupedEntityType (EntityType *groupedEntityType)
- void setGroupedEntityTypeName (std::string groupedEntityTypeName)
- EntityType * getGroupedEntityType () const
- void setAttributeName (std::string attributeName)
- std::string getAttributeName () const
- void setBatchSize (std::string batchSize)
- std::string getBatchSize () const
- void setRule (Batch::Rule _rule)
- Batch::Rule getRule () const
- void setGroupedAttributes (Batch::GroupedAttribs _groupedAttributes)
- Batch::GroupedAttribs getGroupedAttributes () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _createInternalAndAttachedData ()
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.

Additional Inherited Members

3.6.1 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.

3.6.2 Member Enumeration Documentation

3.6.2.1 BatchType `enum class Batch::BatchType : int [strong]`

Enumerator

Temporary	
Permanent	

3.6.2.2 GroupedAttribs `enum class Batch::GroupedAttribs : int [strong]`

Enumerator

FirstEntity	
LastEntity	
SumAttributes	

3.6.2.3 Rule `enum class Batch::Rule : int [strong]`

Enumerator

Any	
ByAttribute	

3.6.3 Constructor & Destructor Documentation

3.6.3.1 Batch()

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

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

3.6.4 Member Function Documentation

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.6.4.2 _createInternalAndAttachedData() `void Batch::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.6.4.3 _loadInstance()

```
bool Batch::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.6.4.4 _onDispatchEvent()

```
void Batch::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.6.4.5 _saveInstance()

```
void Batch::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.6.4.6 getAttributeName() `std::string Batch::getAttributeName ( ) const`

3.6.4.7 getBatchSize() `std::string Batch::getBatchSize ( ) const`

3.6.4.8 getGroupedAttributes() `Batch::GroupedAttribs Batch::getGroupedAttributes ( ) const`

3.6.4.9 getGroupedEntityType() `EntityType * Batch::getGroupedEntityType ( ) const`

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

Here is the call graph for this function:

3.6.4.11 getRule() `Batch::Rule Batch::getRule ( ) const`

3.6.4.12 LoadInstance() `ModelComponent * Batch::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.6.4.13 NewInstance() `ModelDataDefinition * Batch::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.6.4.14 setAttributeName() `void Batch::setAttributeName ( std::string attributeName )`

3.6.4.15 setBatchSize() `void Batch::setBatchSize ( std::string batchSize )`

3.6.4.16 setGroupedAttributes()

```
void Batch::setGroupedAttributes (
    Batch::GroupedAttribs _groupedAttributes )
```

3.6.4.17 setGroupedEntityType()

```
void Batch::setGroupedEntityType (
    EntityType * groupedEntityType )
```

3.6.4.18 setGroupedEntityTypeName()

```
void Batch::setGroupedEntityTypeName (
    std::string groupedEntityTypeName )
```

Here is the call graph for this function:

3.6.4.19 setRule()

```
void Batch::setRule (
    Batch::Rule _rule )
```

3.6.4.20 show()

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

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.7 SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference

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

Inheritance diagram for SamplerBoostImpl::BoostImplRNG_Parameters:

Public Attributes

- double param

Additional Inherited Members

3.7.1 Member Data Documentation

3.7.1.1 param `double SamplerBoostImpl::BoostImplRNG_Parameters::param`

3.8 Buffer Class Reference

```
#include <Buffer.h>
```

Inheritance diagram for Buffer:

Public Types

- enum class AdvanceOn : int { NewArrivals = 0 , Signal = 1 }
- enum class ArrivalOnFullBufferRule : int { Dispose = 0 , SendToBulkPort = 1 , ReplaceLastPosition = 2 }

Public Member Functions

- Buffer (Model ∗model, std::string name="")

- virtual ~Buffer ()=default
- virtual std::string show ()

  virtual public methods
- Buffer::ArrivalOnFullBufferRule getarrivalOnFullBufferRule () const
- void setArrivalOnFullBufferRule (Buffer::ArrivalOnFullBufferRule newArrivalOnFullBufferRule)
- Buffer::AdvanceOn getadvanceOn () const
- void setAdvanceOn (Buffer::AdvanceOn newAdvanceOn)
- unsigned int getcapacity () const
- void setCapacity (unsigned int newCapacity)
- SignalData * getsignal () const
- void setSignal (SignalData *newSignal)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation * _getParserChangesInformation ()
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- virtual void _addProperty (PropertyBase *property)

Additional Inherited Members

3.8.1 Detailed Description

This component ...

3.8.2 Member Enumeration Documentation

3.8.2.1 AdvanceOn `enum class Buffer::AdvanceOn : int [strong]`

Enumerator

NewArrivals	
Signal	

3.8.2.2 ArrivalOnFullBufferRule `enum class Buffer::ArrivalOnFullBufferRule : int [strong]`

Enumerator

Dispose	
SendToBulkPort	
ReplaceLastPosition	

3.8.3 Constructor & Destructor Documentation

3.8.3.1 Buffer()

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

3.8.3.2 ~Buffer()

```
virtual Buffer::~Buffer ( ) [virtual], [default]
```

3.8.4 Member Function Documentation

3.8.4.1 _addProperty()

```
void Buffer::_addProperty (
    PropertyBase * property ) [protected], [virtual]
```

Reimplemented from ModelDataDefinition.

3.8.4.2 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.8.4.3 _createInternalAndAttachedData() `void Buffer::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.8.4.4 _getParserChangesInformation() `ParserChangesInformation * Buffer::_getParserChanges↩ Information ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.8.4.5 _initBetweenReplications() `void Buffer::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.8.4.6 _loadInstance() `bool Buffer::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.8.4.7 _onDispatchEvent()

```
void Buffer::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.8.4.8 _saveInstance()

```
void Buffer::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.8.4.9 getadvanceOn() `Buffer::AdvanceOn Buffer::getadvanceOn ( ) const`

3.8.4.10 getarrivalOnFullBufferRule() `Buffer::ArrivalOnFullBufferRule Buffer::getarrivalOnFull↩ BufferRule ( ) const`

3.8.4.11 getcapacity() `unsigned int Buffer::getcapacity ( ) const`

3.8.4.12 GetPluginInformation() `PluginInformation * Buffer::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.8.4.13 getsignal() `SignalData * Buffer::getsignal ( ) const`

3.8.4.14 LoadInstance() `ModelComponent * Buffer::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.8.4.15 NewInstance()

```
ModelDataDefinition * Buffer::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.8.4.16 setAdvanceOn()

```
void Buffer::setAdvanceOn (
    Buffer::AdvanceOn newAdvanceOn )
```

3.8.4.17 setArrivalOnFullBufferRule()

```
void Buffer::setArrivalOnFullBufferRule (
    Buffer::ArrivalOnFullBufferRule newArrivalOnFullBufferRule )
```

3.8.4.18 setCapacity()

```
void Buffer::setCapacity (
    unsigned int newCapacity )
```

3.8.4.19 setSignal()

```
void Buffer::setSignal (
    SignalData * newSignal )
```

3.8.4.20 show()

```
std::string Buffer::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.9 CellularAutomata Class Reference

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

Inheritance diagram for CellularAutomata:

Public Member Functions

- CellularAutomata (Model *model, std::string name="")

- virtual ~CellularAutomata ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.9.1 Constructor & Destructor Documentation

3.9.1.1 CellularAutomata()

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

3.9.1.2 ~CellularAutomata()

```
virtual CellularAutomata::~CellularAutomata ( ) [virtual], [default]
```

3.9.2 Member Function Documentation

3.9.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.9.2.2 _loadInstance()

```
bool CellularAutomata::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.9.2.3 _onDispatchEvent()

```
void CellularAutomata::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.9.2.4 _saveInstance()

```
void CellularAutomata::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.9.2.5 GetPluginInformation() `PluginInformation * CellularAutomata::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.9.2.6 LoadInstance() `ModelComponent * CellularAutomata::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.9.2.7 NewInstance() `ModelDataDefinition * CellularAutomata::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.9.2.8 show() `std::string CellularAutomata::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.10 Clone Class Reference

```
#include <Clone.h>
```

Inheritance diagram for Clone:

Public Member Functions

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

  virtual public methods
- void setNumClonesExpression (std::string numClones)
- std::string getNumClonesExpression () const

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.10.1 Detailed Description

This component ...

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Clone()

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

3.10.2.2 ∼Clone()

```
virtual Clone::~Clone ( ) [virtual], [default]
```

3.10.3 Member Function Documentation

3.10.3.1 _check() `bool Clone::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.10.3.2 _createInternalAndAttachedData() `void Clone::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.10.3.3 _initBetweenReplications() `void Clone::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.10.3.4 _loadInstance()

```
bool Clone::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.10.3.5 _onDispatchEvent()

```
void Clone::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.10.3.6 _saveInstance()

```
void Clone::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.10.3.7 getNumClonesExpression()

```
std::string Clone::getNumClonesExpression ( ) const
```

3.10.3.8 GetPluginInformation()

```
PluginInformation * Clone::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.10.3.9 LoadInstance()

```
ModelComponent * Clone::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.10.3.10 NewInstance()

```
ModelDataDefinition * Clone::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.10.3.11 setNumClonesExpression()

```
void Clone::setNumClonesExpression (
    std::string numClones )
```

3.10.3.12 show()

```
std::string Clone::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.11 Collector_if Class Reference

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

Inheritance diagram for Collector_if:

Public Member Functions

- virtual void clear ()=0

- virtual void addValue (double value, double weight=1)=0
- virtual double getLastValue ()=0
- virtual unsigned long numElements ()=0
- virtual void setAddValueHandler (CollectorAddValueHandler addValueHandler)=0
- virtual void setClearHandler (CollectorClearHandler clearHandler)=0

3.11.1 Detailed Description

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

3.11.2 Member Function Documentation

3.11.2.1 addValue()

```
virtual void Collector_if::addValue (
    double value,
    double weight = 1 ) [pure virtual]
```

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.11.2.2 clear()

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.11.2.3 getLastValue()

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.11.2.4 numElements()

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.11.2.5 setAddValueHandler()

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.11.2.6 setClearHandler()

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDefaultImpl1.

3.12 CollectorDatafile_if Class Reference

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

Inheritance 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

3.12.1 Detailed Description

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

3.12.2 Member Function Documentation

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

Implemented in CollectorDatafileDefaultImpl1.

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

Get the next value in the file and advances the pointer

Implemented in CollectorDatafileDefaultImpl1.

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

Get a value from a specific position

Implemented in CollectorDatafileDefaultImpl1.

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

Set the pointer to the first value in the file

Implemented in CollectorDatafileDefaultImpl1.

3.12.2.5 setDataFilename()

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

Implemented in CollectorDatafileDefaultImpl1.

3.13 CollectorDatafileDefaultImpl1 Class Reference

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

Inheritance diagram for CollectorDatafileDefaultImpl1:

Public Member Functions

- CollectorDatafileDefaultImpl1 ()
- virtual ~CollectorDatafileDefaultImpl1 ()=default
- virtual void clear () override
- virtual void addValue (double value, double weight=1) override
- virtual double getLastValue () override
- virtual unsigned long numElements () override
- virtual double getValue (unsigned int num) override
- virtual double getNextValue () override
- virtual void seekFirstValue () override
- virtual std::string getDataFilename () override
- virtual void setDataFilename (std::string filename) override
- virtual void setAddValueHandler (CollectorAddValueHandler addValueHandler) override
- virtual void setClearHandler (CollectorClearHandler clearHandler) override

3.13.1 Constructor & Destructor Documentation

3.13.1.1 CollectorDatafileDefaultImpl1() `CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ( )`

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

3.13.2 Member Function Documentation

3.13.2.1 addValue()

```
void CollectorDatafileDefaultImpl1::addValue (
    double value,
    double weight = 1 ) [override], [virtual]
```

Implements Collector_if.

3.13.2.2 clear() `void CollectorDatafileDefaultImpl1::clear ( ) [override], [virtual]`

Implements Collector_if.

3.13.2.3 getDataFilename() `std::string CollectorDatafileDefaultImpl1::getDataFilename ( ) [override], [virtual]`

Implements CollectorDatafile_if.

3.13.2.4 getLastValue() `double CollectorDatafileDefaultImpl1::getLastValue ( ) [override], [virtual]`

Implements Collector_if.

3.13.2.5 getNextValue() `double CollectorDatafileDefaultImpl1::getNextValue ( ) [override], [virtual]`

Get the next value in the file and advances the pointer

Implements CollectorDatafile_if.

3.13.2.6 getValue() `double CollectorDatafileDefaultImpl1::getValue ( unsigned int rank ) [override], [virtual]`

Get a value from a specific position

Implements CollectorDatafile_if.

3.13.2.7 numElements() `unsigned long CollectorDatafileDefaultImpl1::numElements ( ) [override], [virtual]`

Implements Collector_if.

3.13.2.8 seekFirstValue() `void CollectorDatafileDefaultImpl1::seekFirstValue ( ) [override], [virtual]`

Set the pointer to the first value in the file

Implements CollectorDatafile_if.

3.13.2.9 setAddValueHandler() `void CollectorDatafileDefaultImpl1::setAddValueHandler ( CollectorAddValueHandler addValueHandler ) [override], [virtual]`

Implements Collector_if.

3.13.2.10 setClearHandler() `void CollectorDatafileDefaultImpl1::setClearHandler ( CollectorClearHandler clearHandler ) [override], [virtual]`

Implements Collector_if.

3.13.2.11 setDataFilename() `void CollectorDatafileDefaultImpl1::setDataFilename ( std::string filename ) [override], [virtual]`

Implements CollectorDatafile_if.

3.14 CollectorDefaultImpl1 Class Reference

`#include <CollectorDefaultImpl1.h>`

Inheritance diagram for CollectorDefaultImpl1:

Public Member Functions

- CollectorDefaultImpl1 ()
- virtual ~CollectorDefaultImpl1 ()=default
- virtual void clear () override
- virtual void addValue (double value, double weight=1) override
- virtual double getLastValue () override
- virtual unsigned long numElements () override
- virtual void setAddValueHandler (CollectorAddValueHandler addValueHandler) override
- virtual void setClearHandler (CollectorClearHandler clearHandler) override

3.14.1 Constructor & Destructor Documentation

3.14.1.1 CollectorDefaultImpl1() `CollectorDefaultImpl1::CollectorDefaultImpl1 ( )`

3.14.1.2 ~CollectorDefaultImpl1() `virtual CollectorDefaultImpl1::~CollectorDefaultImpl1 ( ) [virtual], [default]`

3.14.2 Member Function Documentation

3.14.2.1 addValue()

```
void CollectorDefaultImpl1::addValue (
    double value,
    double weight = 1 ) [override], [virtual]
```

Implements Collector_if.

3.14.2.2 clear() `void CollectorDefaultImpl1::clear ( ) [override], [virtual]`

Implements Collector_if.

3.14.2.3 getLastValue() `double CollectorDefaultImpl1::getLastValue ( ) [override], [virtual]`

Implements Collector_if.

3.14.2.4 numElements() `unsigned long CollectorDefaultImpl1::numElements ( ) [override], [virtual]`

Implements Collector_if.

3.14.2.5 setAddValueHandler() `void CollectorDefaultImpl1::setAddValueHandler ( CollectorAddValueHandler addValueHandler ) [override], [virtual]`

Implements Collector_if.

3.14.2.6 setClearHandler() `void CollectorDefaultImpl1::setClearHandler ( CollectorClearHandler clearHandler ) [override], [virtual]`

Implements Collector_if.

3.15 CppCompiler::CompilationResult Class Reference

```
#include <CppCompiler.h>
```

Public Member Functions

- CompilationResult ()

Public Attributes

- bool success = false
- std::string compilationStdOutput = ""
- std::string compilationErrOutput = ""
- std::string destinationPath = ""
- unsigned int lineNumber = 0

3.15.1 Constructor & Destructor Documentation

3.15.1.1 CompilationResult() `CppCompiler::CompilationResult::CompilationResult ( ) [inline]`

3.15.2 Member Data Documentation

3.15.2.1 compilationErrOutput `std::string CppCompiler::CompilationResult::compilationErrOutput = ""`

3.15.2.2 compilationStdOutput `std::string CppCompiler::CompilationResult::compilationStdOutput = ""`

3.15.2.3 destinationPath `std::string CppCompiler::CompilationResult::destinationPath = ""`

3.15.2.4 lineNumber `unsigned int CppCompiler::CompilationResult::lineNumber = 0`

3.15.2.5 success `bool CppCompiler::CompilationResult::success = false`

3.16 CompilerThread Class Reference

```
#include <CppCompiler.h>
```

Public Member Functions

- void operator() ()

3.16.1 Member Function Documentation

3.16.1.1 operator()() `void CompilerThread::operator() ( )`

3.17 ComponentManager Class Reference

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

Public Member Functions

- ComponentManager (Model ∗model)
- virtual ∼ComponentManager ()=default
- bool insert (ModelComponent ∗comp)
- void remove (ModelComponent ∗comp)
- ModelComponent ∗ find (std::string name)
- ModelComponent ∗ find (Util::identification id)
- 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)
- std::list< SourceModelComponent ∗ > ∗ getSourceComponents ()
- std::list< ModelComponent ∗ > ∗ getTransferInComponents ()
- std::list< ModelComponent ∗ > ∗ getAllComponents () const

3.17.1 Detailed Description

ComponentManager allows to insert, access, find, remove and over Models

3.17.2 Constructor & Destructor Documentation

3.17.2.1 ComponentManager() `ComponentManager::ComponentManager ( Model ∗ model )`

Components are sorted by IDHere is the call graph for this function:

3.17.2.2 ∼ComponentManager() `virtual ComponentManager::∼ComponentManager ( ) [virtual], [default]`

3.17.3 Member Function Documentation

3.17.3.1 begin()

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

3.17.3.2 clear()

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

3.17.3.3 end()

```
std::list< ModelComponent * >::iterator ComponentManager::end ( )
```

3.17.3.4 find() [1/2]

```
ModelComponent * ComponentManager::find (
    std::string name )
```

3.17.3.5 find() [2/2]

```
ModelComponent * ComponentManager::find (
    Util::identification id )
```

3.17.3.6 front()

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

3.17.3.7 getAllComponents()

```
std::list< ModelComponent * > * ComponentManager::getAllComponents
( ) const
```

3.17.3.8 getNumberOfComponents()

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

3.17.3.9 getSourceComponents()

```
std::list< SourceModelComponent * > * ComponentManager::get↩
SourceComponents ( )
```

3.17.3.10 getTransferInComponents() `std::list< ModelComponent * > * ComponentManager::get↩ TransferInComponents ( )`

Here is the call graph for this function:

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

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

Here is the call graph for this function:

3.17.3.13 next() `ModelComponent * ComponentManager::next ( )`

3.17.3.14 remove()

```
void ComponentManager::remove (
    ModelComponent * comp )
```

Here is the call graph for this function:

3.17.3.15 setHasChanged()

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

3.18 Connection Struct Reference

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

Public Attributes

- ModelComponent * component
- ConnectionChannel channel

3.18.1 Detailed Description

Defines a connection between two or more ModelComponents. A (receiver) component's inputPort may receive connection from one or more (sender) components' outputPorts. A (sender) component's outputPort is connected to one and only one (receiver) component's inputPort.

3.18.2 Member Data Documentation

3.18.2.1 channel

```
ConnectionChannel Connection::channel
```

3.18.2.2 component `ModelComponent* Connection::component`

3.19 ConnectionChannel Struct Reference

`#include <ConnectionManager.h>`

Public Attributes

- unsigned int portNumber = 0
- std::string portDescription = ""

3.19.1 Member Data Documentation

3.19.1.1 portDescription `std::string ConnectionChannel::portDescription = ""`

3.19.1.2 portNumber `unsigned int ConnectionChannel::portNumber = 0`

3.20 ConnectionManager Class Reference

`#include <ConnectionManager.h>`

Public Member Functions

- ConnectionManager ()
- virtual ~ConnectionManager ()=default
- unsigned int size ()
- Connection * getFrontConnection ()
- Connection * getConnectionAtPort (unsigned int rank)
- void insert (ModelComponent *component, unsigned int inputPortNumber=0)
- void insert (Connection *connection)
- void insertAtPort (unsigned int port, Connection *connection)
- void remove (Connection *connection)
- void removeAtPort (unsigned int port)
- std::map< unsigned int, Connection * > * connections () const
- unsigned int getCurrentOutputConnectionsSize () const
- void setMaxOutputConnections (unsigned int _maxOutputConnections)
- unsigned int getMaxOutputConnections () const
- void setMinOutputConnections (unsigned int _minOutputConnections)
- unsigned int getMinOutputConnections () const
- unsigned int getCurrentInputConnectionsSize () const
- void setMaxInputConnections (unsigned int _maxInputConnections)
- unsigned int getMaxInputConnections () const
- void setMinInputConnections (unsigned int _minInputConnections)
- unsigned int getMinInputConnections () const

3.20.1 Detailed Description

ConnectionManager defines how a ModelComponent is output connected to none, one or more following Model↩ Components. It has a list of nextConnections, where each Connection is a pair, defining the next ModelComponent and an input port on that component (usefull only if the next component has more than one input). The number of the output conection is its rank in the nextConnections list. Min and max number of input and output connectons can be defined.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 ConnectionManager() `ConnectionManager::ConnectionManager ( )`

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

3.20.3 Member Function Documentation

3.20.3.1 connections() `std::map< unsigned int, Connection * > * ConnectionManager::connections ( ) const`

3.20.3.2 getConnectionAtPort() `Connection * ConnectionManager::getConnectionAtPort ( unsigned int rank )`

3.20.3.3 getCurrentInputConnectionsSize() `unsigned int ConnectionManager::getCurrentInput↩ ConnectionsSize ( ) const`

3.20.3.4 getCurrentOutputConnectionsSize() `unsigned int ConnectionManager::getCurrentOutput↩ ConnectionsSize ( ) const`

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

Here is the call graph for this function:

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

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

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

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

3.20.3.10 insert() [1/2] `void ConnectionManager::insert ( Connection * connection )`

Here is the call graph for this function:

3.20.3.11 insert() [2/2]

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

Insert a new connection by specifying the destnation component and optionally an input port, if not 0 Here is the call graph for this function:

3.20.3.12 insertAtPort()

```
void ConnectionManager::insertAtPort (
    unsigned int port,
    Connection * connection )
```

3.20.3.13 remove()

```
void ConnectionManager::remove (
    Connection * connection )
```

3.20.3.14 removeAtPort()

```
void ConnectionManager::removeAtPort (
    unsigned int port )
```

3.20.3.15 setMaxInputConnections()

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

3.20.3.16 setMaxOutputConnections()

```
void ConnectionManager::setMaxOutputConnections (
    unsigned int _maxOutputConnections )
```

3.20.3.17 setMinInputConnections()

```
void ConnectionManager::setMinInputConnections (
    unsigned int _minInputConnections )
```

3.20.3.18 setMinOutputConnections()

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

3.20.3.19 size()

```
unsigned int ConnectionManager::size ( )
```

3.21 Counter Class Reference

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

Inheritance diagram for Counter:

Public Member Functions

- Counter (Model ∗model, std::string name="", ModelDataDefinition ∗parent=nullptr)
- virtual ∼Counter ()=default
- virtual std::string show ()
- void clear ()
- void incCountValue (double value=1.0)
- double getCountValue () const
- ModelDataDefinition ∗ getParent () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()

Additional Inherited Members

3.21.1 Detailed Description

The Counter modeldatum is used to count events, and its internal count value is added by a configurable amount, usually incremented by one.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Counter()

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

Here is the call graph for this function:

3.21.2.2 ~Counter() `virtual Counter::~Counter ( ) [virtual], [default]`

3.21.3 Member Function Documentation

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.21.3.2 _initBetweenReplications() `void Counter::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.21.3.3 _loadInstance() `bool Counter::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.21.3.4 _saveInstance() `void Counter::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.21.3.5 clear() `void Counter::clear ( )`

3.21.3.6 getCountValue() `double Counter::getCountValue ( ) const`

3.21.3.7 getParent() `ModelDataDefinition * Counter::getParent ( ) const`

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

Here is the call graph for this function:

3.21.3.9 incCountValue() `void Counter::incCountValue ( double value = 1.0 )`

3.21.3.10 LoadInstance()

```
ModelDataDefinition * Counter::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.21.3.11 NewInstance()

```
ModelDataDefinition * Counter::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.21.3.12 show()

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

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.22 CppCompiler Class Reference

```
#include <CppCompiler.h>
```

Inheritance diagram for CppCompiler:

Classes

- class CompilationResult

Public Member Functions

- CppCompiler (Model *model, std::string name="")
- virtual ~CppCompiler ()=default
- virtual std::string show ()
- void setSourceFilename (std::string sourceFilename)
- std::string getSourceFilename () const
- bool IsLibraryLoaded () const
- void setFlagsExecutable (std::string _flagsExecutable)
- std::string getFlagsExecutable () const
- void setLibraryLoaded (bool libraryLoaded)
- void setFlagsStaticLibrary (std::string _flagsStaticLibrary)
- std::string getFlagsStaticLibrary () const
- void setFlagsDynamicLibrary (std::string _flagsDynamicLibrary)
- std::string getFlagsDynamicLibrary () const
- void setFlagsGeneral (std::string _flagsGeneral)
- std::string getFlagsGeneral () const
- void setCompilerCommand (std::string _compilerCommand)
- std::string getCompilerCommand () const
- void setOutputDir (std::string _outputDir)
- std::string getOutputDir () const
- void setTempDir (std::string _tempDir)
- std::string getTempDir () const
- void setOutputFilename (std::string _outputFilename)
- std::string getOutputFilename () const
- void setObjectFiles (std::string _objectFiles)
- std::string getObjectFiles () const
- CompilationResult compileToExecutable ()
- CompilationResult compileToDynamicLibrary ()
- CompilationResult compileToStaticLibrary ()
- bool loadLibrary (std::string *errorMessage)
- bool unloadLibrary ()
- void * getDynamicLibraryHandler () const

Static Public Member Functions

- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- CompilationResult _invokeCompiler (std::string command)
- std::string _read (std::string filename)

Additional Inherited Members

3.22.1 Constructor & Destructor Documentation

3.22.1.1 CppCompiler()

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

3.22.1.2 ~CppCompiler()

```
virtual CppCompiler::~CppCompiler ( ) [virtual], [default]
```

3.22.2 Member Function Documentation

3.22.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.22.2.2 _createInternalAndAttachedData()

```
void CppCompiler::_createInternalAndAttachedData ( )
[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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

3.22.2.3 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

3.22.2.4 _invokeCompiler()

```
CppCompiler::CompilationResult CppCompiler::_invokeCompiler (
    std::string command ) [protected]
```

Here is the call graph for this function:

3.22.2.5 _loadInstance()

```
bool CppCompiler::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.22.2.6 _read()

```
std::string CppCompiler::_read (
    std::string filename ) [protected]
```

3.22.2.7 _saveInstance()

```
void CppCompiler::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.22.2.8 compileToDynamicLibrary() `CppCompiler::CompilationResult CppCompiler::compileTo↩DynamicLibrary ( )`

Here is the call graph for this function:

3.22.2.9 compileToExecutable() `CppCompiler::CompilationResult CppCompiler::compileToExecutable ( )`

Here is the call graph for this function:

3.22.2.10 compileToStaticLibrary() `CppCompiler::CompilationResult CppCompiler::compileToStaticLibrary ( )`

Here is the call graph for this function:

3.22.2.11 getCompilerCommand() `std::string CppCompiler::getCompilerCommand ( ) const`

3.22.2.12 getDynamicLibraryHandler() `void * CppCompiler::getDynamicLibraryHandler ( ) const`

3.22.2.13 getFlagsDynamicLibrary() `std::string CppCompiler::getFlagsDynamicLibrary ( ) const`

3.22.2.14 getFlagsExecutable() `std::string CppCompiler::getFlagsExecutable ( ) const`

3.22.2.15 getFlagsGeneral() `std::string CppCompiler::getFlagsGeneral ( ) const`

3.22.2.16 getFlagsStaticLibrary() `std::string CppCompiler::getFlagsStaticLibrary ( ) const`

3.22.2.17 getObjectFiles() `std::string CppCompiler::getObjectFiles ( ) const`

3.22.2.18 getOutputDir() `std::string CppCompiler::getOutputDir ( ) const`

3.22.2.19 getOutputFilename() `std::string CppCompiler::getOutputFilename ( ) const`

3.22.2.20 GetPluginInformation() `PluginInformation * CppCompiler::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.22.2.21 getSourceFilename() `std::string CppCompiler::getSourceFilename ( ) const`

3.22.2.22 getTempDir() `std::string CppCompiler::getTempDir ( ) const`

3.22.2.23 IsLibraryLoaded() `bool CppCompiler::IsLibraryLoaded ( ) const`

3.22.2.24 LoadInstance() `ModelDataDefinition * CppCompiler::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.22.2.25 loadLibrary()

```
bool CppCompiler::loadLibrary (
    std::string * errorMessage )
```

3.22.2.26 NewInstance()

```
ModelDataDefinition * CppCompiler::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.22.2.27 setCompilerCommand()

```
void CppCompiler::setCompilerCommand (
    std::string _compilerCommand )
```

3.22.2.28 setFlagsDynamicLibrary()

```
void CppCompiler::setFlagsDynamicLibrary (
    std::string _flagsDynamicLibrary )
```

3.22.2.29 setFlagsExecutable()

```
void CppCompiler::setFlagsExecutable (
    std::string _flagsExecutable )
```

3.22.2.30 setFlagsGeneral()

```
void CppCompiler::setFlagsGeneral (
    std::string _flagsGeneral )
```

3.22.2.31 setFlagsStaticLibrary()

```
void CppCompiler::setFlagsStaticLibrary (
    std::string _flagsStaticLibrary )
```

3.22.2.32 setLibraryLoaded()

```
void CppCompiler::setLibraryLoaded (
    bool libraryLoaded )
```

3.22.2.33 setObjectFiles()

```
void CppCompiler::setObjectFiles (
    std::string _objectFiles )
```

3.22.2.34 setOutputDir()

```
void CppCompiler::setOutputDir (
    std::string _outputDir )
```

3.22.2.35 setOutputFilename()

```
void CppCompiler::setOutputFilename (
    std::string _outputFilename )
```

3.22.2.36 setSourceFilename()

```
void CppCompiler::setSourceFilename (
    std::string sourceFilename )
```

3.22.2.37 setTempDir()

```
void CppCompiler::setTempDir (
    std::string _tempDir )
```

3.22.2.38 show()

```
std::string CppCompiler::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.22.2.39 unloadLibrary()

```
bool CppCompiler::unloadLibrary ( )
```

3.23 CppForG Class Reference

```
#include <CppForG.h>
```

Inheritance diagram for CppForG:

Public Member Functions

- CppForG (Model *model, std::string name="")

- virtual ~CppForG ()=default
- virtual std::string show ()

  virtual public methods
- void setCppCompiler (CppCompiler *_cppCompiler)
- CppCompiler * getCppCompiler () const
- void setInitBetweenReplicationCode (std::string _initBetweenReplicationCode)
- std::string getInitBetweenReplicationCode () const
- void setOnDispatchEventCode (std::string _onDispatchEventCode)
- std::string getOnDispatchEventCode () const
- void setIncludesCode (std::string _includesCode)
- std::string getIncludesCode () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.23.1 Detailed Description

This component ...

3.23.2 Constructor & Destructor Documentation

3.23.2.1 CppForG()

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

Here is the call graph for this function:

3.23.2.2 ~CppForG()

```
virtual CppForG::~CppForG ( ) [virtual], [default]
```

3.23.3 Member Function Documentation

3.23.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.23.3.2 _createInternalAndAttachedData() `void CppForG::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.23.3.3 _initBetweenReplications() `void CppForG::_initBetweenReplications ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.23.3.4 _loadInstance()

```
bool CppForG::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.23.3.5 _onDispatchEvent()

```
void CppForG::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.23.3.6 _saveInstance()

```
void CppForG::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.23.3.7 getCppCompiler() `CppCompiler * CppForG::getCppCompiler ( ) const`

3.23.3.8 getIncludesCode() `std::string CppForG::getIncludesCode ( ) const`

3.23.3.9 getInitBetweenReplicationCode() `std::string CppForG::getInitBetweenReplicationCode ( ) const`

3.23.3.10 getOnDispatchEventCode() `std::string CppForG::getOnDispatchEventCode ( ) const`

3.23.3.11 GetPluginInformation() `PluginInformation * CppForG::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.23.3.12 LoadInstance()

```
ModelComponent * CppForG::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.23.3.13 NewInstance()

```
ModelDataDefinition * CppForG::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.23.3.14 setCppCompiler()

```
void CppForG::setCppCompiler (
    CppCompiler * _cppCompiler )
```

3.23.3.15 setIncludesCode()

```
void CppForG::setIncludesCode (
    std::string _includesCode )
```

3.23.3.16 setInitBetweenReplicationCode()

```
void CppForG::setInitBetweenReplicationCode (
    std::string _initBetweenReplicationCode )
```

3.23.3.17 setOnDispatchEventCode()

```
void CppForG::setOnDispatchEventCode (
    std::string _onDispatchEventCode )
```

3.23.3.18 show()

```
std::string CppForG::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.24 CppSerializer Class Reference

```
#include <CppSerializer.h>
```

Inheritance diagram for CppSerializer:

Public Member Functions

- CppSerializer (Model *model)
- PersistenceRecord * newPersistenceRecord () override

  Creates an empty persistence record.
- bool dump (std::ostream &output) override

  Writes formatted contents to a given output stream.
- bool load (std::istream &input) override

  Attempts to load contents from a given input stream.
- bool get (const std::string &name, PersistenceRecord *entry) override

  Tries to fetch an entry with the given label as a key.
- bool put (const std::string name, const std::string type, const Util::identification id, PersistenceRecord *fields) override

  Upserts an entry into this structure.
- int for_each (std::function< int(const std::string &) > delegate) override

  Iterates over this structure's entries, ordered by ID.

3.24.1 Constructor & Destructor Documentation

3.24.1.1 CppSerializer()

```
CppSerializer::CppSerializer (
    Model * model ) [explicit]
```

3.24.2 Member Function Documentation

3.24.2.1 dump()

```
bool CppSerializer::dump (
    std::ostream & output ) [override], [virtual]
```

Writes formatted contents to a given output stream.

Parameters

output	output stream to write to.

Returns

whether the current state of this structure's contents was deemed valid for serialization.

Implements ModelSerializer.

Here is the call graph for this function:

3.24.2.2 for_each()

```
int CppSerializer::for_each (
    std::function< int(const std::string &) > delegate ) [override], [virtual]
```

Iterates over this structure's entries, ordered by ID.

Parameters

delegate	delegate which is applied to every entry-identifying key, until it returns a non-zero value.

Returns

either zero, or the first non-zero value returned by the delegate.

Implements ModelSerializer.

3.24.2.3 get()

```
bool CppSerializer::get (
    const std::string & name,
    PersistenceRecord * entry ) [override], [virtual]
```

Tries to fetch an entry with the given label as a key.

Parameters

name	a label which uniquely identifies an entry.
entry	(out-param) a pointer to a caller-owned entry, which is filled on success.

Returns

whether the entry was found and the out-param was actually written to.

Implements ModelSerializer.

Here is the call graph for this function:

3.24.2.4 load()

```
bool CppSerializer::load (
    std::istream & input ) [override], [virtual]
```

Attempts to load contents from a given input stream.

Returns

whether the input was well-formatted and loading was successful.

Implements ModelSerializer.

3.24.2.5 newPersistenceRecord() `PersistenceRecord * CppSerializer::newPersistenceRecord ( ) [override], [virtual]`

Creates an empty persistence record.

Although the returned record can be freed normally with `delete`, care must be taken to only use records which come from a implementation, when configuring that implementation's contents.

Implements ModelSerializer.

Here is the call graph for this function:

3.24.2.6 put()

```
bool CppSerializer::put (
    const std::string name,
    const std::string type,
    const Util::identification id,
    PersistenceRecord * fields ) [override], [virtual]
```

Upserts an entry into this structure.

Parameters

name	a label which uniquely identifies the entry.
type	a type name.
id	a component/datadef ID number.
fields	a persistence record for the entry, whose contents will be copied if valid.

Returns

whether the entry was valid and was actually copied into this structure.

Implements ModelSerializer.

Here is the call graph for this function:

3.25 Create Class Reference

`#include <Create.h>`

Inheritance diagram for Create:

Public Member Functions

- Create (Model ∗model, std::string name="")

- virtual ~Create ()=default
- virtual std::string show ()

  virtual public methods
- void setTimeBetweenCreationsFormula (Formula *_timeBetweenCreationsFormula)
- Formula * getTimeBetweenCreationsFormula () const
- void setTimeBetweenCreationsSchedule (Schedule *_timeBetweenCreationsSchedule)
- Schedule * getTimeBetweenCreationsSchedule () const
- double testePropertyCreateDouble () const
- void setTestePropertyCreateDouble (double newTestePropertyCreateDouble)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _initBetweenReplications ()
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.25.1 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. TYPICAL 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.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 Create()

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

3.25.2.2 ~Create()

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

3.25.3 Member Function Documentation

3.25.3.1 _check()

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

could be overriden by derived classes

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

3.25.3.2 _createInternalAndAttachedData()

```
void Create::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

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

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

3.25.3.4 _loadInstance() `bool Create::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

3.25.3.5 _onDispatchEvent() `void Create::_onDispatchEvent ( Entity * entity, unsigned int inputPortNumber ) [protected], [virtual]`

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.25.3.6 _saveInstance()

```
void Create::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

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

Here is the call graph for this function:

3.25.3.8 getTimeBetweenCreationsFormula() `Formula * Create::getTimeBetweenCreationsFormula ( ) const`

3.25.3.9 getTimeBetweenCreationsSchedule() `Schedule * Create::getTimeBetweenCreationsSchedule ( ) const`

3.25.3.10 LoadInstance() `ModelComponent * Create::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.25.3.11 NewInstance()

```
ModelDataDefinition * Create::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.25.3.12 setTestePropertyCreateDouble()

```
void Create::setTestePropertyCreateDouble (
    double newTestePropertyCreateDouble )
```

3.25.3.13 setTimeBetweenCreationsFormula()

```
void Create::setTimeBetweenCreationsFormula (
    Formula * _timeBetweenCreationsFormula )
```

3.25.3.14 setTimeBetweenCreationsSchedule()

```
void Create::setTimeBetweenCreationsSchedule (
    Schedule * _timeBetweenCreationsSchedule )
```

3.25.3.15 show()

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

virtual public methods

Reimplemented from SourceModelComponent.

Here is the call graph for this function:

3.25.3.16 testePropertyCreateDouble() `double Create::testePropertyCreateDouble ( ) const`

3.26 DataAnalyser_if Class Reference

`#include <DataAnalyser_if.h>`

Public Member Functions

- virtual bool loadDataSet (std::string datafilename)=0
- virtual bool saveDataSet (std::string datasetname)=0
- virtual void newDataSet (std::string datasetname, std::string datafilename)=0
- virtual Fitter_if * fitter ()=0
- virtual Sampler_if * sampler ()=0
- virtual ExperimentManager_if * experimenter ()=0
- virtual HypothesisTester_if * tester ()=0

3.26.1 Member Function Documentation

3.26.1.1 experimenter() `virtual ExperimentManager_if * DataAnalyser_if::experimenter ( ) [pure virtual]`

3.26.1.2 fitter() `virtual Fitter_if * DataAnalyser_if::fitter ( ) [pure virtual]`

3.26.1.3 loadDataSet()

```
virtual bool DataAnalyser_if::loadDataSet (
    std::string datafilename ) [pure virtual]
```

3.26.1.4 newDataSet()

```
virtual void DataAnalyser_if::newDataSet (
    std::string datasetname,
    std::string datafilename ) [pure virtual]
```

3.26.1.5 sampler() `virtual Sampler_if * DataAnalyser_if::sampler ( ) [pure virtual]`

3.26.1.6 saveDataSet()

```
virtual bool DataAnalyser_if::saveDataSet (
    std::string datasetname ) [pure virtual]
```

3.26.1.7 tester()

```
virtual HypothesisTester_if * DataAnalyser_if::tester ( ) [pure virtual]
```

3.27 Decide Class Reference

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

Inheritance diagram for Decide:

Public Member Functions

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

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.27.1 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).

3.27.2 Constructor & Destructor Documentation

3.27.2.1 Decide()

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

3.27.2.2 ~Decide()

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

3.27.3 Member Function Documentation

3.27.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.27.3.2 _createInternalAndAttachedData()

```
void Decide::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.27.3.3 _loadInstance()

```
bool Decide::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.27.3.4 _onDispatchEvent()

```
void Decide::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.27.3.5 _saveInstance()

```
void Decide::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

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

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

Here is the call graph for this function:

3.27.3.8 LoadInstance() `ModelComponent * Decide::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.27.3.9 NewInstance()

```
ModelDataDefinition * Decide::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.27.3.10 show()

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

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.28 Delay::DEFAULT_VALUES Struct Reference

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

Public Attributes

- const std::string delayExpression = "1.0"
- const Util::TimeUnit delayTimeUnit = Util::TimeUnit::second
- const Util::AllocationType allocation = Util::AllocationType::Wait

3.28.1 Member Data Documentation

3.28.1.1 allocation

```
const Util::AllocationType Delay::DEFAULT_VALUES::allocation = Util::AllocationType::Wait
```

3.28.1.2 delayExpression `const std::string Delay::DEFAULT_VALUES::delayExpression = "1.0"`

3.28.1.3 delayTimeUnit `const Util::TimeUnit Delay::DEFAULT_VALUES::delayTimeUnit = Util::TimeUnit::second`

3.29 Seize::DEFAULT_VALUES Struct Reference

```
#include <Seize.h>
```

Public Attributes

- const Util::AllocationType allocationType = Util::AllocationType::Others
- const unsigned short priority = 0
- const std::string priorityExpression = ""
- const unsigned int seizeRequestSize = 1

3.29.1 Member Data Documentation

3.29.1.1 allocationType `const Util::AllocationType Seize::DEFAULT_VALUES::allocationType = Util::AllocationType::Others`

3.29.1.2 priority `const unsigned short Seize::DEFAULT_VALUES::priority = 0`

3.29.1.3 priorityExpression `const std::string Seize::DEFAULT_VALUES::priorityExpression = ""`

3.29.1.4 seizeRequestSize `const unsigned int Seize::DEFAULT_VALUES::seizeRequestSize = 1`

3.30 SourceModelComponent::DEFAULT_VALUES Struct Reference

```
#include <SourceModelComponent.h>
```

Public Attributes

- const double firstCreation = 0.0
- const unsigned int entitiesPerCreation = 1
- const std::string maxCreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max())
- const std::string timeBetweenCreationsExpression = "EXPO(1.0)"
- const Util::TimeUnit timeBetweenCreationsTimeUnit = Util::TimeUnit::second
- const std::string entityTypename = "entitytype"

3.30.1 Member Data Documentation

3.30.1.1 entitiesPerCreation

```
const unsigned int SourceModelComponent::DEFAULT_VALUES::entities↩
PerCreation = 1
```

3.30.1.2 entityTypename

```
const std::string SourceModelComponent::DEFAULT_VALUES::entityTypename
= "entitytype"
```

3.30.1.3 firstCreation

```
const double SourceModelComponent::DEFAULT_VALUES::firstCreation = 0.0
```

3.30.1.4 maxCreationsExpression

```
const std::string SourceModelComponent::DEFAULT_VALUES::max↩
CreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max())
```

3.30.1.5 timeBetweenCreationsExpression

```
const std::string SourceModelComponent::DEFAULT_↩
VALUES::timeBetweenCreationsExpression = "EXPO(1.0)"
```

3.30.1.6 timeBetweenCreationsTimeUnit

```
const Util::TimeUnit SourceModelComponent::DEFAULT_↩
VALUES::timeBetweenCreationsTimeUnit = Util::TimeUnit::second
```

3.31 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference

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

Inheritance diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:

Public Member Functions

- ~DefaultImpl1RNG_Parameters ()=default

Public Attributes

- uint32_t seed = 16021974
- uint32_t a = 279470273u
- uint32_t m = 0xfffffffb
- uint32_t c = 0

3.31.1 Constructor & Destructor Documentation

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

3.31.2 Member Data Documentation

3.31.2.1 a `uint32_t SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::a = 279470273u`

3.31.2.2 c `uint32_t SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::c = 0`

3.31.2.3 m `uint32_t SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::m = 0xfffffffb`

3.31.2.4 seed `uint32_t SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::seed = 16021974`

3.32 Delay Class Reference

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

Inheritance diagram for Delay:

Classes

- struct DEFAULT_VALUES

Public Member Functions

- Delay (Model ∗model, std::string name="")

- virtual ~Delay ()=default
- void setDelayExpression (std::string _delayExpression, Util::TimeUnit _delayTimeUnit=Util::TimeUnit::unknown)
- std::string delayExpression () const
- void setDelay (double delay)
- double delay () const
- void setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)
- Util::TimeUnit delayTimeUnit () const
- void setAllocation (Util::AllocationType allocation)
- Util::AllocationType getAllocation () const
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Public Attributes

- const struct Delay::DEFAULT_VALUES DEFAULT

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.32.1 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.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 Delay()

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

Here is the call graph for this function:

3.32.2.2 ~Delay()

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

3.32.3 Member Function Documentation

3.32.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.32.3.2 _createInternalAndAttachedData() `void Delay::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.32.3.3 _loadInstance() `bool Delay::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.32.3.4 _onDispatchEvent()

```
void Delay::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.32.3.5 _saveInstance()

```
void Delay::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.32.3.6 delay() `double Delay::delay ( ) const`

Here is the call graph for this function:

3.32.3.7 delayExpression() `std::string Delay::delayExpression ( ) const`

3.32.3.8 delayTimeUnit() `Util::TimeUnit Delay::delayTimeUnit ( ) const`

3.32.3.9 getAllocation() `Util::AllocationType Delay::getAllocation ( ) const`

3.32.3.10 GetPluginInformation() `PluginInformation * Delay::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.32.3.11 LoadInstance() `ModelComponent * Delay::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.32.3.12 NewInstance() `ModelDataDefinition * Delay::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.32.3.13 setAllocation()

```
void Delay::setAllocation (
    Util::AllocationType allocation )
```

3.32.3.14 setDelay()

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

Here is the call graph for this function:

3.32.3.15 setDelayExpression()

```
void Delay::setDelayExpression (
    std::string _delayExpression,
    Util::TimeUnit _delayTimeUnit = Util::TimeUnit::unknown )
```

3.32.3.16 setDelayTimeUnit()

```
void Delay::setDelayTimeUnit (
    Util::TimeUnit _delayTimeUnit )
```

3.32.3.17 show()

```
std::string Delay::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.32.4 Member Data Documentation

3.32.4.1 DEFAULT `const struct Delay::DEFAULT_VALUES Delay::DEFAULT`

3.33 DiffEquations Class Reference

`#include <DiffEquations.h>`

Inheritance diagram for DiffEquations:

Public Member Functions

- DiffEquations (Model *model, std::string name="")

  constructors
- virtual ~DiffEquations ()=default
- List< std::string > * getEquations () const

  new public user methods for this component
- double getprecision () const
- void setPrecision (double newPrecision)
- double getmaxSteps () const
- void setMaxSteps (double newMaxSteps)
- double getfinalTime () const
- void setFinalTime (double newFinalTime)
- List< double > * getvaluesOnFinalTime () const
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()

  static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  This method is only for ModelComponents, not ModelDataElements.
- virtual bool _check (std::string *errorMessage)

  virtual protected methods that could be overriden by derived classes, if needed
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.33.1 Detailed Description

This component ...

3.33.2 Constructor & Destructor Documentation

3.33.2.1 DiffEquations()

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

constructors

Externalize function GetPluginInformation to be accessible throught dynamic linked library.

3.33.2.2 ~DiffEquations()

```
virtual DiffEquations::~DiffEquations ( ) [virtual], [default]
```

3.33.3 Member Function Documentation

3.33.3.1 _check()

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

virtual protected methods that could be overriden by derived classes, if needed

This method is called by ModelChecker during model check. The component should check itself to verify if user parameters are ok (ex: correct syntax for the parser) and everithing in its parameters allow the model too run without errors in this component

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.33.3.2 _createInternalAndAttachedData() `void DiffEquations::_createInternalAndAttachedData ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

This method is called by ModelSimulation when initianting the replication. The model should set all value for a new replication (Ex: setting back to 0 any internal counter, clearing lists, etc.

This method is called by ModelChecker and 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(This↩CSTAT)") and therefore the modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.33.3.3 _loadInstance() `bool DiffEquations::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.33.3.4 _onDispatchEvent()

```
void DiffEquations::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

This method is only for ModelComponents, not ModelDataElements.

Implements ModelComponent.

Here is the call graph for this function:

3.33.3.5 _saveInstance()

```
void DiffEquations::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.33.3.6 getEquations() `List< std::string > * DiffEquations::getEquations ( ) const`

new public user methods for this component

equations should be in terms of x[i] ; f[i](x[],t)

3.33.3.7 getfinalTime() `double DiffEquations::getfinalTime ( ) const`

3.33.3.8 getmaxSteps() `double DiffEquations::getmaxSteps ( ) const`

3.33.3.9 GetPluginInformation() `PluginInformation * DiffEquations::GetPluginInformation ( ) [static]`

static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component

Here is the call graph for this function:

3.33.3.10 getprecision() `double DiffEquations::getprecision ( ) const`

3.33.3.11 getvaluesOnFinalTime() `List< double > * DiffEquations::getvaluesOnFinalTime ( ) const`

3.33.3.12 LoadInstance() `ModelComponent * DiffEquations::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.33.3.13 NewInstance()

```
ModelDataDefinition * DiffEquations::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.33.3.14 setFinalTime()

```
void DiffEquations::setFinalTime (
    double newFinalTime )
```

3.33.3.15 setMaxSteps()

```
void DiffEquations::setMaxSteps (
    double newMaxSteps )
```

3.33.3.16 setPrecision()

```
void DiffEquations::setPrecision (
    double newPrecision )
```

3.33.3.17 show()

```
std::string DiffEquations::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.34 Dispose Class Reference

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

Inheritance diagram for Dispose:

Public Member Functions

- Dispose (Model *model, std::string name="")

- virtual ~Dispose ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _initBetweenReplications ()
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.34.1 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.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 Dispose()

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

Here is the call graph for this function:

3.34.2.2 ~Dispose() `virtual Dispose::~Dispose ( ) [virtual], [default]`

3.34.3 Member Function Documentation

3.34.3.1 _check() `bool Dispose::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from SinkModelComponent.

3.34.3.2 _createInternalAndAttachedData() `void Dispose::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.34.3.3 _initBetweenReplications() `void Dispose::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.34.3.4 _loadInstance()

```
bool Dispose::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from SinkModelComponent.

Here is the call graph for this function:

3.34.3.5 _onDispatchEvent()

```
void Dispose::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.34.3.6 _saveInstance()

```
void Dispose::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from SinkModelComponent.

Here is the call graph for this function:

3.34.3.7 GetPluginInformation() `PluginInformation * Dispose::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.34.3.8 LoadInstance() `ModelComponent * Dispose::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.34.3.9 NewInstance()

```
ModelDataDefinition * Dispose::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.34.3.10 show()

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

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.35 DropOff Class Reference

```
#include <DropOff.h>
```

Inheritance diagram for DropOff:

Public Member Functions

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

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.35.1 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

3.35.2 Constructor & Destructor Documentation

3.35.2.1 DropOff()

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

3.35.2.2 ∼DropOff()

```
virtual DropOff::~DropOff ( ) [virtual], [default]
```

3.35.3 Member Function Documentation

3.35.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.35.3.2 _loadInstance()

```
bool DropOff::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.35.3.3 _onDispatchEvent()

```
void DropOff::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.35.3.4 _saveInstance()

```
void DropOff::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.35.3.5 GetPluginInformation() `PluginInformation * DropOff::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.35.3.6 LoadInstance() `ModelComponent * DropOff::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.35.3.7 NewInstance()

```
ModelDataDefinition * DropOff::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.35.3.8 show()

```
std::string DropOff::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.36 DummyComponent Class Reference

```
#include <DummyComponent.h>
```

Inheritance diagram for DummyComponent:

Public Member Functions

- DummyComponent (Model *model, std::string name="")

  constructors
- virtual ~DummyComponent ()=default
- virtual std::string show ()

  new public user methods for this component

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()

  static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  This method is only for ModelComponents, not ModelDataElements.

Additional Inherited Members

3.36.1 Detailed Description

This component ...

3.36.2 Constructor & Destructor Documentation

3.36.2.1 DummyComponent()

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

constructors

Externalize function GetPluginInformation to be accessible throught dynamic linked library.

3.36.2.2 ∼DummyComponent()

```
virtual DummyComponent::~DummyComponent ( ) [virtual], [default]
```

3.36.3 Member Function Documentation

3.36.3.1 _loadInstance()

```
bool DummyComponent::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.36.3.2 _onDispatchEvent()

```
void DummyComponent::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

This method is only for ModelComponents, not ModelDataElements.

Implements ModelComponent.

Here is the call graph for this function:

3.36.3.3 _saveInstance()

```
void DummyComponent::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.36.3.4 GetPluginInformation() `PluginInformation * DummyComponent::GetPluginInformation ( ) [static]`

static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component

Here is the call graph for this function:

3.36.3.5 LoadInstance() `ModelComponent * DummyComponent::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.36.3.6 NewInstance()

```
ModelDataDefinition * DummyComponent::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.36.3.7 show()

```
std::string DummyComponent::show ( )  [virtual]
```

new public user methods for this component

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.37 DummyElement Class Reference

```
#include <DummyElement.h>
```

Inheritance diagram for DummyElement:

Public Member Functions

- DummyElement (Model *model, std::string name="")

  constructors
- virtual ~DummyElement ()=default
- virtual std::string show ()

  new public user methods for this component

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()

  static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component
- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.

Additional Inherited Members

3.37.1 Constructor & Destructor Documentation

3.37.1.1 DummyElement()

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

constructors

3.37.1.2 ∼DummyElement()

```
virtual DummyElement::~DummyElement ( ) [virtual], [default]
```

3.37.2 Member Function Documentation

3.37.2.1 _loadInstance()

```
bool DummyElement::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.37.2.2 _saveInstance()

```
void DummyElement::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.37.2.3 GetPluginInformation()

```
PluginInformation * DummyElement::GetPluginInformation ( ) [static]
```

static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component

Here is the call graph for this function:

3.37.2.4 LoadInstance()

```
ModelDataDefinition * DummyElement::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.37.2.5 NewInstance()

```
ModelDataDefinition * DummyElement::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.37.2.6 show()

```
std::string DummyElement::show ( )  [virtual]
```

new public user methods for this component

virtual public methods

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.38 Enter Class Reference

```
#include <Enter.h>
```

Inheritance diagram for Enter:

Public Member Functions

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

  virtual public methods
- void setStation (Station *_station)

- void setStationName (std::string stationName)
- Station ∗ getStation () const

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.38.1 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.

3.38.2 Constructor & Destructor Documentation

3.38.2.1 Enter()

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

3.38.2.2 ~Enter()

```
virtual Enter::~Enter ( ) [virtual], [default]
```

3.38.3 Member Function Documentation

3.38.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.38.3.2 _createInternalAndAttachedData() `void Enter::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.38.3.3 _loadInstance() `bool Enter::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.38.3.4 _onDispatchEvent()

```
void Enter::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.38.3.5 _saveInstance()

```
void Enter::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.38.3.6 GetPluginInformation() `PluginInformation * Enter::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.38.3.7 getStation() `Station * Enter::getStation ( ) const`

3.38.3.8 LoadInstance() `ModelComponent * Enter::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.38.3.9 NewInstance()

```
ModelDataDefinition * Enter::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.38.3.10 setStation()

```
void Enter::setStation (
    Station * _station )
```

Here is the call graph for this function:

3.38.3.11 setStationName()

```
void Enter::setStationName (
    std::string stationName )
```

Here is the call graph for this function:

3.38.3.12 show() `std::string Enter::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.39 Entity Class Reference

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

Inheritance diagram for Entity:

Public Member Functions

- virtual std::string show ()
- void setEntityTypeName (std::string entityTypeName)
- std::string getEntityTypeName () 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, bool createIfNotFound=false)
- void setAttributeValue (std::string index, std::string attributeName, double value, bool createIfNotFound=false)
- 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 (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Friends

- class Model

Additional Inherited Members

3.39.1 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.

3.39.2 Member Function Documentation

3.39.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.39.2.2 _loadInstance()

```
bool Entity::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.39.2.3 _saveInstance()

```
void Entity::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

3.39.2.4 entityNumber()

```
Util::identification Entity::entityNumber ( ) const
```

3.39.2.5 getAttributeValue() [1/4]

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

Here is the call graph for this function:

3.39.2.6 getAttributeValue() [2/4]

```
double Entity::getAttributeValue (
    std::string index,
    std::string attributeName )
```

Here is the call graph for this function:

3.39.2.7 getAttributeValue() [3/4]

```
double Entity::getAttributeValue (
    std::string index,
    Util::identification attributeID )
```

Here is the call graph for this function:

3.39.2.8 getAttributeValue() [4/4]

```
double Entity::getAttributeValue (
    Util::identification attributeID )
```

Here is the call graph for this function:

3.39.2.9 getEntityType()

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

3.39.2.10 getEntityTypeName()

```
std::string Entity::getEntityTypeName ( ) const
```

Here is the call graph for this function:

3.39.2.11 setAttributeValue() [1/4]

```
void Entity::setAttributeValue (
    std::string attributeName,
    double value,
    bool createIfNotFound = false )
```

Here is the call graph for this function:

3.39.2.12 setAttributeValue() [2/4]

```
void Entity::setAttributeValue (
    std::string index,
    std::string attributeName,
    double value,
    bool createIfNotFound = false )
```

Here is the call graph for this function:

3.39.2.13 setAttributeValue() [3/4]

```
void Entity::setAttributeValue (
    std::string index,
    Util::identification attributeID,
    double value )
```

Here is the call graph for this function:

3.39.2.14 setAttributeValue() [4/4]

```
void Entity::setAttributeValue (
    Util::identification attributeID,
    double value )
```

Here is the call graph for this function:

3.39.2.15 setEntityType()

```
void Entity::setEntityType (
    EntityType * entityType )
```

3.39.2.16 setEntityTypeName()

```
void Entity::setEntityTypeName (
    std::string entityTypeName )
```

Here is the call graph for this function:

3.39.2.17 show()

```
std::string Entity::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.39.3 Friends And Related Function Documentation

3.39.3.1 Model

```
friend class Model [friend]
```

3.40 EntityGroup Class Reference

```
#include <EntityGroup.h>
```

Inheritance diagram for EntityGroup:

Public Member Functions

- EntityGroup (Model *model, std::string name="")

- virtual ~EntityGroup ()
- virtual std::string show ()
- void insertElement (unsigned int idKey, Entity *modeldatum)
- void removeElement (unsigned int idKey, Entity *modeldatum)
- List< Entity * > * getGroup (unsigned int idKey)
- void initBetweenReplications ()

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.40.1 Constructor & Destructor Documentation

3.40.1.1 EntityGroup()

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

Here is the call graph for this function:

3.40.1.2 ~EntityGroup()

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

3.40.2 Member Function Documentation

3.40.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.40.2.2 _createInternalAndAttachedData()

```
void EntityGroup::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.40.2.3 _loadInstance()

```
bool EntityGroup::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.40.2.4 _saveInstance()

```
void EntityGroup::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.40.2.5 getGroup()

```
List< Entity * > * EntityGroup::getGroup (
    unsigned int idKey )
```

3.40.2.6 GetPluginInformation()

```
PluginInformation * EntityGroup::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.40.2.7 initBetweenReplications()

```
void EntityGroup::initBetweenReplications ( )
```

3.40.2.8 insertElement()

```
void EntityGroup::insertElement (
    unsigned int idKey,
    Entity * modeldatum )
```

Here is the call graph for this function:

3.40.2.9 LoadInstance()

```
ModelDataDefinition * EntityGroup::LoadInstance (
    Model * model,
    PersistenceRecord * fields )  [static]
```

Here is the call graph for this function:

3.40.2.10 NewInstance()

```
ModelDataDefinition * EntityGroup::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.40.2.11 removeElement()

```
void EntityGroup::removeElement (
    unsigned int idKey,
    Entity * modeldatum )
```

Here is the call graph for this function:

3.40.2.12 show()

```
std::string EntityGroup::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.41 EntityType Class Reference

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

Inheritance 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 ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.41.1 Constructor & Destructor Documentation

3.41.1.1 EntityType()

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

3.41.1.2 ~EntityType()

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

Here is the call graph for this function:

3.41.2 Member Function Documentation

3.41.2.1 _check() `bool EntityType::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.41.2.2 _createInternalAndAttachedData() `void EntityType::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.41.2.3 _initBetweenReplications() `void EntityType::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.41.2.4 _loadInstance()

```
bool EntityType::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.41.2.5 _saveInstance()

```
void EntityType::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.41.2.6 addGetStatisticsCollector()

```
StatisticsCollector * EntityType::addGetStatisticsCollector
(
    std::string name )
```

Here is the call graph for this function:

3.41.2.7 GetPluginInformation() `PluginInformation * EntityType::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.41.2.8 initialNVACost() `double EntityType::initialNVACost ( ) const`

3.41.2.9 initialOtherCost() `double EntityType::initialOtherCost ( ) const`

3.41.2.10 initialPicture() `std::string EntityType::initialPicture ( ) const`

3.41.2.11 initialVACost() `double EntityType::initialVACost ( ) const`

3.41.2.12 initialWaitingCost() `double EntityType::initialWaitingCost ( ) const`

3.41.2.13 LoadInstance()

```
ModelDataDefinition * EntityType::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.41.2.14 NewInstance()

```
ModelDataDefinition * EntityType::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.41.2.15 setInitialNVACost()

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

3.41.2.16 setInitialOtherCost()

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

3.41.2.17 setInitialPicture()

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

3.41.2.18 setInitialVACost()

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

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

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

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.42 PersistenceRecord::Entry Struct Reference

"Raw" entry, users should treat this as an opaque type since payloads could be in any format.

`#include <Persistence.h>`

Public Types

- enum Kind { text , numeric }

Public Attributes

- std::string first
- Payload second
- Kind kind {Kind::text}

3.42.1 Detailed Description

"Raw" entry, users should treat this as an opaque type since payloads could be in any format.

3.42.2 Member Enumeration Documentation

3.42.2.1 Kind `enum PersistenceRecord::Entry::Kind`

Enumerator

text	
numeric	

3.42.3 Member Data Documentation

3.42.3.1 first `std::string PersistenceRecord::Entry::first`

3.42.3.2 kind `Kind PersistenceRecord::Entry::kind {Kind::text}`

3.42.3.3 second `Payload PersistenceRecord::Entry::second`

3.43 Event Class Reference

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

Inheritance diagram for Event:

Public Member Functions

- Event (double time, Entity ∗entity, ModelComponent ∗component, unsigned int componentinputPort↩ Number=0)
- Event (double time, Entity ∗entity, Connection ∗connection)
- virtual ∼Event ()=default
- double getTime () const
- ModelComponent ∗ getComponent () const
- Entity ∗ getEntity () const
- unsigned int getComponentinputPortNumber () const
- void setEntity (Entity ∗_entity)
- virtual std::string show ()

Protected Attributes

- double _time

3.43.1 Detailed Description

An an instantaneaous 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.

3.43.2 Constructor & Destructor Documentation

3.43.2.1 Event() [1/2]

```
Event::Event (
    double time,
    Entity * entity,
    ModelComponent * component,
    unsigned int componentinputPortNumber = 0 )
```

3.43.2.2 Event() [2/2]

```
Event::Event (
    double time,
    Entity * entity,
    Connection * connection )
```

3.43.2.3 ~Event()

```
virtual Event::~Event ( ) [virtual], [default]
```

3.43.3 Member Function Documentation

3.43.3.1 getComponent()

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

3.43.3.2 getComponentinputPortNumber()

```
unsigned int Event::getComponentinputPortNumber ( ) const
```

3.43.3.3 getEntity()

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

3.43.3.4 getTime()

```
double Event::getTime ( ) const
```

3.43.3.5 setEntity()

```
void Event::setEntity (
    Entity * _entity )
```

3.43.3.6 show()

```
std::string Event::show ( ) [virtual]
```

Reimplemented in InternalEvent.

Here is the call graph for this function:

3.43.4 Member Data Documentation

3.43.4.1 _time

```
double Event::_time [protected]
```

3.44 Exact Class Reference

```
#include <Exact.h>
```

Public Member Functions

- Exact ()
- Exact (int num, int den=1)
- const Exact & operator+ (const Exact &right)
- const Exact & operator+= (const Exact &right)
- const Exact & operator-= (const Exact &right)
- const Exact & operator*= (const Exact &right)
- const Exact & operator/= (const Exact &right)
- const Exact & operator= (const Exact &right)
- const Exact & operator= (const int &right)
- bool operator== (const Exact &right)
- bool operator== (const double &right)
- bool operator< (const Exact &right)
- bool operator< (const double &right)
- bool operator<= (const Exact &right)
- bool operator<= (const double &right)
- bool operator> (const Exact &right)
- bool operator> (const double &right)
- bool operator>= (const Exact &right)
- bool operator>= (const double &right)
- void Simplify ()
- double toFloat ()

3.44.1 Constructor & Destructor Documentation

3.44.1.1 Exact() [1/2] `Exact::Exact ( ) [inline]`

3.44.1.2 Exact() [2/2]

```
Exact::Exact (
    int num,
    int den = 1 ) [inline]
```

3.44.2 Member Function Documentation

3.44.2.1 operator*=()

```
const Exact & Exact::operator*= (
    const Exact & right ) [inline]
```

3.44.2.2 operator+()

```
const Exact & Exact::operator+ (
    const Exact & right ) [inline]
```

3.44.2.3 operator+=()

```
const Exact & Exact::operator+= (
    const Exact & right ) [inline]
```

3.44.2.4 operator-=()

```
const Exact & Exact::operator-= (
    const Exact & right ) [inline]
```

3.44.2.5 operator/=()

```
const Exact & Exact::operator/= (
    const Exact & right ) [inline]
```

3.44.2.6 operator<() [1/2]

```
bool Exact::operator< (
    const double & right ) [inline]
```

3.44.2.7 operator<() [2/2]

```
bool Exact::operator< (
    const Exact & right ) [inline]
```

3.44.2.8 operator<=() [1/2]

```
bool Exact::operator<= (
    const double & right ) [inline]
```

3.44.2.9 operator<=() [2/2]

```
bool Exact::operator<= (
    const Exact & right ) [inline]
```

3.44.2.10 operator=() [1/2]

```
const Exact & Exact::operator= (
    const Exact & right ) [inline]
```

3.44.2.11 operator=() [2/2]

```
const Exact & Exact::operator= (
    const int & right ) [inline]
```

3.44.2.12 operator==() [1/2]

```
bool Exact::operator== (
    const double & right ) [inline]
```

3.44.2.13 operator==() [2/2]

```
bool Exact::operator== (
    const Exact & right ) [inline]
```

3.44.2.14 operator>() [1/2]

```
bool Exact::operator> (
    const double & right ) [inline]
```

3.44.2.15 operator>() [2/2]

```
bool Exact::operator> (
    const Exact & right ) [inline]
```

3.44.2.16 operator>=() [1/2]

```
bool Exact::operator>= (
    const double & right ) [inline]
```

3.44.2.17 operator>=() [2/2] bool Exact::operator>= (const Exact & *right*) [inline]

3.44.2.18 Simplify() void Exact::Simplify ( ) [inline]

3.44.2.19 toFloat() double Exact::toFloat ( ) [inline]

3.45 Exit Class Reference

```
#include <Exit.h>
```

Inheritance diagram for Exit:

Public Member Functions

- Exit (Model ∗model, std::string name="")
- virtual ∼Exit ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.45.1 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.

3.45.2 of Cells Number of contiguous conveyor cells the entity will relinquish.

3.45.3 Constructor & Destructor Documentation

3.45.3.1 Exit()

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

3.45.3.2 ∼Exit()

```
virtual Exit::~Exit ( ) [virtual], [default]
```

3.45.4 Member Function Documentation

3.45.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.45.4.2 _loadInstance()

```
bool Exit::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.45.4.3 _onDispatchEvent()

```
void Exit::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.45.4.4 _saveInstance()

```
void Exit::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.45.4.5 GetPluginInformation() `PluginInformation * Exit::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.45.4.6 LoadInstance() `ModelComponent * Exit::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.45.4.7 NewInstance()

```
ModelDataDefinition * Exit::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.45.4.8 show()

```
std::string Exit::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.46 ExperimentManager Class Reference

```
#include <ExperimentManager.h>
```

Public Member Functions

- ExperimentManager (Simulator *simulator)
- SimulationExperiment * newSimulationExperiment ()
- void insert (SimulationExperiment *experiment)
- void remove (SimulationExperiment *experiment)
- void setCurrent (SimulationExperiment *experiment)
- bool saveSimulationExperiment (std::string filename)
- bool loadSimulationExperiment (std::string filename)
- unsigned int size ()
- SimulationExperiment * front ()
- SimulationExperiment * current ()
- SimulationExperiment * next ()
- List< SimulationExperiment * > * getExperiments () const

3.46.1 Constructor & Destructor Documentation

3.46.1.1 ExperimentManager() `ExperimentManager::ExperimentManager ( Simulator * simulator )`

3.46.2 Member Function Documentation

3.46.2.1 current() `SimulationExperiment * ExperimentManager::current ( )`

3.46.2.2 front() `SimulationExperiment * ExperimentManager::front ( )`

3.46.2.3 getExperiments() `List< SimulationExperiment * > * ExperimentManager::getExperiments ( ) const`

3.46.2.4 insert() `void ExperimentManager::insert ( SimulationExperiment * experiment )`

Here is the call graph for this function:

3.46.2.5 loadSimulationExperiment()

```
bool ExperimentManager::loadSimulationExperiment (
    std::string filename )
```

Here is the call graph for this function:

3.46.2.6 newSimulationExperiment()

```
SimulationExperiment * ExperimentManager::newSimulation↩
Experiment ( )
```

3.46.2.7 next()

```
SimulationExperiment * ExperimentManager::next ( )
```

3.46.2.8 remove()

```
void ExperimentManager::remove (
    SimulationExperiment * experiment )
```

Here is the call graph for this function:

3.46.2.9 saveSimulationExperiment()

```
bool ExperimentManager::saveSimulationExperiment (
    std::string filename )
```

3.46.2.10 setCurrent()

```
void ExperimentManager::setCurrent (
    SimulationExperiment * experiment )
```

3.46.2.11 size()

```
unsigned int ExperimentManager::size ( )
```

3.47 ExperimentManager_if Class Reference

```
#include <ExperimetManager_if.h>
```

Inheritance diagram for ExperimentManager_if:

Public Member Functions

- virtual List< SimulationScenario * > * getScenarios () const =0
- virtual void startSimulationOfScenario (SimulationScenario *scenario)=0
- virtual void startExperiment ()=0
- virtual void stopExperiment ()=0
- virtual void addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)=0

3.47.1 Detailed Description

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

3.47.2 Member Function Documentation

3.47.2.1 addTraceSimulationHandler()

```
virtual void ExperimentManager_if::addTraceSimulation↩
Handler (
            traceSimulationProcessListener traceSimulationProcessListener ) [pure virtual]
```

Implemented in ExperimentManagerDefaultImpl1.

3.47.2.2 getScenarios()

```
virtual List< SimulationScenario * > * ExperimentManager_if::get↩
Scenarios ( ) const [pure virtual]
```

Implemented in ExperimentManagerDefaultImpl1.

3.47.2.3 startExperiment()

```
virtual void ExperimentManager_if::startExperiment ( ) [pure virtual]
```

Implemented in ExperimentManagerDefaultImpl1.

3.47.2.4 startSimulationOfScenario()

```
virtual void ExperimentManager_if::startSimulationOf↩
Scenario (
            SimulationScenario * scenario ) [pure virtual]
```

Implemented in ExperimentManagerDefaultImpl1.

3.47.2.5 stopExperiment()

```
virtual void ExperimentManager_if::stopExperiment ( ) [pure virtual]
```

Implemented in ExperimentManagerDefaultImpl1.

3.48 ExperimentManagerDefaultImpl1 Class Reference

```
#include <ExperimentManagerDefaultImpl1.h>
```

Inheritance diagram for ExperimentManagerDefaultImpl1:

Public Member Functions

- ExperimentManagerDefaultImpl1 ()
- virtual ~ExperimentManagerDefaultImpl1 ()=default
- virtual List< SimulationScenario * > * getScenarios () const
- virtual void startSimulationOfScenario (SimulationScenario *scenario)
- virtual void startExperiment ()
- virtual void stopExperiment ()
- virtual void addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)

3.48.1 Constructor & Destructor Documentation

3.48.1.1 ExperimentManagerDefaultImpl1() `ExperimentManagerDefaultImpl1::ExperimentManager↩ DefaultImpl1 ( )`

3.48.1.2 ~ExperimentManagerDefaultImpl1() `virtual ExperimentManagerDefaultImpl1::~Experiment↩ ManagerDefaultImpl1 ( ) [virtual], [default]`

3.48.2 Member Function Documentation

3.48.2.1 addTraceSimulationHandler() `void ExperimentManagerDefaultImpl1::addTraceSimulation↩ Handler (`
`traceSimulationProcessListener` *traceSimulationProcessListener* `) [virtual]`

Implements ExperimentManager_if.

3.48.2.2 getScenarios() `List< SimulationScenario * > * ExperimentManagerDefaultImpl1::get↩ Scenarios ( ) const [virtual]`

Implements ExperimentManager_if.

3.48.2.3 startExperiment() `void ExperimentManagerDefaultImpl1::startExperiment ( ) [virtual]`

Implements ExperimentManager_if.

3.48.2.4 startSimulationOfScenario() `void ExperimentManagerDefaultImpl1::startSimulationOf↩ Scenario (`
`SimulationScenario *` *scenario* `) [virtual]`

Implements ExperimentManager_if.

3.48.2.5 stopExperiment() `void ExperimentManagerDefaultImpl1::stopExperiment ( ) [virtual]`

Implements ExperimentManager_if.

3.49 ExtendedFSM Class Reference

`#include <EFSM.h>`

Inheritance diagram for ExtendedFSM:

Public Member Functions

- ExtendedFSM (Model ∗model, std::string name="")
- virtual ∼ExtendedFSM ()=default
- virtual std::string show ()

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.49.1 Constructor & Destructor Documentation

3.49.1.1 ExtendedFSM()

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

3.49.1.2 ∼ExtendedFSM()

```
virtual ExtendedFSM::~ExtendedFSM ( ) [virtual], [default]
```

3.49.2 Member Function Documentation

3.49.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.49.2.2 _createInternalAndAttachedData() `void ExtendedFSM::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.49.2.3 _initBetweenReplications() `void ExtendedFSM::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.49.2.4 _loadInstance() `bool ExtendedFSM::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.49.2.5 _saveInstance() `void ExtendedFSM::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.49.2.6 GetPluginInformation() `PluginInformation * ExtendedFSM::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.49.2.7 LoadInstance() `ModelDataDefinition * ExtendedFSM::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.49.2.8 NewInstance()

```
ModelDataDefinition * ExtendedFSM::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.49.2.9 show()

```
std::string ExtendedFSM::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.50 Failure Class Reference

```
#include <Failure.h>
```

Inheritance diagram for Failure:

Public Types

- enum class FailureType : int { COUNT = 1 , TIME = 2 }
- enum class FailureRule : int { IGNORE = 1 , PREEMPT = 2 , WAIT = 3 }

Public Member Functions

- Failure (Model ∗model, std::string name="")

- virtual ~Failure ()=default
- virtual std::string show ()
- void checkIsGoingToFailByCount (Resource *resource)
- void setFailureType (FailureType _failureType)
- Failure::FailureType getFailureType () const
- void setCountExpression (std::string countExpression)
- std::string getCountExpression () const
- void setDownTimeTimeUnit (Util::TimeUnit downTimeTimeUnit)
- Util::TimeUnit getDownTimeTimeUnit () const
- void setDownTimeExpression (std::string downTimeExpression)
- std::string getDownTimeExpression () const
- void setUpTimeTimeUnit (Util::TimeUnit upTimeTimeUnit)
- Util::TimeUnit getUpTimeTimeUnit () const
- void setUpTimeExpression (std::string upTimeExpression)
- std::string getUpTimeExpression () const
- void setFailureRule (FailureRule _failureRule)
- FailureRule getFailureRule () const
- List< Resource * > * falingResources () const

Static Public Member Functions

- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()

Additional Inherited Members

3.50.1 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.

3.50.2 Member Enumeration Documentation

3.50.2.1 FailureRule `enum class Failure::FailureRule : int [strong]`

Enumerator

IGNORE	
PREEMPT	
WAIT	

3.50.2.2 FailureType `enum class Failure::FailureType : int [strong]`

Enumerator

COUNT	
TIME	

3.50.3 Constructor & Destructor Documentation

3.50.3.1 Failure()

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

3.50.3.2 ~Failure() `virtual Failure::~Failure ( ) [virtual], [default]`

3.50.4 Member Function Documentation

3.50.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.50.4.2 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.50.4.3 _loadInstance()

```
bool Failure::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.50.4.4 _saveInstance()

```
void Failure::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.50.4.5 checkIsGoingToFailByCount()

```
void Failure::checkIsGoingToFailByCount ( Resource * resource )
```

Here is the call graph for this function:

3.50.4.6 falingResources()

```
List< Resource * > * Failure::falingResources ( ) const
```

3.50.4.7 getCountExpression()

```
std::string Failure::getCountExpression ( ) const
```

3.50.4.8 getDownTimeExpression()

```
std::string Failure::getDownTimeExpression ( ) const
```

3.50.4.9 getDownTimeTimeUnit()

```
Util::TimeUnit Failure::getDownTimeTimeUnit ( ) const
```

3.50.4.10 getFailureRule() `Failure::FailureRule Failure::getFailureRule ( ) const`

3.50.4.11 getFailureType() `Failure::FailureType Failure::getFailureType ( ) const`

3.50.4.12 GetPluginInformation() `PluginInformation * Failure::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.50.4.13 getUpTimeExpression() `std::string Failure::getUpTimeExpression ( ) const`

3.50.4.14 getUpTimeTimeUnit() `Util::TimeUnit Failure::getUpTimeTimeUnit ( ) const`

3.50.4.15 LoadInstance() `ModelDataDefinition * Failure::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.50.4.16 NewInstance()

```
ModelDataDefinition * Failure::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.50.4.17 setCountExpression()

```
void Failure::setCountExpression (
    std::string countExpression )
```

3.50.4.18 setDownTimeExpression()

```
void Failure::setDownTimeExpression (
    std::string downTimeExpression )
```

3.50.4.19 setDownTimeTimeUnit()

```
void Failure::setDownTimeTimeUnit (
    Util::TimeUnit downTimeTimeUnit )
```

3.50.4.20 setFailureRule()

```
void Failure::setFailureRule (
    FailureRule _failureRule )
```

3.50.4.21 setFailureType()

```
void Failure::setFailureType (
    FailureType _failureType )
```

3.50.4.22 setUpTimeExpression()

```
void Failure::setUpTimeExpression (
    std::string upTimeExpression )
```

3.50.4.23 setUpTimeTimeUnit()

```
void Failure::setUpTimeTimeUnit (
    Util::TimeUnit upTimeTimeUnit )
```

3.50.4.24 show()

```
std::string Failure::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.51 File Class Reference

```
#include <File.h>
```

Inheritance diagram for File:

Public Member Functions

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

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation ∗ _getParserChangesInformation ()

Additional Inherited Members

3.51.1 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 PROMPTS 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.

3.51.2 Constructor & Destructor Documentation

3.51.2.1 File()

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

3.51.2.2 ∼File()

```
virtual File::~File ( ) [virtual], [default]
```

3.51.3 Member Function Documentation

3.51.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.51.3.2 _getParserChangesInformation()

```
ParserChangesInformation * File::_getParserChanges↩
Information ( ) [protected], [virtual]
```

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.51.3.3 _loadInstance()

```
bool File::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.51.3.4 _saveInstance()

```
void File::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.51.3.5 GetPluginInformation() `PluginInformation * File::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.51.3.6 LoadInstance() `ModelDataDefinition * File::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.51.3.7 NewInstance()

```
ModelDataDefinition * File::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.51.3.8 show()

```
std::string File::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.52 Fitter_if Class Reference

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

Inheritance diagram for Fitter_if:

Public Member Functions

- virtual bool isNormalDistributed (double confidencelevel)=0
- virtual void fitUniform (double ∗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

3.52.1 Member Function Documentation

3.52.1.1 fitAll()

```
virtual void Fitter_if::fitAll (
    double * sqrerror,
    std::string * name )  [pure virtual]
```

Implemented in FitterDummyImpl.

3.52.1.2 fitBeta()

```
virtual void Fitter_if::fitBeta (
    double * sqrerror,
    double * alpha,
    double * beta,
    double * infLimit,
    double * supLimit )  [pure virtual]
```

Implemented in FitterDummyImpl.

3.52.1.3 fitErlang()

```
virtual void Fitter_if::fitErlang (
    double * sqrerror,
    double * avg,
    double * m )  [pure virtual]
```

Implemented in FitterDummyImpl.

3.52.1.4 fitExpo()

```
virtual void Fitter_if::fitExpo (
    double * sqrerror,
    double * avg1 )  [pure virtual]
```

Implemented in FitterDummyImpl.

3.52.1.5 fitNormal()

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

Implemented in FitterDummyImpl.

3.52.1.6 fitTriangular()

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

Implemented in FitterDummyImpl.

3.52.1.7 fitUniform()

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

Implemented in FitterDummyImpl.

3.52.1.8 fitWeibull()

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

Implemented in FitterDummyImpl.

3.52.1.9 getDataFilename()

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

Implemented in FitterDummyImpl.

3.52.1.10 isNormalDistributed()

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

Implemented in FitterDummyImpl.

3.52.1.11 setDataFilename()

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

Implemented in FitterDummyImpl.

3.53 FitterDummyImpl Class Reference

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

Inheritance diagram for FitterDummyImpl:

Public Member Functions

- FitterDummyImpl ()

- FitterDummyImpl (const FitterDummyImpl &orig)
- ~FitterDummyImpl ()
- 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 ()

3.53.1 Constructor & Destructor Documentation

3.53.1.1 FitterDummyImpl() [1/2] `FitterDummyImpl::FitterDummyImpl ( )`

3.53.1.2 FitterDummyImpl() [2/2]

```
FitterDummyImpl::FitterDummyImpl (
    const FitterDummyImpl & orig )
```

3.53.1.3 ~FitterDummyImpl() `FitterDummyImpl::~FitterDummyImpl ( )`

3.53.2 Member Function Documentation

3.53.2.1 fitAll()

```
void FitterDummyImpl::fitAll (
    double * sqrerror,
    std::string * name )  [virtual]
```

Implements Fitter_if.

3.53.2.2 fitBeta()

```
void FitterDummyImpl::fitBeta (
    double * sqrerror,
    double * alpha,
    double * beta,
    double * infLimit,
    double * supLimit )  [virtual]
```

Implements Fitter_if.

3.53.2.3 fitErlang()

```
void FitterDummyImpl::fitErlang (
    double * sqrerror,
    double * avg,
    double * m ) [virtual]
```

Implements Fitter_if.

3.53.2.4 fitExpo()

```
void FitterDummyImpl::fitExpo (
    double * sqrerror,
    double * avg1 ) [virtual]
```

Implements Fitter_if.

3.53.2.5 fitNormal()

```
void FitterDummyImpl::fitNormal (
    double * sqrerror,
    double * avg,
    double * stddev ) [virtual]
```

Implements Fitter_if.

3.53.2.6 fitTriangular()

```
void FitterDummyImpl::fitTriangular (
    double * sqrerror,
    double * min,
    double * mo,
    double * max ) [virtual]
```

Implements Fitter_if.

3.53.2.7 fitUniform()

```
void FitterDummyImpl::fitUniform (
    double * sqrerror,
    double * min,
    double * max ) [virtual]
```

Implements Fitter_if.

3.53.2.8 fitWeibull()

```
void FitterDummyImpl::fitWeibull (
    double * sqrerror,
    double * alpha,
    double * scale ) [virtual]
```

Implements Fitter_if.

3.53.2.9 getDataFilename() `std::string FitterDummyImpl::getDataFilename ( ) [virtual]`

Implements Fitter_if.

3.53.2.10 isNormalDistributed() `bool FitterDummyImpl::isNormalDistributed ( double confidencelevel ) [virtual]`

Implements Fitter_if.

3.53.2.11 setDataFilename() `void FitterDummyImpl::setDataFilename ( std::string dataFilename ) [virtual]`

Implements Fitter_if.

3.54 Formula Class Reference

```
#include <Formula.h>
```

Inheritance 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 formulaExpression, std::string index="")
- std::string getExpression (std::string index="")
- double getValue (std::string index)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.54.1 Constructor & Destructor Documentation

3.54.1.1 Formula()

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

3.54.1.2 ~Formula()

```
virtual Formula::~Formula ( ) [virtual], [default]
```

3.54.2 Member Function Documentation

3.54.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.54.2.2 _createInternalAndAttachedData() `void Formula::_createInternalAndAttachedData ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.54.2.3 _loadInstance() `bool Formula::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.54.2.4 _saveInstance() `void Formula::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.54.2.5 getExpression()

```
std::string Formula::getExpression (
    std::string index = "" )
```

3.54.2.6 GetPluginInformation()

```
PluginInformation * Formula::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.54.2.7 getValue()

```
double Formula::getValue (
    std::string index )
```

Here is the call graph for this function:

3.54.2.8 LoadInstance()

```
ModelDataDefinition * Formula::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.54.2.9 NewInstance()

```
ModelDataDefinition * Formula::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.54.2.10 setExpression()

```
void Formula::setExpression (
    std::string formulaExpression,
    std::string index = "" )
```

3.54.2.11 show()

```
std::string Formula::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.54.2.12 size() `unsigned int Formula::size ( )`

3.55 FSM_State Class Reference

```
#include <EFSM.h>
```

Inheritance diagram for FSM_State:

Public Member Functions

- FSM_State (std::string name, std::string refinementName="", bool isInitialState=false, bool isFinalState=false)
- void setName (std::string _name)
- std::string getName () const
- void setIsFinalState (bool _isFinalState)
- bool isFinalState () const
- void setIsInitialState (bool _isInitialState)
- bool isInitialState () const
- void setRefinementName (std::string _refinementName)
- std::string getRefinementName () const

Additional Inherited Members

3.55.1 Constructor & Destructor Documentation

3.55.1.1 FSM_State()

```
FSM_State::FSM_State (
    std::string name,
    std::string refinementName = "",
    bool isInitialState = false,
    bool isFinalState = false ) [inline]
```

Here is the call graph for this function:

3.55.2 Member Function Documentation

3.55.2.1 getName()

```
std::string FSM_State::getName ( ) const [inline]
```

3.55.2.2 getRefinementName()

```
std::string FSM_State::getRefinementName ( ) const [inline]
```

3.55.2.3 isFinalState()

```
bool FSM_State::isFinalState ( ) const [inline]
```

3.55.2.4 isInitialState()

```
bool FSM_State::isInitialState ( ) const [inline]
```

3.55.2.5 setIsFinalState()

```
void FSM_State::setIsFinalState (
    bool _isFinalState ) [inline]
```

3.55.2.6 setIsInitialState()

```
void FSM_State::setIsInitialState (
    bool _isInitialState ) [inline]
```

3.55.2.7 setName()

```
void FSM_State::setName (
    std::string _name ) [inline]
```

3.55.2.8 setRefinementName()

```
void FSM_State::setRefinementName (
    std::string _refinementName ) [inline]
```

3.56 FSM_Transition Class Reference

```
#include <EFSM.h>
```

Inheritance diagram for FSM_Transition:

Classes

- struct TransitionType

Public Member Functions

- FSM_Transition (std::string parameterName, FSM_State *originState, FSM_State *destinationState, std↩ ::string guardExpression="", TransitionType *transitionType=nullptr)

Additional Inherited Members

3.56.1 Constructor & Destructor Documentation

3.56.1.1 FSM_Transition()

```
FSM_Transition::FSM_Transition (
    std::string parameterName,
    FSM_State * originState,
    FSM_State * destinationState,
    std::string guardExpression = "",
    TransitionType * transitionType = nullptr )  [inline]
```

3.57 ParserManager::GenerateNewParserResult Struct Reference

```
#include <ParserManager.h>
```

Public Attributes

- bool result
- std::string bisonMessages
- std::string lexMessages
- std::string compilationMessages
- NewParser newParser

3.57.1 Member Data Documentation

3.57.1.1 bisonMessages

```
std::string ParserManager::GenerateNewParserResult::bisonMessages
```

3.57.1.2 compilationMessages

```
std::string ParserManager::GenerateNewParserResult::compilation↩
Messages
```

3.57.1.3 lexMessages

```
std::string ParserManager::GenerateNewParserResult::lexMessages
```

3.57.1.4 newParser `NewParser ParserManager::GenerateNewParserResult::newParser`

3.57.1.5 result `bool ParserManager::GenerateNewParserResult::result`

3.58 GenesysApplication_if Class Reference

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

Inheritance diagram for GenesysApplication_if:

Public Member Functions

- virtual int main (int argc, char ∗∗argv)=0

3.58.1 Member Function Documentation

3.58.1.1 main()

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

Implemented in BaseGenesysTerminalApplication, and GenesysShell.

3.59 genesyspp_driver Class Reference

```
#include <Genesys++-driver.h>
```

Public Member Functions

- genesyspp_driver ()
- genesyspp_driver (Model ∗model, Sampler_if ∗sampler, bool throws=false)
- virtual ∼genesyspp_driver ()=default
- void scan_begin_file ()
- void scan_end_file ()
- void scan_begin_str ()
- void scan_end_str ()
- int parse_file (const std::string &f)
- int parse_str (const std::string &str)
- void error (const yy::location &l, const std::string &m)
- void error (const std::string &m)
- double getResult ()
- void setResult (double value)
- bool getThrowsException ()
- void setThrowsException (bool throws)
- void setErrorMessage (std::string message)
- std::string getErrorMessage ()
- Model ∗ getModel ()
- std::string getFile ()
- void setFile (std::string f)
- std::string getStrToParse ()
- void setStrToParse (std::string str)
- void setSampler (Sampler_if ∗_sampler)
- Sampler_if ∗ getSampler () const
- void setRegisterReferedDataElements (bool value)
- bool getRegisterReferedDataElements ()
- std::map< std::string, std::list< std::string > ∗ > ∗ getReferedDataElements ()
- void clearReferedDataElements ()
- void addRefered (std::pair< std::string, std::string > referedElement)

3.59.1 Constructor & Destructor Documentation

3.59.1.1 genesyspp_driver() [1/2] genesyspp_driver::genesyspp_driver ( )

3.59.1.2 genesyspp_driver() [2/2] genesyspp_driver::genesyspp_driver (

```
Model * model,
Sampler_if * sampler,
bool throws = false )
```

3.59.1.3 ~genesyspp_driver() virtual genesyspp_driver::~genesyspp_driver ( ) [virtual], [default]

3.59.2 Member Function Documentation

3.59.2.1 addRefered() void genesyspp_driver::addRefered (

```
std::pair< std::string, std::string > referedElement )
```

3.59.2.2 clearReferedDataElements() void genesyspp_driver::clearReferedDataElements ( )

3.59.2.3 error() [1/2] void genesyspp_driver::error (

```
const std::string & m )
```

Here is the call graph for this function:

3.59.2.4 error() [2/2]

```
void genesyspp_driver::error (
    const yy::location & l,
    const std::string & m )
```

Here is the call graph for this function:

3.59.2.5 getErrorMessage()

```
std::string genesyspp_driver::getErrorMessage ( )
```

3.59.2.6 getFile()

```
std::string genesyspp_driver::getFile ( )
```

3.59.2.7 getModel()

```
Model * genesyspp_driver::getModel ( )
```

3.59.2.8 getReferedDataElements()

```
std::map< std::string, std::list< std::string > * > * genesyspp↩
_driver::getReferedDataElements ( )
```

3.59.2.9 getRegisterReferedDataElements()

```
bool genesyspp_driver::getRegisterReferedDataElements
( )
```

3.59.2.10 getResult()

```
double genesyspp_driver::getResult ( )
```

3.59.2.11 getSampler()

```
Sampler_if * genesyspp_driver::getSampler ( ) const
```

3.59.2.12 getStrToParse() `std::string genesyspp_driver::getStrToParse ( )`

3.59.2.13 getThrowsException() `bool genesyspp_driver::getThrowsException ( )`

3.59.2.14 parse_file() `int genesyspp_driver::parse_file ( const std::string & f )`

Here is the call graph for this function:

3.59.2.15 parse_str() `int genesyspp_driver::parse_str ( const std::string & str )`

Here is the call graph for this function:

3.59.2.16 scan_begin_file() `void genesyspp_driver::scan_begin_file ( )`

Here is the call graph for this function:

3.59.2.17 scan_begin_str() `void genesyspp_driver::scan_begin_str ( )`

3.59.2.18 scan_end_file() `void genesyspp_driver::scan_end_file ( )`

3.59.2.19 scan_end_str() `void genesyspp_driver::scan_end_str ( )`

3.59.2.20 setErrorMessage()

```
void genesyspp_driver::setErrorMessage (
    std::string message )
```

3.59.2.21 setFile()

```
void genesyspp_driver::setFile (
    std::string f )
```

3.59.2.22 setRegisterReferedDataElements()

```
void genesyspp_driver::setRegisterReferedDataElements (
    bool value )
```

3.59.2.23 setResult()

```
void genesyspp_driver::setResult (
    double value )
```

3.59.2.24 setSampler()

```
void genesyspp_driver::setSampler (
    Sampler_if * _sampler )
```

3.59.2.25 setStrToParse()

```
void genesyspp_driver::setStrToParse (
    std::string str )
```

3.59.2.26 setThrowsException()

```
void genesyspp_driver::setThrowsException (
    bool throws )
```

3.60 GenesysShell Class Reference

```
#include <GenesysShell.h>
```

Inheritance diagram for GenesysShell:

Public Member Functions

- GenesysShell ()

- GenesysShell (const GenesysShell &orig)
- virtual ~GenesysShell ()=default
- virtual int main (int argc, char **argv)
- void cmdScript ()
- void cmdHelp ()
- void cmdQuit ()
- void cmdBash ()
- void cmdParser ()
- void cmdTraceLevel ()
- void cmdPlugin ()
- void cmdSimulation ()
- void cmdReplication ()
- void cmdModel ()
- void cmdModelLoad ()
- void cmdModelSave ()

Additional Inherited Members

3.60.1 Constructor & Destructor Documentation

3.60.1.1 GenesysShell() [1/2] `GenesysShell::GenesysShell ( )`

Here is the call graph for this function:

3.60.1.2 GenesysShell() [2/2] `GenesysShell::GenesysShell ( const GenesysShell & orig )`

3.60.1.3 ~GenesysShell() `virtual GenesysShell::~GenesysShell ( ) [virtual], [default]`

3.60.2 Member Function Documentation

3.60.2.1 cmdBash() `void GenesysShell::cmdBash ( )`

3.60.2.2 cmdHelp() `void GenesysShell::cmdHelp ( )`

3.60.2.3 cmdModel() `void GenesysShell::cmdModel ( )`

Here is the call graph for this function:

3.60.2.4 cmdModelLoad() `void GenesysShell::cmdModelLoad ( )`

Here is the call graph for this function:

3.60.2.5 cmdModelSave() `void GenesysShell::cmdModelSave ( )`

Here is the call graph for this function:

3.60.2.6 cmdParser() `void GenesysShell::cmdParser ( )`

Here is the call graph for this function:

3.60.2.7 cmdPlugin() `void GenesysShell::cmdPlugin ( )`

Here is the call graph for this function:

3.60.2.8 cmdQuit() `void GenesysShell::cmdQuit ( )`

3.60.2.9 cmdReplication() `void GenesysShell::cmdReplication ( )`

Here is the call graph for this function:

3.60.2.10 cmdScript() `void GenesysShell::cmdScript ( )`

Here is the call graph for this function:

3.60.2.11 cmdSimulation() `void GenesysShell::cmdSimulation ( )`

Here is the call graph for this function:

3.60.2.12 cmdTraceLevel() `void GenesysShell::cmdTraceLevel ( )`

Here is the call graph for this function:

3.60.2.13 main() `int GenesysShell::main ( int argc, char ** argv ) [virtual]`

Reimplemented from BaseGenesysTerminalApplication.

Here is the call graph for this function:

3.61 GenesysShell_if Class Reference

```
#include <GenesysShell_if.h>
```

Inheritance 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 (TraceManager::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

3.61.1 Member Function Documentation

3.61.1.1 addFromFile()

```
virtual void GenesysShell_if::addFromFile (
    std::string filename ) [pure virtual]
```

3.61.1.2 addPlugin()

```
virtual void GenesysShell_if::addPlugin (
    std::string filename ) [pure virtual]
```

3.61.1.3 check()

```
virtual void GenesysShell_if::check ( ) [pure virtual]
```

3.61.1.4 closeModel()

```
virtual void GenesysShell_if::closeModel ( ) [pure virtual]
```

3.61.1.5 createModel()

`virtual void GenesysShell_if::createModel ( ) [pure virtual]`

3.61.1.6 deleteTraceFiles()

`virtual void GenesysShell_if::deleteTraceFiles ( ) [pure virtual]`

3.61.1.7 execLinuxCommand()

`virtual void GenesysShell_if::execLinuxCommand ( std::string command ) [pure virtual]`

3.61.1.8 getCommandLine()

`virtual void GenesysShell_if::getCommandLine ( ) [pure virtual]`

3.61.1.9 getGenesysInfo()

`virtual void GenesysShell_if::getGenesysInfo ( ) [pure virtual]`

3.61.1.10 listComponents()

`virtual void GenesysShell_if::listComponents ( ) [pure virtual]`

3.61.1.11 listElements()

`virtual void GenesysShell_if::listElements ( ) [pure virtual]`

3.61.1.12 listHosts()

`virtual void GenesysShell_if::listHosts ( ) [pure virtual]`

3.61.1.13 listPlugins()

`virtual void GenesysShell_if::listPlugins ( ) [pure virtual]`

3.61.1.14 openModel()

`virtual void GenesysShell_if::openModel ( std::string filename ) [pure virtual]`

3.61.1.15 readCommandsFromFile()

`virtual void GenesysShell_if::readCommandsFromFile ( std::string filename ) [pure virtual]`

3.61.1.16 receiveFile()

```
virtual void GenesysShell_if::receiveFile (
    std::string filename ) [pure virtual]
```

3.61.1.17 redirectTrace()

```
virtual void GenesysShell_if::redirectTrace (
    std::string trace,
    std::string dest,
    std::string filename ) [pure virtual]
```

3.61.1.18 saveModel()

```
virtual void GenesysShell_if::saveModel ( ) [pure virtual]
```

3.61.1.19 saveModelAs()

```
virtual void GenesysShell_if::saveModelAs (
    std::string filename ) [pure virtual]
```

3.61.1.20 sendFile()

```
virtual void GenesysShell_if::sendFile (
    std::string filename,
    std::string hostname,
    std::string portname ) [pure virtual]
```

3.61.1.21 setActivationCode()

```
virtual void GenesysShell_if::setActivationCode (
    std::string code ) [pure virtual]
```

3.61.1.22 showHelp()

```
virtual void GenesysShell_if::showHelp ( ) [pure virtual]
```

3.61.1.23 showHostName()

```
virtual void GenesysShell_if::showHostName ( ) [pure virtual]
```

3.61.1.24 showInit()

```
virtual void GenesysShell_if::showInit ( ) [pure virtual]
```

3.61.1.25 startSimulation()

```
virtual void GenesysShell_if::startSimulation ( ) [pure virtual]
```

3.61.1.26 stepSimulation() `virtual void GenesysShell_if::stepSimulation ( ) [pure virtual]`

3.61.1.27 stopSimulation() `virtual void GenesysShell_if::stopSimulation ( ) [pure virtual]`

3.61.1.28 traceLevel() `virtual void GenesysShell_if::traceLevel ( TraceManager::TraceLevel tracelevel ) [pure virtual]`

3.61.1.29 verboseMode() `virtual void GenesysShell_if::verboseMode ( bool on ) [pure virtual]`

3.62 GenSerializer Class Reference

`#include <GenSerializer.h>`

Inheritance diagram for GenSerializer:

Public Member Functions

- GenSerializer (Model *model)
- PersistenceRecord * newPersistenceRecord () override

  Creates an empty persistence record.
- bool dump (std::ostream &output) override

  Writes formatted contents to a given output stream.
- bool load (std::istream &input) override

  Attempts to load contents from a given input stream.
- bool get (const std::string &name, PersistenceRecord *entry) override

  Tries to fetch an entry with the given label as a key.
- bool put (const std::string name, const std::string type, const Util::identification id, PersistenceRecord *fields) override

  Upserts an entry into this structure.
- int for_each (std::function< int(const std::string &) > delegate) override

  Iterates over this structure's entries, ordered by ID.

Friends

- class ModelPersistenceDefaultImpl2

3.62.1 Constructor & Destructor Documentation

3.62.1.1 GenSerializer() `GenSerializer::GenSerializer ( Model * model ) [explicit]`

3.62.2 Member Function Documentation

3.62.2.1 dump() `bool GenSerializer::dump ( std::ostream & output ) [override], [virtual]`

Writes formatted contents to a given output stream.

Parameters

output	output stream to write to.

Returns

whether the current state of this structure's contents was deemed valid for serialization.

Implements ModelSerializer.

Here is the call graph for this function:

3.62.2.2 for_each()

```
int GenSerializer::for_each (
    std::function< int(const std::string &) > delegate )  [override], [virtual]
```

Iterates over this structure's entries, ordered by ID.

Parameters

delegate	delegate which is applied to every entry-identifying key, until it returns a non-zero value.

Returns

either zero, or the first non-zero value returned by the delegate.

Implements ModelSerializer.

3.62.2.3 get()

```
bool GenSerializer::get (
    const std::string & name,
    PersistenceRecord * entry )  [override], [virtual]
```

Tries to fetch an entry with the given label as a key.

Parameters

name	a label which uniquely identifies an entry.
entry	(out-param) a pointer to a caller-owned entry, which is filled on success.

Returns

whether the entry was found and the out-param was actually written to.

Implements ModelSerializer.

Here is the call graph for this function:

3.62.2.4 load() `bool GenSerializer::load ( std::istream & input ) [override], [virtual]`

Attempts to load contents from a given input stream.

Returns

whether the input was well-formatted and loading was successful.

Implements ModelSerializer.

Here is the call graph for this function:

3.62.2.5 newPersistenceRecord() `PersistenceRecord * GenSerializer::newPersistenceRecord ( ) [override], [virtual]`

Creates an empty persistence record.

Although the returned record can be freed normally with `delete`, care must be taken to only use records which come from a implementation, when configuring that implementation's contents.

Implements ModelSerializer.

Here is the call graph for this function:

3.62.2.6 put()

```
bool GenSerializer::put (
    const std::string name,
    const std::string type,
    const Util::identification id,
    PersistenceRecord * fields ) [override], [virtual]
```

Upserts an entry into this structure.

Parameters

name	a label which uniquely identifies the entry.
type	a type name.
id	a component/datadef ID number.
fields	a persistence record for the entry, whose contents will be copied if valid.

Returns

whether the entry was valid and was actually copied into this structure.

Implements ModelSerializer.

Here is the call graph for this function:

3.62.3 Friends And Related Function Documentation

3.62.3.1 ModelPersistenceDefaultImpl2 `friend class ModelPersistenceDefaultImpl2 [friend]`

3.63 InternalEvent Class Reference

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

Inheritance diagram for InternalEvent:

Public Member Functions

- InternalEvent (double time, std::string description)
- virtual ~InternalEvent ()=default
- template<typename Class >
  void setEventHandler (Class *object, void(Class::*function)(void *), void *parameter)
- virtual std::string show () override
- const InternalEventHandler & eventHandler () const
- const std::string & description () const
- void dispatchEvent () const
- void * object () const
- void * parameter () const

Additional Inherited Members

3.63.1 Constructor & Destructor Documentation

3.63.1.1 InternalEvent()

```
InternalEvent::InternalEvent (
    double time,
    std::string description ) [inline]
```

Here is the call graph for this function:

3.63.1.2 ~InternalEvent()

```
virtual InternalEvent::~InternalEvent ( ) [virtual], [default]
```

3.63.2 Member Function Documentation

3.63.2.1 description()

```
const std::string & InternalEvent::description ( ) const [inline]
```

3.63.2.2 dispatchEvent()

```
void InternalEvent::dispatchEvent ( ) const [inline]
```

3.63.2.3 eventHandler() `const InternalEventHandler & InternalEvent::eventHandler ( ) const [inline]`

3.63.2.4 object() `void * InternalEvent::object ( ) const [inline]`

3.63.2.5 parameter() `void * InternalEvent::parameter ( ) const [inline]`

3.63.2.6 setEventHandler()

```
template<typename Class >
void InternalEvent::setEventHandler (
            Class * object,
            void(Class::*)(void *) function,
            void * parameter ) [inline]
```

Here is the call graph for this function:

3.63.2.7 show() `virtual std::string InternalEvent::show ( ) [inline], [override], [virtual]`

Reimplemented from Event.

3.64 Json Struct Reference

Public Types

- enum Kind {
  null , boolean , number , string ,
  array , object }

Public Attributes

- enum Json::Kind kind
- double value
- std::string text
- std::vector< std::unique_ptr< Json > > sequence
- std::unordered_map< std::string, std::unique_ptr< Json > > fields

3.64.1 Member Enumeration Documentation

3.64.1.1 Kind `enum Json::Kind`

Enumerator

null	
boolean	
number	
string	
array	
object	

3.64.2 Member Data Documentation

3.64.2.1 fields `std::unordered_map<std::string, std::unique_ptr<Json> > Json::fields`

Initial value:

```
{
    }
```

3.64.2.2 kind `enum Json::Kind Json::kind`

3.64.2.3 sequence `std::vector<std::unique_ptr<Json> > Json::sequence`

Initial value:

```
{
    }
```

3.64.2.4 text `std::string Json::text`

3.64.2.5 value `double Json::value`

3.65 JsonSerializer Class Reference

`#include <JsonSerializer.h>`

Inheritance diagram for JsonSerializer:

Public Member Functions

- JsonSerializer (Model *model)
- PersistenceRecord * newPersistenceRecord () override

  Creates an empty persistence record.
- bool dump (std::ostream &output) override

Writes formatted contents to a given output stream.

- bool load (std::istream &input) override

  Attempts to load contents from a given input stream.

- bool get (const std::string &name, PersistenceRecord *entry) override

  Tries to fetch an entry with the given label as a key.

- bool put (const std::string name, const std::string type, const Util::identification id, PersistenceRecord *fields) override

  Upserts an entry into this structure.

- int for_each (std::function< int(const std::string &) > delegate) override

  Iterates over this structure's entries, ordered by ID.

3.65.1 Constructor & Destructor Documentation

3.65.1.1 JsonSerializer() `JsonSerializer::JsonSerializer ( Model * model ) [explicit]`

3.65.2 Member Function Documentation

3.65.2.1 dump() `bool JsonSerializer::dump ( std::ostream & output ) [override], [virtual]`

Writes formatted contents to a given output stream.

Parameters

output	output stream to write to.

Returns

whether the current state of this structure's contents was deemed valid for serialization.

Implements ModelSerializer.

Here is the call graph for this function:

3.65.2.2 for_each()

```
int JsonSerializer::for_each (
    std::function< int(const std::string &) > delegate )  [override], [virtual]
```

Iterates over this structure's entries, ordered by ID.

Parameters

delegate	delegate which is applied to every entry-identifying key, until it returns a non-zero value.

Returns

either zero, or the first non-zero value returned by the delegate.

Implements ModelSerializer.

3.65.2.3 get()

```
bool JsonSerializer::get (
    const std::string & name,
    PersistenceRecord * entry )  [override], [virtual]
```

Tries to fetch an entry with the given label as a key.

Parameters

name	a label which uniquely identifies an entry.
entry	(out-param) a pointer to a caller-owned entry, which is filled on success.

Returns

whether the entry was found and the out-param was actually written to.

Implements ModelSerializer.

Here is the call graph for this function:

3.65.2.4 load()

```
bool JsonSerializer::load (
    std::istream & input ) [override], [virtual]
```

Attempts to load contents from a given input stream.

Returns

whether the input was well-formatted and loading was successful.

Implements ModelSerializer.

Here is the call graph for this function:

3.65.2.5 newPersistenceRecord()

```
PersistenceRecord * JsonSerializer::newPersistenceRecord ( ) [override], [virtual]
```

Creates an empty persistence record.

Although the returned record can be freed normally with `delete`, care must be taken to only use records which come from a implementation, when configuring that implementation's contents.

Implements ModelSerializer.

Here is the call graph for this function:

3.65.2.6 put()

```
bool JsonSerializer::put (
    const std::string name,
    const std::string type,
    const Util::identification id,
    PersistenceRecord * fields ) [override], [virtual]
```

Upserts an entry into this structure.

Parameters

name	a label which uniquely identifies the entry.
type	a type name.
id	a component/datadef ID number.
fields	a persistence record for the entry, whose contents will be copied if valid.

Returns

whether the entry was valid and was actually copied into this structure.

Implements ModelSerializer.

Here is the call graph for this function:

3.66 Label Class Reference

```
#include <Label.h>
```

Inheritance diagram for Label:

Public Member Functions

- Label (Model *model, std::string name="")
- virtual ~Label ()=default
- virtual std::string show ()
- void setLabel (std::string _label)
- std::string getLabel () const
- ModelComponent * getEnterIntoLabelComponent () const
- void sendEntityToLabelComponent (Entity *entity, double timeDelay)

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.66.1 Constructor & Destructor Documentation

3.66.1.1 Label()

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

3.66.1.2 ∼Label()

```
virtual Label::~Label ( ) [virtual], [default]
```

3.66.2 Member Function Documentation

3.66.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.66.2.2 _loadInstance()

```
bool Label::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.66.2.3 _saveInstance()

```
void Label::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.66.2.4 getEnterIntoLabelComponent()

```
ModelComponent * Label::getEnterIntoLabelComponent ( ) const
```

3.66.2.5 getLabel() `std::string Label::getLabel ( ) const`

3.66.2.6 GetPluginInformation() `PluginInformation * Label::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.66.2.7 LoadInstance() `ModelDataDefinition * Label::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.66.2.8 NewInstance() `ModelDataDefinition * Label::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.66.2.9 sendEntityToLabelComponent()

```
void Label::sendEntityToLabelComponent (
    Entity * entity,
    double timeDelay )
```

Here is the call graph for this function:

3.66.2.10 setLabel()

```
void Label::setLabel (
    std::string _label )
```

3.66.2.11 show()

```
std::string Label::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.67 Leave Class Reference

```
#include <Leave.h>
```

Inheritance diagram for Leave:

Public Member Functions

- Leave (Model ∗model, std::string name="")
- virtual ∼Leave ()=default
- virtual std::string show ()

  virtual public methods
- void setStation (Station ∗_station)

- void setStationName (std::string stationName)
- Station ∗ getStation () const

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.67.1 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.

3.67.2 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.

3.67.3 Constructor & Destructor Documentation

3.67.3.1 Leave()

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

3.67.3.2 ~Leave()

```
virtual Leave::~Leave ( ) [virtual], [default]
```

3.67.4 Member Function Documentation

3.67.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.67.4.2 _createInternalAndAttachedData() `void Leave::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.67.4.3 _loadInstance() `bool Leave::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.67.4.4 _onDispatchEvent() `void Leave::_onDispatchEvent ( Entity * entity, unsigned int inputPortNumber ) [protected], [virtual]`

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.67.4.5 _saveInstance()

```
void Leave::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.67.4.6 GetPluginInformation() `PluginInformation * Leave::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.67.4.7 getStation() `Station * Leave::getStation ( ) const`

3.67.4.8 LoadInstance() `ModelComponent * Leave::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.67.4.9 NewInstance() `ModelDataDefinition * Leave::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.67.4.10 setStation()

```
void Leave::setStation (
    Station * _station )
```

3.67.4.11 setStationName()

```
void Leave::setStationName (
    std::string stationName )
```

Here is the call graph for this function:

3.67.4.12 show()

```
std::string Leave::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.68 LicenceManager Class Reference

```
#include <LicenceManager.h>
```

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 getModelDatasLimit ()
- unsigned int getEntityLimit ()
- unsigned int getHostsLimit ()
- unsigned int getThreadsLimit ()

3.68.1 Detailed Description

LicenceManager just prints a licence agreement and checks for some resource limits.

3.68.2 Constructor & Destructor Documentation

3.68.2.1 LicenceManager() `LicenceManager::LicenceManager ( Simulator * simulator )`

3.68.2.2 ~LicenceManager() `virtual LicenceManager::~LicenceManager ( ) [virtual], [default]`

3.68.3 Member Function Documentation

3.68.3.1 getEntityLimit() `unsigned int LicenceManager::getEntityLimit ( )`

3.68.3.2 getHostsLimit() `unsigned int LicenceManager::getHostsLimit ( )`

3.68.3.3 getModelComponentsLimit() `unsigned int LicenceManager::getModelComponentsLimit ( )`

3.68.3.4 getModelDatasLimit() `unsigned int LicenceManager::getModelDatasLimit ( )`

3.68.3.5 getThreadsLimit() `unsigned int LicenceManager::getThreadsLimit ( )`

3.68.3.6 insertActivationCode() `bool LicenceManager::insertActivationCode ( )`

3.68.3.7 lookforActivationCode() `bool LicenceManager::lookforActivationCode ( )`

3.68.3.8 removeActivationCode() `void LicenceManager::removeActivationCode ( )`

3.68.3.9 showActivationCode() `const std::string LicenceManager::showActivationCode ( ) const`

3.68.3.10 showLicence() `const std::string LicenceManager::showLicence ( ) const`

3.68.3.11 showLimits() `const std::string LicenceManager::showLimits ( ) const`

3.69 List< T > Class Template Reference

`#include <List.h>`

Inherited by ListObservable< T >.

Public Types

- using CompFunct = std::function< bool(const T, const T) >

Public Member Functions

- List ()
- List (List< T > &origin)
- 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)

Protected Attributes

- std::list< T > * _list
- CompFunct _sortFunc
- std::list< T >::iterator _it

  Default function: insert at the end of the list.

3.69.1 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.

3.69.2 Member Typedef Documentation

3.69.2.1 CompFunct `template<typename T >`

```
using List< T >::CompFunct = std::function<bool(const T, const T) >
```

3.69.3 Constructor & Destructor Documentation

3.69.3.1 List() [1/2] `template<typename T >`

```
List< T >::List
```

3.69.3.2 List() [2/2] `template<typename T >`

```
List< T >::List (
            List< T > & origin )
```

3.69.3.3 ~List() `template<typename T >`

```
virtual List< T >::~List ( ) [virtual], [default]
```

3.69.4 Member Function Documentation

3.69.4.1 clear()

```
template<typename T >
void List< T >::clear
```

3.69.4.2 create() [1/2]

```
template<typename T >
T List< T >::create
```

3.69.4.3 create() [2/2]

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

3.69.4.4 current()

```
template<typename T >
T List< T >::current
```

3.69.4.5 empty()

```
template<typename T >
bool List< T >::empty
```

3.69.4.6 find()

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

3.69.4.7 front()

```
template<typename T >
T List< T >::front
```

3.69.4.8 getAtRank()

```
template<typename T >
T List< T >::getAtRank (
            unsigned int rank )
```

3.69.4.9 insert()

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

3.69.4.10 last()

```
template<typename T >
T List< T >::last
```

3.69.4.11 list()

```
template<typename T >
std::list< T > * List< T >::list
```

3.69.4.12 next()

```
template<typename T >
T List< T >::next
```

3.69.4.13 pop_front()

```
template<typename T >
void List< T >::pop_front
```

3.69.4.14 previous()

```
template<typename T >
T List< T >::previous
```

3.69.4.15 remove()

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

3.69.4.16 setAtRank()

```
template<typename T >
void List< T >::setAtRank (
            unsigned int rank,
            T element )
```

3.69.4.17 setSortFunc()

```
template<typename T >
void List< T >::setSortFunc (
            CompFunct _sortFunc )
```

3.69.4.18 show()

```
template<typename T >
std::string List< T >::show
```

3.69.4.19 size() template<typename T >

```
unsigned int List< T >::size
```

3.69.4.20 sort() template<typename T >

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

3.69.5 Member Data Documentation

3.69.5.1 _it template<typename T >

```
std::list<T>::iterator List< T >::_it  [protected]
```

Default function: insert at the end of the list.

3.69.5.2 _list template<typename T >

```
std::list<T>* List< T >::_list  [protected]
```

3.69.5.3 _sortFunc template<typename T >

```
CompFunct List< T >::_sortFunc  [protected]
```

Initial value:

```
{[](const T, const T) {
            return false;
        }}
```

3.70 ListObservable< T > Class Template Reference

```
#include <ListObservable.h>
```

Inheritance diagram for ListObservable< T >:

Public Types

- using CompFunct = std::function< bool(const T, const T) >

Public Member Functions

- ListObservable ()
- virtual ~ListObservable ()=default
- void clear ()
- void pop_front ()
- void insert (T element)
- void remove (T element)
- void setAtRank (unsigned int rank, T element)

Additional Inherited Members

3.70.1 Detailed Description

template<typename T>
class ListObservable< T >

ListObservable corresponds to an extended version of the List that allows other classes to be notified when the list has changed.

3.70.2 Member Typedef Documentation

3.70.2.1 CompFunct `template<typename T >`

```
using ListObservable< T >::CompFunct = std::function<bool(const T, const T) >
```

3.70.3 Constructor & Destructor Documentation

3.70.3.1 ListObservable() `template<typename T >`

```
ListObservable< T >::ListObservable
```

3.70.3.2 ~ListObservable() `template<typename T >`

```
virtual ListObservable< T >::~ListObservable ( )  [virtual], [default]
```

3.70.4 Member Function Documentation

3.70.4.1 clear() `template<typename T >`

```
void ListObservable< T >::clear
```

3.70.4.2 insert() `template<typename T >`

```
void ListObservable< T >::insert (
            T element )
```

3.70.4.3 pop_front()

```
template<typename T >
void ListObservable< T >::pop_front
```

3.70.4.4 remove()

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

3.70.4.5 setAtRank()

```
template<typename T >
void ListObservable< T >::setAtRank (
            unsigned int rank,
            T element )
```

3.71 LSODE Class Reference

```
#include <LSODE.h>
```

Inheritance diagram for LSODE:

Public Member Functions

- LSODE (Model ∗model, std::string name="")
- virtual ∼LSODE ()=default
- virtual std::string show ()

  virtual public methods
- void setTimeVariable (Variable ∗_timeVariable)

- Variable ∗ getTimeVariable () const
- void setStep (double _step)
- double getStep () const
- void setVariable (Variable ∗_variables)
- Variable ∗ getVariable () const
- List< std::string > ∗ getDiffEquations () const
- void setFilename (std::string filename)
- std::string getFileName () const

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.71.1 Detailed Description

This component ...

3.71.2 Constructor & Destructor Documentation

3.71.2.1 LSODE()

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

3.71.2.2 ∼LSODE()

```
virtual LSODE::~LSODE ( ) [virtual], [default]
```

3.71.3 Member Function Documentation

3.71.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.71.3.2 _loadInstance()

```
bool LSODE::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.71.3.3 _onDispatchEvent()

```
void LSODE::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.71.3.4 _saveInstance()

```
void LSODE::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.71.3.5 getDiffEquations() `List< std::string > * LSODE::getDiffEquations ( ) const`

3.71.3.6 getFileName() `std::string LSODE::getFileName ( ) const`

3.71.3.7 GetPluginInformation() `PluginInformation * LSODE::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.71.3.8 getStep() `double LSODE::getStep ( ) const`

3.71.3.9 getTimeVariable() `Variable * LSODE::getTimeVariable ( ) const`

3.71.3.10 getVariable() `Variable * LSODE::getVariable ( ) const`

3.71.3.11 LoadInstance() `ModelComponent * LSODE::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.71.3.12 NewInstance()

```
ModelDataDefinition * LSODE::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.71.3.13 setFilename()

```
void LSODE::setFilename (
    std::string filename )
```

3.71.3.14 setStep()

```
void LSODE::setStep (
    double _step )
```

3.71.3.15 setTimeVariable()

```
void LSODE::setTimeVariable (
    Variable * _timeVariable )
```

3.71.3.16 setVariable()

```
void LSODE::setVariable (
    Variable * _variables )
```

3.71.3.17 show()

```
std::string LSODE::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.72 MarkovChain Class Reference

```
#include <MarkovChain.h>
```

Inheritance diagram for MarkovChain:

Public Member Functions

- MarkovChain (Model *model, std::string name="")

- virtual ~MarkovChain ()=default
- virtual std::string show ()

  virtual public methods
- void setTransitionProbabilityMatrix (Variable *_transitionMatrix)
- Variable * getTransitionMatrix () const
- Variable * getCurrentState () const
- void setInitialDistribution (Variable *_initialDistribution)
- Variable * getInitialState () const
- void setInitilized (bool _initilized)
- bool isInitilized () const
- void setCurrentState (Variable *_currentState)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual void _initBetweenReplications ()
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.72.1 Constructor & Destructor Documentation

3.72.1.1 MarkovChain()

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

3.72.1.2 ~MarkovChain()

```
virtual MarkovChain::~MarkovChain ( ) [virtual], [default]
```

3.72.2 Member Function Documentation

3.72.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.72.2.2 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

3.72.2.3 _loadInstance()

```
bool MarkovChain::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.72.2.4 _onDispatchEvent()

```
void MarkovChain::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.72.2.5 _saveInstance()

```
void MarkovChain::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.72.2.6 getCurrentState()

```
Variable * MarkovChain::getCurrentState ( ) const
```

3.72.2.7 getInitialState() `Variable * MarkovChain::getInitialState ( ) const`

3.72.2.8 GetPluginInformation() `PluginInformation * MarkovChain::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.72.2.9 getTransitionMatrix() `Variable * MarkovChain::getTransitionMatrix ( ) const`

3.72.2.10 isInitilized() `bool MarkovChain::isInitilized ( ) const`

3.72.2.11 LoadInstance() `ModelComponent * MarkovChain::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.72.2.12 NewInstance() `ModelDataDefinition * MarkovChain::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.72.2.13 setCurrentState()

```
void MarkovChain::setCurrentState (
    Variable * _currentState )
```

3.72.2.14 setInitialDistribution()

```
void MarkovChain::setInitialDistribution (
    Variable * _initialDistribution )
```

3.72.2.15 setInitilized()

```
void MarkovChain::setInitilized (
    bool _initilized )
```

3.72.2.16 setTransitionProbabilityMatrix()

```
void MarkovChain::setTransitionProbabilityMatrix (
    Variable * _transitionMatrix )
```

3.72.2.17 show()

```
std::string MarkovChain::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.73 Match Class Reference

```
#include <Match.h>
```

Inheritance diagram for Match:

Public Types

- enum class Rule : int { Any = 0 , ByAttribute = 1 }

Public Member Functions

- Match (Model ∗model, std::string name="")

- virtual ~Match ()=default
- virtual std::string show ()

  virtual public methods
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- void setRule (Match::Rule _rule)
- Match::Rule getRule () const
- void setAttributeName (std::string _attributeName)
- std::string getAttributeName () const
- void setMatchSize (std::string _matchSize)
- std::string getMatchSize () const
- void setNumberOfQueues (unsigned int _numberOfQueues)
- unsigned int getNumberOfQueues () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.73.1 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.

3.73.2 Member Enumeration Documentation

3.73.2.1 Rule `enum class Match::Rule : int [strong]`

Enumerator

Any	
ByAttribute	

3.73.3 Constructor & Destructor Documentation

3.73.3.1 Match()

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

3.73.3.2 ~Match()

```
virtual Match::~Match ( ) [virtual], [default]
```

3.73.4 Member Function Documentation

3.73.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.73.4.2 _createInternalAndAttachedData()

```
void Match::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.73.4.3 _loadInstance()

```
bool Match::_loadInstance (
    PersistenceRecord * fields ) [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.73.4.4 _onDispatchEvent()

```
void Match::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.73.4.5 _saveInstance()

```
void Match::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.73.4.6 getAttributeName() `std::string Match::getAttributeName ( ) const`

3.73.4.7 getMatchSize() `std::string Match::getMatchSize ( ) const`

3.73.4.8 getNumberOfQueues() `unsigned int Match::getNumberOfQueues ( ) const`

3.73.4.9 GetPluginInformation() `PluginInformation * Match::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.73.4.10 getRule()

```
Match::Rule Match::getRule ( ) const
```

3.73.4.11 LoadInstance()

```
ModelComponent * Match::LoadInstance (
    Model * model,
    PersistenceRecord * fields )  [static]
```

Here is the call graph for this function:

3.73.4.12 NewInstance()

```
ModelDataDefinition * Match::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.73.4.13 setAttributeName()

```
void Match::setAttributeName (
    std::string _attributeName )
```

3.73.4.14 setMatchSize()

```
void Match::setMatchSize (
    std::string _matchSize )
```

3.73.4.15 setNumberOfQueues()

```
void Match::setNumberOfQueues (
    unsigned int _numberOfQueues )
```

3.73.4.16 setRule() `void Match::setRule ( Match::Rule _rule )`

3.73.4.17 show() `std::string Match::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.74 Model Class Reference

`#include <Model.h>`

Public Member Functions

- Model (Simulator *simulator, unsigned int level=0)
- 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 ()
- std::string showLanguage ()
- void show ()
- bool insert (ModelDataDefinition *elemOrComp)

  Insert a new ModelDataDefinition or ModelComponent into the model (since 20191015). It's a generic access to ComponentManager->insert() or ModelDatao->insert()
- void remove (ModelDataDefinition *elemOrComp)

  Remove a new ModelDataDefinition or ModelComponent into the model (since 20191015). It's a generic access to ComponentManager->remove() or ModelDatao->remove()
- Entity * createEntity (std::string name, bool insertIntoModel=true)
- void removeEntity (Entity *entity)
- void sendEntityToComponent (Entity *entity, Connection *connection, double timeDelay=0.0)

  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).
- void sendEntityToComponent (Entity *entity, ModelComponent *component, double timeDelay=0.0, unsigned int componentinputPortNumber=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)

Invokes the parser to evaluate tyhe expression. Result is always a double, even if expression has syntatic errors (returns 0)

- double parseExpression (const std::string expression, bool ∗success, std::string ∗errorMessage)

  Invokes the parser to evaluate tyhe expression. Result is always a double, even if expression has syntatic errors (returns 0). Explicitly informs if there was an error.

- bool checkExpression (const std::string expression, const std::string expressionName, std::string ∗error↩ Message)

  This is invoked by ModelComponents and ModelDatas in their private method _check() to verify if an expression defined by user is valid or not.

- void checkReferencesToDataDefinitions (std::string expression, std::map< std::string, std::list< std::string > ∗ > ∗referencedDataDefinitions)
- Util::identification getId () const
- bool hasChanged () const
- OnEventManager ∗ getOnEvents () const

  Provides acccess to the class that manages events generated by the model, such as the beggining of a new simulation or replication, the processig of an event and much more.

- ModelDataManager ∗ getDataManager () 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
- ModelInfo ∗ getInfos () const
- Simulator ∗ getParentSimulator () const
- ModelSimulation ∗ getSimulation () const

  Provides access to the class that manages the model simulation.

- List< Event ∗ > ∗ getFutureEvents () 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).

- 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< SimulationControl ∗ > ∗ 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
- ModelPersistence_if ∗ getPersistence () const
- void setAutomaticallyCreatesModelDataDefinitions (bool _automaticallyCreatesModelDataDefinitions)
- bool isAutomaticallyCreatesModelDataDefinitions () const
- unsigned int getLevel () const

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

3.74.1 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 modeldatum (ModelDataDefinition).

3.74.2 Constructor & Destructor Documentation

3.74.2.1 Model()

```
Model::Model (
    Simulator * simulator,
    unsigned int level = 0 )
```

The future events list must be chronologicaly sorted

Events are sorted chronologicallyHere is the call graph for this function:

3.74.2.2 ~Model()

```
virtual Model::~Model ( ) [virtual], [default]
```

3.74.3 Member Function Documentation

3.74.3.1 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 the simulated.

Here is the call graph for this function:

3.74.3.2 checkExpression()

```
bool Model::checkExpression (
    const std::string expression,
    const std::string expressionName,
    std::string * errorMessage )
```

This is invoked by ModelComponents and ModelDatas in their private method _check() to verify if an expression defined by user is valid or not.

Here is the call graph for this function:

3.74.3.3 checkReferencesToDataDefinitions()

```
void Model::checkReferencesToDataDefinitions (
    std::string expression,
    std::map< std::string, std::list< std::string > * > * referencedDataDefinitions
)
```

Here is the call graph for this function:

3.74.3.4 clear()

```
void Model::clear ( )
```

Here is the call graph for this function:

3.74.3.5 createEntity()

```
Entity * Model::createEntity (
    std::string name,
    bool insertIntoModel = true )
```

Here is the call graph for this function:

3.74.3.6 getComponents() `ComponentManager * Model::getComponents ( ) const`

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

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

3.74.3.8 getDataManager() `ModelDataManager * Model::getDataManager ( ) const`

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

3.74.3.9 getFutureEvents() `List< Event * > * Model::getFutureEvents ( ) 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).

3.74.3.10 getId() `Util::identification Model::getId ( ) const`

3.74.3.11 getInfos() `ModelInfo * Model::getInfos ( ) const`

3.74.3.12 getLevel() `unsigned int Model::getLevel ( ) const`

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

3.74.3.13 getOnEvents() `OnEventManager * Model::getOnEvents ( ) const`

Provides acccess to the class that manages events generated by the model, such as the beggining of a new simulation or replication, the processig of an event and much more.

3.74.3.14 getParentSimulator() `Simulator * Model::getParentSimulator ( ) const`

3.74.3.15 getPersistence() `ModelPersistence_if * Model::getPersistence ( ) const`

3.74.3.16 getResponses() `List< SimulationControl * > * 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.

3.74.3.17 getSimulation() `ModelSimulation * Model::getSimulation ( ) const`

Provides access to the class that manages the model simulation.

3.74.3.18 getTracer() `TraceManager * Model::getTracer ( ) const`

3.74.3.19 hasChanged() `bool Model::hasChanged ( ) const`

Here is the call graph for this function:

3.74.3.20 insert()

```
bool Model::insert (
    ModelDataDefinition * elemOrComp )
```

Insert a new ModelDataDefinition or ModelComponent into the model (since 20191015). It's a generic access to ComponentManager->insert() or ModelDatao->insert()

Here is the call graph for this function:

3.74.3.21 isAutomaticallyCreatesModelDataDefinitions()

```
bool Model::isAutomaticallyCreatesModel↩
DataDefinitions ( ) const
```

3.74.3.22 load()

```
bool Model::load (
    std::string filename )
```

Here is the call graph for this function:

3.74.3.23 parseExpression() [1/2]

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

Invokes the parser to evaluate tyhe expression. Result is always a double, even if expression has syntatic errors (returns 0)

Here is the call graph for this function:

3.74.3.24 parseExpression() [2/2]

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

Invokes the parser to evaluate tyhe expression. Result is always a double, even if expression has syntatic errors (returns 0). Explicitly informs if there was an error.

Here is the call graph for this function:

3.74.3.25 remove()

```
void Model::remove (
    ModelDataDefinition * elemOrComp )
```

Remove a new ModelDataDefinition or ModelComponent into the model (since 20191015). It's a generic access to ComponentManager->remove() or ModelDatao->remove()

Here is the call graph for this function:

3.74.3.26 removeEntity()

```
void Model::removeEntity (
    Entity * entity )
```

Here is the call graph for this function:

3.74.3.27 save()

```
bool Model::save (
    std::string filename )
```

Here is the call graph for this function:

3.74.3.28 sendEntityToComponent() [1/2]

```
void Model::sendEntityToComponent (
    Entity * entity,
    Connection * connection,
    double timeDelay = 0.0 )
```

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).

Here is the call graph for this function:

3.74.3.29 sendEntityToComponent() [2/2]

```
void Model::sendEntityToComponent (
    Entity * entity,
    ModelComponent * component,
    double timeDelay = 0.0,
    unsigned int componentinputPortNumber = 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).

Here is the call graph for this function:

3.74.3.30 setAutomaticallyCreatesModelDataDefinitions()

```
void Model::setAutomaticallyCreates↩
ModelDataDefinitions (
            bool _automaticallyCreatesModelDataDefinitions )
```

3.74.3.31 setTracer()

```
void Model::setTracer (
            TraceManager * _traceManager )
```

3.74.3.32 show()

```
void Model::show ( )
```

Here is the call graph for this function:

3.74.3.33 showLanguage()

```
std::string Model::showLanguage ( )
```

Here is the call graph for this function:

3.75 ModelChecker_if Class Reference

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

Inheritance 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
- virtual bool checkOrphaned ()=0

3.75.1 Detailed Description

The ModelChecker is responsable for verifying the model consistency, fixing inconsistencies wheneaver possible

3.75.2 Member Function Documentation

3.75.2.1 checkActivationCode() `virtual bool ModelChecker_if::checkActivationCode ( ) [pure virtual]`

Checks if the installed version has acquired a valid activation code for commercial use

Implemented in ModelCheckerDefaultImpl1.

3.75.2.2 checkAll() `virtual bool ModelChecker_if::checkAll ( ) [pure virtual]`

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

Implemented in ModelCheckerDefaultImpl1.

3.75.2.3 checkConnected() `virtual bool ModelChecker_if::checkConnected ( ) [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.

3.75.2.4 checkLimits() `virtual bool ModelChecker_if::checkLimits ( ) [pure virtual]`

Checks if the model (its components and elements) fits the limits based on the activation code

Implemented in ModelCheckerDefaultImpl1.

3.75.2.5 checkOrphaned() `virtual bool ModelChecker_if::checkOrphaned ( ) [pure virtual]`

Checks if there are ModelDataDefinitions not referred by any ModelComponent, which means they are orphan ModelDataDefinitions

Implemented in ModelCheckerDefaultImpl1.

3.75.2.6 checkSymbols() `virtual bool ModelChecker_if::checkSymbols ( ) [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.

3.76 ModelCheckerDefaultImpl1 Class Reference

```
#include <ModelCheckerDefaultImpl1.h>
```

Inheritance 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 ()
- virtual bool checkOrphaned ()

3.76.1 Constructor & Destructor Documentation

3.76.1.1 ModelCheckerDefaultImpl1() `ModelCheckerDefaultImpl1::ModelCheckerDefaultImpl1 ( Model * model )`

3.76.1.2 ~ModelCheckerDefaultImpl1() `virtual ModelCheckerDefaultImpl1::~ModelCheckerDefaultImpl1 ( ) [virtual], [default]`

3.76.2 Member Function Documentation

3.76.2.1 checkActivationCode() `bool ModelCheckerDefaultImpl1::checkActivationCode ( ) [virtual]`

Checks if the installed version has acquired a valid activation code for commercial use

Implements ModelChecker_if.

Here is the call graph for this function:

3.76.2.2 checkAll() `bool ModelCheckerDefaultImpl1::checkAll ( ) [virtual]`

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

Implements ModelChecker_if.

Here is the call graph for this function:

3.76.2.3 checkConnected() `bool ModelCheckerDefaultImpl1::checkConnected ( ) [virtual]`

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

Implements ModelChecker_if.

Here is the call graph for this function:

3.76.2.4 checkLimits() `bool ModelCheckerDefaultImpl1::checkLimits ( ) [virtual]`

Checks if the model (its components and elements) fits the limits based on the activation code

Implements ModelChecker_if.

Here is the call graph for this function:

3.76.2.5 checkOrphaned() `bool ModelCheckerDefaultImpl1::checkOrphaned ( ) [virtual]`

Checks if there are ModelDataDefinitions not referred by any ModelComponent, which means they are orphan ModelDataDefinitions

Implements ModelChecker_if.

Here is the call graph for this function:

3.76.2.6 checkSymbols() `bool ModelCheckerDefaultImpl1::checkSymbols ( ) [virtual]`

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

Implements ModelChecker_if.

Here is the call graph for this function:

3.77 ModelComponent Class Reference

`#include <ModelComponent.h>`

Inherits ModelDataDefinition.

Inherited by Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Decide, Delay, DiffEquations, DropOff, DummyComponent, Enter, Exit, LSODE, Leave, MarkovChain, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, SinkModelComponent, SourceModelComponent, Start, Stop, Store, Submodel, Unstore, Wait, and Write.

Public Member Functions

- ModelComponent (Model *model, std::string componentTypename, std::string name="")

  constructors
- virtual ~ModelComponent ()
- ConnectionManager * getConnections () const

  new public user methods for this component
- bool hasBreakpointAt ()
- void setDescription (std::string _description)
- std::string getDescription () const
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)

  static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component
- static bool Check (ModelComponent ∗component)

  new static methods for all ModelComponents
- static void CreateInternalData (ModelComponent ∗component)
- static void SaveInstance (PersistenceRecord ∗fields, ModelComponent ∗component)
- static void DispatchEvent (Event ∗event)

  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.

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)=0

  new virtual methods for all ModelComponents

Protected Attributes

- ConnectionManager ∗ _connections = new ConnectionManager()

  virtual protected methods that could be overriden by derived classes, if needed

3.77.1 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

model	The model this component belongs to

3.77.2 Constructor & Destructor Documentation

3.77.2.1 ModelComponent()

```
ModelComponent::ModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "" )
```

constructors

Here is the call graph for this function:

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

Here is the call graph for this function:

3.77.3 Member Function Documentation

3.77.3.1 _loadInstance() `bool ModelComponent::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelDataDefinition.

Reimplemented in SinkModelComponent, SourceModelComponent, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, and Write.

Here is the call graph for this function:

3.77.3.2 _onDispatchEvent()

```
virtual void ModelComponent::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [pure virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implemented in Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, and Write.

3.77.3.3 _saveInstance()

```
void ModelComponent::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Reimplemented in SinkModelComponent, SourceModelComponent, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, and Write.

Here is the call graph for this function:

3.77.3.4 Check()

```
bool ModelComponent::Check (
    ModelComponent * component ) [static]
```

new static methods for all ModelComponents

Here is the call graph for this function:

3.77.3.5 CreateInternalData()

```
void ModelComponent::CreateInternalData ( ModelComponent * component ) [static]
```

Here is the call graph for this function:

3.77.3.6 DispatchEvent()

```
void ModelComponent::DispatchEvent ( Event * event ) [static]
```

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

Here is the call graph for this function:

3.77.3.7 getConnections() `ConnectionManager * ModelComponent::getConnections ( ) const`

new public user methods for this component

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 inputPortNumber 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).

3.77.3.8 getDescription() `std::string ModelComponent::getDescription ( ) const`

3.77.3.9 hasBreakpointAt() `bool ModelComponent::hasBreakpointAt ( )`

Here is the call graph for this function:

3.77.3.10 LoadInstance() `static ModelComponent * ModelComponent::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

static public methods that must have implementations (Load and New just the same. GetInformation must provide specific infos for the new component

3.77.3.11 SaveInstance()

```
void ModelComponent::SaveInstance (
    PersistenceRecord * fields,
    ModelComponent * component ) [static]
```

Here is the call graph for this function:

3.77.3.12 setDescription()

```
void ModelComponent::setDescription (
    std::string _description )
```

3.77.3.13 show()

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

virtual public methods

Reimplemented from ModelDataDefinition.

Reimplemented in SourceModelComponent, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, and Write.

Here is the call graph for this function:

3.77.4 Member Data Documentation

3.77.4.1 _connections `ConnectionManager* ModelComponent::_connections = new ConnectionManager() [protected]`

virtual protected methods that could be overriden by derived classes, if needed

This method is called by ModelChecker during model check. The component should check itself to verify if user parameters are ok (ex: correct syntax for the parser) and everithing in its parameters allow the model too run without errors in this component

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

This method is called by ModelSimulation when initianting the replication. The model should set all value for a new replication (Ex: setting back to 0 any internal counter, clearing lists, etc.

This method is called by ModelChecker and 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(This↩CSTAT)") and therefore the modeldatum must exist before checking such expression

This method is not used yet. It should be usefull for new UIs

3.78 ModelDataDefinition Class Reference

`#include <ModelDataDefinition.h>`

Inheritance diagram for ModelDataDefinition:

Public Member Functions

- ModelDataDefinition (Model ∗model, std::string datadefinitionTypename, std::string name="", bool insert↩IntoModel=true)
- virtual ∼ModelDataDefinition ()
- Util::identification getId () const
- void setName (std::string name)
- std::string getName () const
- std::string getClassname () const
- bool isReportStatistics () const
- void setReportStatistics (bool reportStatistics)
- ModelDataDefinition ∗ getInternalData (std::string name) const
- std::map< std::string, ModelDataDefinition ∗ > ∗ getInternalData () const
- std::map< std::string, ModelDataDefinition ∗ > ∗ getAttachedData () const
- bool hasChanged () const
- unsigned int getLevel () const
- void setModelLevel (unsigned int _modelLevel)
- List< PropertyBase ∗ > ∗ getProperties () const
- virtual std::string show ()

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields, bool insertIntoModel)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")
- static void SaveInstance (PersistenceRecord ∗fields, ModelDataDefinition ∗modeldatum)
- static bool Check (ModelDataDefinition ∗modeldatum, std::string ∗errorMessage)
- static void CreateInternalData (ModelDataDefinition ∗modeldatum)
- static void InitBetweenReplications (ModelDataDefinition ∗modeldatum)

Protected Member Functions

- void _internalDataClear ()

  methods to be called inside the _createInternalAndAttachedData() method
- void _internalDataInsert (std::string key, ModelDataDefinition ∗child)
- void _internalDataRemove (std::string key)
- void _attachedDataInsert (std::string key, ModelDataDefinition ∗data)
- void _attachedDataRemove (std::string key)
- void _attachedDataClear ()
- void _attachedAttributesInsert (std::vector< std::string > neededNames)
- void _checkCreateAttachedReferencedDataDefinition (std::string expression)

  method to be called to insert attached dataelements that are referenced by string expressions (detected by the parser), to avoid orphaned data definitions
- bool _getSaveDefaultsOption ()
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation ∗ _getParserChangesInformation ()
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- virtual void _addProperty (PropertyBase ∗property)
- void trace (std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)

  just an easy access to trace manager
- void traceError (std::string text, std::exception e)
- void traceError (std::string text, TraceManager::Level level=TraceManager::Level::L1_errorFatal)
- void traceReport (std::string text, TraceManager::Level level=TraceManager::Level::L2_results)
- void traceSimulation (void ∗thisobject, std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)
- void traceSimulation (void ∗thisobject, TraceManager::Level level, std::string text)

Protected Attributes

- Util::identification _id
- std::string _typename
- bool _reportStatistics
- bool _hasChanged
- unsigned int _modelLevel = 0
- Model ∗ _parentModel
- List< PropertyBase ∗ > ∗ _properties = new List<PropertyBase∗>()

3.78.1 Detailed Description

This class is the basis for any modeldatum 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.

3.78.2 Constructor & Destructor Documentation

3.78.2.1 ModelDataDefinition()

```
ModelDataDefinition::ModelDataDefinition (
    Model * model,
    std::string datadefinitionTypename,
    std::string name = "",
    bool insertIntoModel = true )
```

Here is the call graph for this function:

3.78.2.2 ~ModelDataDefinition()

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

Here is the call graph for this function:

3.78.3 Member Function Documentation

3.78.3.1 _addProperty()

```
void ModelDataDefinition::_addProperty (
    PropertyBase * property ) [protected], [virtual]
```

Reimplemented in Buffer, PickStation, and Schedule.

3.78.3.2 _attachedAttributesInsert()

```
void ModelDataDefinition::_attachedAttributesInsert (
    std::vector< std::string > neededNames ) [protected]
```

Here is the call graph for this function:

3.78.3.3 _attachedDataClear()

```
void ModelDataDefinition::_attachedDataClear ( ) [protected]
```

3.78.3.4 _attachedDataInsert()

```
void ModelDataDefinition::_attachedDataInsert (
    std::string key,
    ModelDataDefinition * data ) [protected]
```

3.78.3.5 _attachedDataRemove()

```
void ModelDataDefinition::_attachedDataRemove (
    std::string key ) [protected]
```

Here is the call graph for this function:

3.78.3.6 _check()

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

could be overriden by derived classes

Reimplemented in Attribute, Counter, Entity, EntityType, SinkModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, Sequence, Set, SignalData, Station, Storage, and Variable.

3.78.3.7 _checkCreateAttachedReferencedDataDefinition()

```
void ModelDataDefinition::_checkCreateAttachedReferencedDataDefinition (
            std::string expression ) [protected]
```

method to be called to insert attached dataelements that are referenced by string expressions (detected by the parser), to avoid orphaned data definitions

Here is the call graph for this function:

3.78.3.8 _createInternalAndAttachedData()

```
void ModelDataDefinition::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented in EntityType, SourceModelComponent, Assign, Batch, Buffer, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, Enter, Leave, Match, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Signal, Wait, CppCompiler, ExtendedFSM, EntityGroup, Formula, Queue, Resource, Schedule, Set, and Station.

3.78.3.9 _getParserChangesInformation()

```
ParserChangesInformation * ModelDataDefinition::_getParserChangesInformation ( ) [protected], [virtual]
```

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented in Buffer, PickStation, File, Queue, Schedule, Set, and Storage.

3.78.3.10 _getSaveDefaultsOption() `bool ModelDataDefinition::_getSaveDefaultsOption ( ) [protected]`

Here is the call graph for this function:

3.78.3.11 _initBetweenReplications() `void ModelDataDefinition::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented in Counter, EntityType, SourceModelComponent, Buffer, Clone, CppForG, Create, Dispose, MarkovChain, PickStation, Record, Release, Seize, Signal, Wait, Write, CppCompiler, ExtendedFSM, Failure, Queue, Resource, Schedule, SignalData, and Variable.

3.78.3.12 _internalDataClear() `void ModelDataDefinition::_internalDataClear ( ) [protected]`

methods to be called inside the _createInternalAndAttachedData() method

Returns a list of keys (names) of internal ModelDatas, cuch as Counters, StatisticsCollectors and others. Children↩ Elements are ModelDatas used by this ModelDataDefinition thar are needed before model checking Here is the call graph for this function:

3.78.3.13 _internalDataInsert() `void ModelDataDefinition::_internalDataInsert ( std::string key, ModelDataDefinition * child ) [protected]`

3.78.3.14 _internalDataRemove()

```
void ModelDataDefinition::_internalDataRemove ( std::string key ) [protected]
```

Here is the call graph for this function:

3.78.3.15 _loadInstance()

```
bool ModelDataDefinition::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Implements PersistentObject_base.

Reimplemented in Attribute, Counter, Entity, EntityType, ModelComponent, SinkModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, DummyElement, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, Sequence, Set, SignalData, Station, Storage, and Variable.

Here is the call graph for this function:

3.78.3.16 _saveInstance()

```
void ModelDataDefinition::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Implements PersistentObject_base.

Reimplemented in Attribute, Counter, Entity, EntityType, ModelComponent, SinkModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, DummyElement, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, Sequence, Set, SignalData, Station, Storage, and Variable.

Here is the call graph for this function:

3.78.3.17 Check()

```
bool ModelDataDefinition::Check (
    ModelDataDefinition * modeldatum,
    std::string * errorMessage )  [static]
```

This class method takes an instance of a ModelDataDefinition and invokes the private method_check() method of that instance, which checks itself Here is the call graph for this function:

3.78.3.18 CreateInternalData()

```
void ModelDataDefinition::CreateInternalData (
    ModelDataDefinition * modeldatum )  [static]
```

This class method is responsible for invoking the protected method _check() of the instance modeldatum, which creates any internal ModelDataDefinition (such as internelElements) or even other external needed ModelDatas, such as attributes or variables Here is the call graph for this function:

3.78.3.19 getAttachedData()

`std::map< std::string, ModelDataDefinition * > * ModelDataDefinition↩ ::getAttachedData ( ) const`

3.78.3.20 getClassname()

`std::string ModelDataDefinition::getClassname ( ) const`

3.78.3.21 getId()

`Util::identification ModelDataDefinition::getId ( ) const`

3.78.3.22 getInternalData() [1/2]

`std::map< std::string, ModelDataDefinition * > * ModelData↩ Definition::getInternalData ( ) const`

3.78.3.23 getInternalData() [2/2]

`ModelDataDefinition * ModelDataDefinition::getInternalData ( std::string name ) const`

3.78.3.24 getLevel()

`unsigned int ModelDataDefinition::getLevel ( ) const`

3.78.3.25 getName()

`std::string ModelDataDefinition::getName ( ) const`

3.78.3.26 getProperties() `List< PropertyBase * > * ModelDataDefinition::getProperties ( ) const`

3.78.3.27 hasChanged() `bool ModelDataDefinition::hasChanged ( ) const`

3.78.3.28 InitBetweenReplications() `void ModelDataDefinition::InitBetweenReplications ( ModelDataDefinition * modeldatum ) [static]`

Here is the call graph for this function:

3.78.3.29 isReportStatistics() `bool ModelDataDefinition::isReportStatistics ( ) const`

Return true if this ModelDataDefinition generates statics for simulation reports

3.78.3.30 LoadInstance() `ModelDataDefinition * ModelDataDefinition::LoadInstance ( Model * model, PersistenceRecord * fields, bool insertIntoModel ) [static]`

This class method receives a map of fields readed from a file (or somewhere else) creates an instace of the Model↩ Datas and inokes the protected method _loadInstance() of that instance, whch fills the field values. The instance can be automatticaly inserted into the simulation model if required Here is the call graph for this function:

3.78.3.31 NewInstance()

```
ModelDataDefinition * ModelDataDefinition::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

This class method invokes the constructor and returns a new instance (that demands a typecast to the rigth subclass). It is used to construct a new instance when plugins are connected using dynamic loaded libraries

3.78.3.32 SaveInstance()

```
void ModelDataDefinition::SaveInstance (
    PersistenceRecord * fields,
    ModelDataDefinition * modeldatum )  [static]
```

This class method takes an instance of a ModelDataDefinition, invokes the protected method _saveInstance() of that instance and retorns a map of filds (name=value) that can be saved on a file (or somewhere else) Here is the call graph for this function:

3.78.3.33 setModelLevel()

```
void ModelDataDefinition::setModelLevel (
    unsigned int _modelLevel )
```

3.78.3.34 setName()

```
void ModelDataDefinition::setName (
    std::string name )
```

Here is the call graph for this function:

3.78.3.35 setReportStatistics()

```
void ModelDataDefinition::setReportStatistics (
    bool reportStatistics )
```

Defnes if this ModelDataDefinition generates statics for simulation reports

3.78.3.36 show()

```
std::string ModelDataDefinition::show ( ) [virtual]
```

Reimplemented in Attribute, Counter, Entity, EntityType, ModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, DummyElement, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, Sequence, Set, SignalData, Station, Storage, and Variable.

3.78.3.37 trace()

```
void ModelDataDefinition::trace (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed ) [protected]
```

just an easy access to trace manager

Here is the call graph for this function:

3.78.3.38 traceError() [1/2]

```
void ModelDataDefinition::traceError (
    std::string text,
    std::exception e ) [protected]
```

Here is the call graph for this function:

3.78.3.39 traceError() [2/2]

```
void ModelDataDefinition::traceError (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L1_errorFatal ) [protected]
```

Here is the call graph for this function:

3.78.3.40 traceReport()

```
void ModelDataDefinition::traceReport (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L2_results ) [protected]
```

Here is the call graph for this function:

3.78.3.41 traceSimulation() [1/2]

```
void ModelDataDefinition::traceSimulation (
    void * thisobject,
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed ) [protected]
```

Here is the call graph for this function:

3.78.3.42 traceSimulation() [2/2]

```
void ModelDataDefinition::traceSimulation (
    void * thisobject,
    TraceManager::Level level,
    std::string text ) [protected]
```

Here is the call graph for this function:

3.78.4 Member Data Documentation

3.78.4.1 _hasChanged

```
bool ModelDataDefinition::_hasChanged [protected]
```

3.78.4.2 _id

```
Util::identification ModelDataDefinition::_id [protected]
```

3.78.4.3 _modelLevel

```
unsigned int ModelDataDefinition::_modelLevel = 0 [protected]
```

3.78.4.4 _parentModel

```
Model* ModelDataDefinition::_parentModel [protected]
```

3.78.4.5 _properties

```
List<PropertyBase*>* ModelDataDefinition::_properties = new List<PropertyBase*>()
[protected]
```

3.78.4.6 _reportStatistics

```
bool ModelDataDefinition::_reportStatistics [protected]
```

3.78.4.7 _typename

```
std::string ModelDataDefinition::_typename [protected]
```

3.79 ModelDataManager Class Reference

```
#include <ModelDataManager.h>
```

Public Member Functions

- ModelDataManager (Model *model)
- virtual ~ModelDataManager ()=default
- bool insert (ModelDataDefinition *anElement)
- void remove (ModelDataDefinition *anElement)

  Deprected.
- bool insert (std::string datadefinitionTypename, ModelDataDefinition *anElement)

  Deprected.
- void remove (std::string datadefinitionTypename, ModelDataDefinition *anElement)
- bool check (std::string datadefinitionTypename, ModelDataDefinition *anElement, std::string expression↩ Name, std::string *errorMessage)
- bool check (std::string datadefinitionTypename, std::string elementName, std::string expressionName, bool mandatory, std::string *errorMessage)
- void clear ()
- ModelDataDefinition * getDataDefinition (std::string datadefinitionTypename, Util::identification id)
- ModelDataDefinition * getDataDefinition (std::string datadefinitionTypename, std::string name)
- unsigned int getNumberOfDataDefinitions (std::string datadefinitionTypename)
- unsigned int getNumberOfDataDefinitions ()
- int getRankOf (std::string datadefinitionTypename, std::string name)

  returns the position (1st position=0) of the modeldatum if found, or negative value if not found
- std::list< std::string > * getDataDefinitionClassnames () const
- List< ModelDataDefinition * > * getDataDefinitionList (std::string datadefinitionTypename) const
- void show ()
- Model * getParentModel () const
- bool hasChanged () const
- void setHasChanged (bool _hasChanged)

3.79.1 Detailed Description

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

3.79.2 Constructor & Destructor Documentation

3.79.2.1 ModelDataManager() `ModelDataManager::ModelDataManager ( Model * model )`

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

3.79.2.2 ~ModelDataManager() `virtual ModelDataManager::~ModelDataManager ( ) [virtual], [default]`

3.79.3 Member Function Documentation

3.79.3.1 check() [1/2]

```
bool ModelDataManager::check (
    std::string datadefinitionTypename,
    ModelDataDefinition * anElement,
    std::string expressionName,
    std::string * errorMessage )
```

Here is the call graph for this function:

3.79.3.2 check() [2/2]

```
bool ModelDataManager::check (
    std::string datadefinitionTypename,
    std::string elementName,
    std::string expressionName,
    bool mandatory,
    std::string * errorMessage )
```

Here is the call graph for this function:

3.79.3.3 clear()

```
void ModelDataManager::clear ( )
```

3.79.3.4 getDataDefinition() [1/2]

```
ModelDataDefinition * ModelDataManager::getDataDefinition (
    std::string datadefinitionTypename,
    std::string name )
```

Here is the call graph for this function:

3.79.3.5 getDataDefinition() [2/2]

```
ModelDataDefinition * ModelDataManager::getDataDefinition (
    std::string datadefinitionTypename,
    Util::identification id )
```

Here is the call graph for this function:

3.79.3.6 getDataDefinitionClassnames()

```
std::list< std::string > * ModelDataManager::getDataDefinitionClassnames ( ) const
```

3.79.3.7 getDataDefinitionList()

```
List< ModelDataDefinition * > * ModelDataManager::getDataDefinitionList (
    std::string datadefinitionTypename ) const
```

Here is the call graph for this function:

3.79.3.8 getNumberOfDataDefinitions() [1/2]

```
unsigned int ModelDataManager::getNumberOfDataDefinitions ( )
```

3.79.3.9 getNumberOfDataDefinitions() [2/2]

```
unsigned int ModelDataManager::getNumberOfDataDefinitions (
    std::string datadefinitionTypename )
```

Here is the call graph for this function:

3.79.3.10 getParentModel() `Model * ModelDataManager::getParentModel ( ) const`

3.79.3.11 getRankOf()

```
int ModelDataManager::getRankOf (
    std::string datadefinitionTypename,
    std::string name )
```

returns the position (1st position=0) of the modeldatum if found, or negative value if not found

Here is the call graph for this function:

3.79.3.12 hasChanged() `bool ModelDataManager::hasChanged ( ) const`

3.79.3.13 insert() [1/2]

```
bool ModelDataManager::insert (
    ModelDataDefinition * anElement )
```

Here is the call graph for this function:

3.79.3.14 insert() [2/2]

```
bool ModelDataManager::insert (
    std::string datadefinitionTypename,
    ModelDataDefinition * anElement )
```

Deprected.

Here is the call graph for this function:

3.79.3.15 remove() [1/2]

```
void ModelDataManager::remove (
    ModelDataDefinition * anElement )
```

Deprected.

Here is the call graph for this function:

3.79.3.16 remove() [2/2]

```
void ModelDataManager::remove (
    std::string datadefinitionTypename,
    ModelDataDefinition * anElement )
```

Here is the call graph for this function:

3.79.3.17 setHasChanged() `void ModelDataManager::setHasChanged ( bool _hasChanged )`

3.79.3.18 show() `void ModelDataManager::show ( )`

Here is the call graph for this function:

3.80 ModelEvent Class Reference

`#include <OnEventManager.h>`

Public Member Functions

- ModelEvent (Model *model)
- const Model * getModel ()

3.80.1 Detailed Description

Stores an event that happened on a specific model

3.80.2 Constructor & Destructor Documentation

3.80.2.1 ModelEvent() `ModelEvent::ModelEvent ( Model * model ) [inline]`

3.80.3 Member Function Documentation

3.80.3.1 getModel() `const Model * ModelEvent::getModel ( ) [inline]`

3.81 ModelInfo Class Reference

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

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 (PersistenceRecord *fields)
- void saveInstance (PersistenceRecord *fields)
- bool hasChanged () const

3.81.1 Detailed Description

ModelInfo stores basic model project information, such as the project and analyst names.

3.81.2 Constructor & Destructor Documentation

3.81.2.1 ModelInfo() `ModelInfo::ModelInfo ( )`

3.81.2.2 ~ModelInfo() `virtual ModelInfo::~ModelInfo ( ) [virtual], [default]`

3.81.3 Member Function Documentation

3.81.3.1 getAnalystName() `std::string ModelInfo::getAnalystName ( ) const`

3.81.3.2 getDescription() `std::string ModelInfo::getDescription ( ) const`

3.81.3.3 getName() `std::string ModelInfo::getName ( ) const`

3.81.3.4 getProjectTitle() `std::string ModelInfo::getProjectTitle ( ) const`

3.81.3.5 getVersion() `std::string ModelInfo::getVersion ( ) const`

3.81.3.6 hasChanged() `bool ModelInfo::hasChanged ( ) const`

3.81.3.7 loadInstance()

```
void ModelInfo::loadInstance (
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.81.3.8 saveInstance()

```
void ModelInfo::saveInstance (
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.81.3.9 setAnalystName()

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

3.81.3.10 setDescription()

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

3.81.3.11 setName()

```
void ModelInfo::setName (
    std::string _name )
```

3.81.3.12 setProjectTitle()

```
void ModelInfo::setProjectTitle (
    std::string _projectTitle )
```

3.81.3.13 setVersion()

```
void ModelInfo::setVersion (
    std::string _version )
```

3.81.3.14 show()

```
std::string ModelInfo::show ( )
```

3.82 ModelManager Class Reference

```
#include <ModelManager.h>
```

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)
- Model * loadModel (std::string filename)
- Model * createFromLanguage (std::string modelSpecification)
- unsigned int size ()
- Model * front ()
- Model * current ()
- Model * next ()

3.82.1 Constructor & Destructor Documentation

3.82.1.1 ModelManager()

```
ModelManager::ModelManager (
    Simulator * simulator )
```

3.82.1.2 ~ModelManager()

```
virtual ModelManager::~ModelManager ( ) [virtual], [default]
```

3.82.2 Member Function Documentation

3.82.2.1 createFromLanguage()

```
Model * ModelManager::createFromLanguage (
    std::string modelSpecification )
```

Here is the call graph for this function:

3.82.2.2 current() `Model * ModelManager::current ( )`

3.82.2.3 front() `Model * ModelManager::front ( )`

3.82.2.4 insert() `void ModelManager::insert ( Model * model )`

Here is the call graph for this function:

3.82.2.5 loadModel() `Model * ModelManager::loadModel ( std::string filename )`

Here is the call graph for this function:

3.82.2.6 newModel() `Model * ModelManager::newModel ( )`

3.82.2.7 next() `Model * ModelManager::next ( )`

3.82.2.8 remove() `void ModelManager::remove ( Model * model )`

Here is the call graph for this function:

3.82.2.9 saveModel() `bool ModelManager::saveModel ( std::string filename )`

Here is the call graph for this function:

3.82.2.10 setCurrent() `void ModelManager::setCurrent ( Model * model )`

3.82.2.11 size() `unsigned int ModelManager::size ( )`

3.83 ModelPersistence_if Class Reference

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

Inheritance diagram for ModelPersistence_if:

Public Types

- enum class Options : int {
  SAVEDEFAULTS = 1 , HIDEIDKEY = 2 , HIDETYPEKEY = 4 , HIDENAMEKEY = 8 ,
  SORTALPHLY = 16 }

Public Member Functions

- virtual bool save (std::string filename)=0
- virtual bool load (std::string filename)=0
- virtual bool hasChanged ()=0
- virtual bool getOption (ModelPersistence_if::Options option)=0
- virtual void setOption (ModelPersistence_if::Options option, bool value)=0
- virtual std::string getFormatedField (PersistenceRecord *fields)=0

3.83.1 Member Enumeration Documentation

3.83.1.1 Options `enum class ModelPersistence_if::Options : int [strong]`

Enumerator

SAVEDEFAULTS	
HIDEIDKEY	
HIDETYPEKEY	
HIDENAMEKEY	
SORTALPHLY	

3.83.2 Member Function Documentation

3.83.2.1 getFormatedField() `virtual std::string ModelPersistence_if::getFormatedField ( PersistenceRecord * fields ) [pure virtual]`

Implemented in ModelPersistenceDefaultImpl2.

3.83.2.2 getOption() `virtual bool ModelPersistence_if::getOption ( ModelPersistence_if::Options option ) [pure virtual]`

Implemented in ModelPersistenceDefaultImpl2.

3.83.2.3 hasChanged() `virtual bool ModelPersistence_if::hasChanged ( ) [pure virtual]`

Implemented in ModelPersistenceDefaultImpl2.

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

Implemented in ModelPersistenceDefaultImpl2.

3.83.2.5 save()

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

Implemented in ModelPersistenceDefaultImpl2.

3.83.2.6 setOption()

```
virtual void ModelPersistence_if::setOption (
    ModelPersistence_if::Options option,
    bool value ) [pure virtual]
```

Implemented in ModelPersistenceDefaultImpl2.

3.84 ModelPersistenceDefaultImpl2 Class Reference

```
#include <ModelPersistenceDefaultImpl2.h>
```

Inheritance diagram for ModelPersistenceDefaultImpl2:

Public Member Functions

- ModelPersistenceDefaultImpl2 (Model *model)
- bool save (std::string filename) override
- bool load (std::string filename) override
- bool hasChanged () override
- bool getOption (ModelPersistence_if::Options option) override
- void setOption (ModelPersistence_if::Options option, bool value) override
- std::string getFormatedField (PersistenceRecord *fields) override

Additional Inherited Members

3.84.1 Constructor & Destructor Documentation

3.84.1.1 ModelPersistenceDefaultImpl2()

```
ModelPersistenceDefaultImpl2::ModelPersistenceDefault↩
Impl2 (
            Model * model )
```

3.84.2 Member Function Documentation

3.84.2.1 getFormatedField()

```
std::string ModelPersistenceDefaultImpl2::getFormatedField (
            PersistenceRecord * fields )  [override], [virtual]
```

Implements ModelPersistence_if.

3.84.2.2 getOption()

```
bool ModelPersistenceDefaultImpl2::getOption (
            ModelPersistence_if::Options option )  [override], [virtual]
```

Implements ModelPersistence_if.

3.84.2.3 hasChanged()

```
bool ModelPersistenceDefaultImpl2::hasChanged ( )  [override], [virtual]
```

Implements ModelPersistence_if.

3.84.2.4 load() `bool ModelPersistenceDefaultImpl2::load ( std::string filename ) [override], [virtual]`

Implements ModelPersistence_if.

Here is the call graph for this function:

3.84.2.5 save() `bool ModelPersistenceDefaultImpl2::save ( std::string filename ) [override], [virtual]`

Implements ModelPersistence_if.

Here is the call graph for this function:

3.84.2.6 setOption()

```
void ModelPersistenceDefaultImpl2::setOption (
    ModelPersistence_if::Options option,
    bool value ) [override], [virtual]
```

Implements ModelPersistence_if.

3.85 ModelSerializer Class Reference

Serialization/Deserialization interface for simulation models.

```
#include <ModelSerializer.h>
```

Inheritance diagram for ModelSerializer:

Public Member Functions

- virtual ~ModelSerializer ()
- virtual PersistenceRecord * newPersistenceRecord ()=0

  Creates an empty persistence record.
- virtual bool dump (std::ostream &output)=0

  Writes formatted contents to a given output stream.
- virtual bool load (std::istream &input)=0

  Attempts to load contents from a given input stream.
- virtual bool get (const std::string &name, PersistenceRecord *entry)=0

  Tries to fetch an entry with the given label as a key.
- virtual bool put (const std::string name, const std::string type, const Util::identification id, PersistenceRecord *fields)=0

  Upserts an entry into this structure.
- virtual int for_each (std::function< int(const std::string &) > delegate)=0

  Iterates over this structure's entries, ordered by ID.

3.85.1 Detailed Description

Serialization/Deserialization interface for simulation models.

This can be seen as an associative array which knows how to ser/deserialize itself. The "contents" of a serializer are the set of entries it contains.

Its entries are identified by strings, which correspond to unique component/datadef names. Each entry has its own name, a type, an id, and an associated set of fields. All of these are contained within a `PersistenceRecord` instance.

Serializers also act as factories for `PersistenceRecord`s, which can then only be used with the serializer which originated them.

3.85.2 Constructor & Destructor Documentation

3.85.2.1 ~ModelSerializer() `ModelSerializer::~ModelSerializer ( ) [virtual]`

3.85.3 Member Function Documentation

3.85.3.1 dump() `virtual bool ModelSerializer::dump ( std::ostream & output ) [pure virtual]`

Writes formatted contents to a given output stream.

Parameters

output	output stream to write to.

Returns

whether the current state of this structure's contents was deemed valid for serialization.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.85.3.2 for_each() `virtual int ModelSerializer::for_each ( std::function< int(const std::string &) > delegate ) [pure virtual]`

Iterates over this structure's entries, ordered by ID.

Parameters

delegate	delegate which is applied to every entry-identifying key, until it returns a non-zero value.

Returns

either zero, or the first non-zero value returned by the delegate.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.85.3.3 get() `virtual bool ModelSerializer::get ( const std::string & name, PersistenceRecord * entry ) [pure virtual]`

Tries to fetch an entry with the given label as a key.

Parameters

name	a label which uniquely identifies an entry.
entry	(out-param) a pointer to a caller-owned entry, which is filled on success.

Returns

whether the entry was found and the out-param was actually written to.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.85.3.4 load()

```
virtual bool ModelSerializer::load (
    std::istream & input ) [pure virtual]
```

Attempts to load contents from a given input stream.

Returns

whether the input was well-formatted and loading was successful.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.85.3.5 newPersistenceRecord()

```
virtual PersistenceRecord * ModelSerializer::newPersistence↩
Record ( ) [pure virtual]
```

Creates an empty persistence record.

Although the returned record can be freed normally with `delete`, care must be taken to only use records which come from a implementation, when configuring that implementation's contents.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.85.3.6 put()

```
virtual bool ModelSerializer::put (
    const std::string name,
    const std::string type,
    const Util::identification id,
    PersistenceRecord * fields ) [pure virtual]
```

Upserts an entry into this structure.

Parameters

name	a label which uniquely identifies the entry.
type	a type name.
id	a component/datadef ID number.
fields	a persistence record for the entry, whose contents will be copied if valid.

Returns

whether the entry was valid and was actually copied into this structure.

Implemented in CppSerializer, GenSerializer, JsonSerializer, and XmlSerializer.

3.86 ModelSimulation Class Reference

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

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, Util::TimeUnit _replicationLengthTimeUnit=Util::TimeUnit::unknown)
- double getReplicationLength () const
- void setReplicationLengthTimeUnit (Util::TimeUnit _replicationLengthTimeUnit)
- Util::TimeUnit getReplicationLengthTimeUnit () const
- void setReplicationReportBaseTimeUnit (Util::TimeUnit _replicationReportBaseTimeUnit)
- Util::TimeUnit getReplicationBaseTimeUnit () const
- void setWarmUpPeriod (double warmUpPeriod, Util::TimeUnit warmUpPeriodTimeUnit=Util::TimeUnit::unknown)
- 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
- 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 (PersistenceRecord *fields)
- void saveInstance (PersistenceRecord *fields, bool saveDefaults)
- Event * getCurrentEvent () const
- void setShowSimulationResposesInReport (bool _showSimulationResposesInReport)
- bool isShowSimulationResposesInReport () const
- void setShowSimulationControlsInReport (bool _showSimulationControlsInReport)
- bool isShowSimulationControlsInReport () const

Friends

- class Model

3.86.1 Detailed Description

The ModelSimulation controls the simulation of a model, alowing to start, pause, resume e stop a simulation, composed by a set of replications.

3.86.2 Constructor & Destructor Documentation

3.86.2.1 ModelSimulation()

```
ModelSimulation::ModelSimulation (
    Model * model )
```

Here is the call graph for this function:

3.86.2.2 ~ModelSimulation() `virtual ModelSimulation::~ModelSimulation ( ) [virtual], [default]`

3.86.3 Member Function Documentation

3.86.3.1 getBreakpointsOnComponent() `List< ModelComponent * > * ModelSimulation::getBreakpoints↩ OnComponent ( ) const`

3.86.3.2 getBreakpointsOnEntity()

`List< Entity * > * ModelSimulation::getBreakpointsOnEntity ( ) const`

3.86.3.3 getBreakpointsOnTime()

`List< double > * ModelSimulation::getBreakpointsOnTime ( ) const`

3.86.3.4 getCurrentEvent()

`Event * ModelSimulation::getCurrentEvent ( ) const`

3.86.3.5 getCurrentReplicationNumber()

`unsigned int ModelSimulation::getCurrentReplication↩ Number ( ) const`

3.86.3.6 getNumberOfReplications()

`unsigned int ModelSimulation::getNumberOfReplications ( ) const`

3.86.3.7 getReplicationBaseTimeUnit()

`Util::TimeUnit ModelSimulation::getReplicationBaseTime↩ Unit ( ) const`

3.86.3.8 getReplicationLength()

`double ModelSimulation::getReplicationLength ( ) const`

3.86.3.9 getReplicationLengthTimeUnit()

`Util::TimeUnit ModelSimulation::getReplicationLength↩ TimeUnit ( ) const`

3.86.3.10 getReporter()

`SimulationReporter_if * ModelSimulation::getReporter ( ) const`

3.86.3.11 getSimulatedTime()

`double ModelSimulation::getSimulatedTime ( ) const`

3.86.3.12 getTerminatingCondition() `std::string ModelSimulation::getTerminatingCondition ( ) const`

3.86.3.13 getWarmUpPeriod() `double ModelSimulation::getWarmUpPeriod ( ) const`

3.86.3.14 getWarmUpPeriodTimeUnit() `Util::TimeUnit ModelSimulation::getWarmUpPeriodTimeUnit ( ) const`

3.86.3.15 isInitializeStatistics() `bool ModelSimulation::isInitializeStatistics ( ) const`

3.86.3.16 isInitializeSystem() `bool ModelSimulation::isInitializeSystem ( ) const`

3.86.3.17 isPaused() `bool ModelSimulation::isPaused ( ) const`

3.86.3.18 isPauseOnEvent() `bool ModelSimulation::isPauseOnEvent ( ) const`

3.86.3.19 isPauseOnReplication() `bool ModelSimulation::isPauseOnReplication ( ) const`

3.86.3.20 isRunning() `bool ModelSimulation::isRunning ( ) const`

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

3.86.3.21 isShowReportsAfterReplication() `bool ModelSimulation::isShowReportsAfterReplication ( ) const`

3.86.3.22 isShowReportsAfterSimulation() `bool ModelSimulation::isShowReportsAfterSimulation ( ) const`

3.86.3.23 isShowSimulationControlsInReport()

`bool ModelSimulation::isShowSimulationControlsIn↩Report ( ) const`

3.86.3.24 isShowSimulationResposesInReport()

`bool ModelSimulation::isShowSimulationResposes↩InReport ( ) const`

3.86.3.25 isStepByStep()

`bool ModelSimulation::isStepByStep ( ) const`

3.86.3.26 loadInstance()

```
void ModelSimulation::loadInstance (
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.86.3.27 pause()

`void ModelSimulation::pause ( )`

3.86.3.28 saveInstance()

```
void ModelSimulation::saveInstance (
    PersistenceRecord * fields,
    bool saveDefaults )
```

Here is the call graph for this function:

3.86.3.29 setInitializeStatistics()

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

3.86.3.30 setInitializeSystem()

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

3.86.3.31 setNumberOfReplications()

```
void ModelSimulation::setNumberOfReplications (
    unsigned int _numberOfReplications )
```

3.86.3.32 setPauseOnEvent()

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

3.86.3.33 setPauseOnReplication()

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

3.86.3.34 setReplicationLength()

```
void ModelSimulation::setReplicationLength (
    double _replicationLength,
    Util::TimeUnit _replicationLengthTimeUnit = Util::TimeUnit::unknown )
```

3.86.3.35 setReplicationLengthTimeUnit()

```
void ModelSimulation::setReplicationLengthTimeUnit (
    Util::TimeUnit _replicationLengthTimeUnit )
```

3.86.3.36 setReplicationReportBaseTimeUnit()

```
void ModelSimulation::setReplicationReportBase↩
TimeUnit (
    Util::TimeUnit _replicationReportBaseTimeUnit )
```

3.86.3.37 setReporter()

```
void ModelSimulation::setReporter (
    SimulationReporter_if * _simulationReporter )
```

3.86.3.38 setShowReportsAfterReplication()

```
void ModelSimulation::setShowReportsAfterReplication (
    bool showReportsAfterReplication )
```

3.86.3.39 setShowReportsAfterSimulation()

```
void ModelSimulation::setShowReportsAfterSimulation (
    bool showReportsAfterSimulation )
```

3.86.3.40 setShowSimulationControlsInReport()

```
void ModelSimulation::setShowSimulationControls↩
InReport (
    bool _showSimulationControlsInReport )
```

3.86.3.41 setShowSimulationResposesInReport()

```
void ModelSimulation::setShowSimulationResposes↩
InReport (
    bool _showSimulationResposesInReport )
```

3.86.3.42 setStepByStep()

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

3.86.3.43 setTerminatingCondition()

```
void ModelSimulation::setTerminatingCondition (
    std::string _terminatingCondition )
```

3.86.3.44 setWarmUpPeriod()

```
void ModelSimulation::setWarmUpPeriod (
    double warmUpPeriod,
    Util::TimeUnit warmUpPeriodTimeUnit = Util::TimeUnit::unknown )
```

3.86.3.45 setWarmUpPeriodTimeUnit()

```
void ModelSimulation::setWarmUpPeriodTimeUnit (
    Util::TimeUnit _warmUpPeriodTimeUnit )
```

3.86.3.46 show() `std::string ModelSimulation::show ( )`

Here is the call graph for this function:

3.86.3.47 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 Here is the call graph for this function:

3.86.3.48 step() `void ModelSimulation::step ( )`

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

Here is the call graph for this function:

3.86.3.49 stop() `void ModelSimulation::stop ( )`

3.86.4 Friends And Related Function Documentation

3.86.4.1 Model `friend class Model [friend]`

3.87 ParserManager::NewParser Struct Reference

```
#include <ParserManager.h>
```

Public Attributes

- std::string bisonFilename
- std::string flexFilename
- std::string compiledParserFilename

3.87.1 Member Data Documentation

3.87.1.1 bisonFilename `std::string ParserManager::NewParser::bisonFilename`

3.87.1.2 compiledParserFilename `std::string ParserManager::NewParser::compiledParserFilename`

3.87.1.3 flexFilename `std::string ParserManager::NewParser::flexFilename`

3.88 obj_t Class Reference

```
#include <obj_t.h>
```

Public Member Functions

- obj_t ()
- virtual ~obj_t ()
- obj_t (double v, std::string t)
- obj_t (double v, std::string t, unsigned long uid)

Public Attributes

- double valor
- std::string tipo
- unsigned long id

3.88.1 Constructor & Destructor Documentation

3.88.1.1 obj_t() [1/3] `obj_t::obj_t ( )`

3.88.1.2 ~obj_t() `obj_t::~obj_t ( ) [virtual]`

3.88.1.3 obj_t() [2/3]

```
obj_t::obj_t (
    double v,
    std::string t )
```

3.88.1.4 obj_t() [3/3]

```
obj_t::obj_t (
    double v,
    std::string t,
    unsigned long uid )
```

3.88.2 Member Data Documentation

3.88.2.1 id

```
unsigned long obj_t::id
```

3.88.2.2 tipo

```
std::string obj_t::tipo
```

3.88.2.3 valor

```
double obj_t::valor
```

3.89 ODEfunction Class Reference

```
#include <OLD_ODEelement.h>
```

Public Member Functions

- ODEfunction (std::string expression, double initialPoint, double initialValue)

Public Attributes

- std::string expression
- double initialPoint
- double initialValue

3.89.1 Constructor & Destructor Documentation

3.89.1.1 ODEfunction()

```
ODEfunction::ODEfunction (
    std::string expression,
    double initialPoint,
    double initialValue ) [inline]
```

3.89.2 Member Data Documentation

3.89.2.1 expression

```
std::string ODEfunction::expression
```

3.89.2.2 initialPoint

```
double ODEfunction::initialPoint
```

3.89.2.3 initialValue

```
double ODEfunction::initialValue
```

3.90 OLD_ODEelement Class Reference

```
#include <OLD_ODEelement.h>
```

Inheritance 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 ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.90.1 Constructor & Destructor Documentation

3.90.1.1 OLD_ODEelement()

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

3.90.1.2 ~OLD_ODEelement()

```
virtual OLD_ODEelement::~OLD_ODEelement ( ) [virtual], [default]
```

3.90.2 Member Function Documentation

3.90.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.90.2.2 _loadInstance()

```
bool OLD_ODEelement::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.90.2.3 _saveInstance()

```
void OLD_ODEelement::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.90.2.4 getEndTime() `double OLD_ODEelement::getEndTime ( ) const`

3.90.2.5 getODEfunctions() `List< ODEfunction * > * OLD_ODEelement::getODEfunctions ( ) const`

3.90.2.6 GetPluginInformation() `PluginInformation * OLD_ODEelement::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.90.2.7 getStepH() `double OLD_ODEelement::getStepH ( ) const`

3.90.2.8 LoadInstance() `ModelDataDefinition * OLD_ODEelement::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.90.2.9 NewInstance()

```
ModelDataDefinition * OLD_ODEelement::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.90.2.10 setEndTime()

```
void OLD_ODEelement::setEndTime (
    double _endTime )
```

3.90.2.11 setStepH()

```
void OLD_ODEelement::setStepH (
    double _h )
```

3.90.2.12 show()

```
std::string OLD_ODEelement::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.90.2.13 solve()

```
double OLD_ODEelement::solve ( )
```

3.91 OnEventManager Class Reference

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

Public Member Functions

- OnEventManager ()
- virtual ~OnEventManager ()=default
- void addOnModelCheckSucessHandler (modelEventHandler EventHandler)
- void addOnModelLoadHandler (modelEventHandler EventHandler)
- void addOnModelSaveHandler (modelEventHandler EventHandler)
- void addOnReplicationStartHandler (simulationEventHandler EventHandler)
- void addOnReplicationStepHandler (simulationEventHandler EventHandler)
- void addOnReplicationEndHandler (simulationEventHandler EventHandler)
- void addOnProcessEventHandler (simulationEventHandler EventHandler)
- void addOnAfterProcessEventHandler (simulationEventHandler EventHandler)
- void addOnEntityCreateHandler (simulationEventHandler EventHandler)
- void addOnEntityMoveHandler (simulationEventHandler EventHandler)
- void addOnEntityRemoveHandler (simulationEventHandler EventHandler)
- void addOnSimulationStartHandler (simulationEventHandler EventHandler)
- void addOnSimulationPausedHandler (simulationEventHandler EventHandler)
- void addOnSimulationResumeHandler (simulationEventHandler EventHandler)
- void addOnSimulationEndHandler (simulationEventHandler EventHandler)
- void addOnBreakpointHandler (simulationEventHandler EventHandler)
- template<typename Class >
  void addOnModelCheckSuccessHandler (Class *object, void(Class::*function)(ModelEvent *))
- template<typename Class >
  void addOnModelLoadHandler (Class *object, void(Class::*function)(ModelEvent *))
- template<typename Class >
  void addOnModelSaveHandler (Class *object, void(Class::*function)(ModelEvent *))
- template<typename Class >
  void addOnReplicationStartHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnReplicationStepHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnReplicationEndHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnProcessEventHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnAfterProcessEventHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnEntityCreateHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnEntityMoveHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnEntityRemoveHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnSimulationStartHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnSimulationPausedHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnSimulationResumeHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnSimulationEndHandler (Class *object, void(Class::*function)(SimulationEvent *))
- template<typename Class >
  void addOnBreakpointHandler (Class *object, void(Class::*function)(SimulationEvent *))
- void NotifyModelCheckSuccessHandlers (ModelEvent *se)
- void NotifyModelLoadHandlers (ModelEvent *se)
- void NotifyModelSaveHandlers (ModelEvent *se)
- void NotifyReplicationStartHandlers (SimulationEvent *se)

- void NotifyReplicationStepHandlers (SimulationEvent *se)
- void NotifyReplicationEndHandlers (SimulationEvent *se)
- void NotifyProcessEventHandlers (SimulationEvent *se)
- void NotifyAfterProcessEventHandlers (SimulationEvent *se)
- void NotifyEntityCreateHandlers (SimulationEvent *se)
- void NotifyEntityMoveHandlers (SimulationEvent *se)
- void NotifyEntityRemoveHandlers (SimulationEvent *se)
- void NotifySimulationStartHandlers (SimulationEvent *se)
- void NotifySimulationPausedHandlers (SimulationEvent *se)
- void NotifySimulationResumeHandlers (SimulationEvent *se)
- void NotifySimulationEndHandlers (SimulationEvent *se)
- void NotifyBreakpointHandlers (SimulationEvent *se)

3.91.1 Detailed Description

OnEventManager allows external methods to hook interval simulation events as listeners (or observers) of pecific events. All methods added as listeners of an event will be invovked when that event is triggered.

3.91.2 Constructor & Destructor Documentation

3.91.2.1 OnEventManager() `OnEventManager::OnEventManager ( )`

3.91.2.2 ~OnEventManager() `virtual OnEventManager::~OnEventManager ( ) [virtual], [default]`

3.91.3 Member Function Documentation

3.91.3.1 addOnAfterProcessEventHandler() [1/2] `template<typename Class >`

```
void OnEventManager::addOnAfterProcessEventHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.2 addOnAfterProcessEventHandler() [2/2] `void OnEventManager::addOnAfterProcessEvent↩`

```
Handler (
            simulationEventHandler EventHandler )
```

3.91.3.3 addOnBreakpointHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnBreakpointHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.4 addOnBreakpointHandler() [2/2] void OnEventManager::addOnBreakpointHandler (

```
            simulationEventHandler EventHandler )
```

3.91.3.5 addOnEntityCreateHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnEntityCreateHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.6 addOnEntityCreateHandler() [2/2] void OnEventManager::addOnEntityCreateHandler (

```
            simulationEventHandler EventHandler )
```

3.91.3.7 addOnEntityMoveHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnEntityMoveHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.8 addOnEntityMoveHandler() [2/2] void OnEventManager::addOnEntityMoveHandler (

```
            simulationEventHandler EventHandler )
```

3.91.3.9 addOnEntityRemoveHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnEntityRemoveHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.10 addOnEntityRemoveHandler() [2/2] void OnEventManager::addOnEntityRemoveHandler (

```
            simulationEventHandler EventHandler )
```

3.91.3.11 addOnModelCheckSuccessHandler() template<typename Class >

```
void OnEventManager::addOnModelCheckSuccessHandler (
            Class * object,
            void(Class::*)(ModelEvent *) function )
```

3.91.3.12 addOnModelCheckSucessHandler() void OnEventManager::addOnModelCheckSucessHandler (

```
            modelEventHandler EventHandler )
```

3.91.3.13 addOnModelLoadHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnModelLoadHandler (
            Class * object,
            void(Class::*)(ModelEvent *) function )
```

3.91.3.14 addOnModelLoadHandler() [2/2] void OnEventManager::addOnModelLoadHandler (

```
            modelEventHandler EventHandler )
```

3.91.3.15 addOnModelSaveHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnModelSaveHandler (
            Class * object,
            void(Class::*)(ModelEvent *) function )
```

3.91.3.16 addOnModelSaveHandler() [2/2] void OnEventManager::addOnModelSaveHandler (

```
            modelEventHandler EventHandler )
```

3.91.3.17 addOnProcessEventHandler() [1/2] template<typename Class >

```
void OnEventManager::addOnProcessEventHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.18 addOnProcessEventHandler() [2/2] void OnEventManager::addOnProcessEventHandler (

```
            simulationEventHandler EventHandler )
```

3.91.3.19 addOnReplicationEndHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnReplicationEndHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.20 addOnReplicationEndHandler() [2/2]

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

3.91.3.21 addOnReplicationStartHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnReplicationStartHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.22 addOnReplicationStartHandler() [2/2]

```
void OnEventManager::addOnReplicationStartHandler
(
            simulationEventHandler EventHandler )
```

3.91.3.23 addOnReplicationStepHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnReplicationStepHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.24 addOnReplicationStepHandler() [2/2]

```
void OnEventManager::addOnReplicationStepHandler
(
            simulationEventHandler EventHandler )
```

3.91.3.25 addOnSimulationEndHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnSimulationEndHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.26 addOnSimulationEndHandler() [2/2]

```
void OnEventManager::addOnSimulationEndHandler (
            simulationEventHandler EventHandler )
```

3.91.3.27 addOnSimulationPausedHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnSimulationPausedHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.28 addOnSimulationPausedHandler() [2/2]

```
void OnEventManager::addOnSimulationPaused↩
Handler (
            simulationEventHandler EventHandler )
```

3.91.3.29 addOnSimulationResumeHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnSimulationResumeHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.30 addOnSimulationResumeHandler() [2/2]

```
void OnEventManager::addOnSimulationResume↩
Handler (
            simulationEventHandler EventHandler )
```

3.91.3.31 addOnSimulationStartHandler() [1/2]

```
template<typename Class >
void OnEventManager::addOnSimulationStartHandler (
            Class * object,
            void(Class::*)(SimulationEvent *) function )
```

3.91.3.32 addOnSimulationStartHandler() [2/2]

```
void OnEventManager::addOnSimulationStartHandler
(
            simulationEventHandler EventHandler )
```

3.91.3.33 NotifyAfterProcessEventHandlers()

```
void OnEventManager::NotifyAfterProcessEventHandlers
(
            SimulationEvent * se )
```

3.91.3.34 NotifyBreakpointHandlers()

```
void OnEventManager::NotifyBreakpointHandlers (
            SimulationEvent * se )
```

3.91.3.35 NotifyEntityCreateHandlers()

```
void OnEventManager::NotifyEntityCreateHandlers (
    SimulationEvent * se )
```

3.91.3.36 NotifyEntityMoveHandlers()

```
void OnEventManager::NotifyEntityMoveHandlers (
    SimulationEvent * se )
```

3.91.3.37 NotifyEntityRemoveHandlers()

```
void OnEventManager::NotifyEntityRemoveHandlers (
    SimulationEvent * se )
```

3.91.3.38 NotifyModelCheckSuccessHandlers()

```
void OnEventManager::NotifyModelCheckSuccess↩
Handlers (
    ModelEvent * se )
```

3.91.3.39 NotifyModelLoadHandlers()

```
void OnEventManager::NotifyModelLoadHandlers (
    ModelEvent * se )
```

3.91.3.40 NotifyModelSaveHandlers()

```
void OnEventManager::NotifyModelSaveHandlers (
    ModelEvent * se )
```

3.91.3.41 NotifyProcessEventHandlers()

```
void OnEventManager::NotifyProcessEventHandlers (
    SimulationEvent * se )
```

3.91.3.42 NotifyReplicationEndHandlers()

```
void OnEventManager::NotifyReplicationEndHandlers (
    SimulationEvent * se )
```

3.91.3.43 NotifyReplicationStartHandlers()

```
void OnEventManager::NotifyReplicationStartHandlers (
    SimulationEvent * se )
```

3.91.3.44 NotifyReplicationStepHandlers()

```
void OnEventManager::NotifyReplicationStepHandlers (
    SimulationEvent * se )
```

3.91.3.45 NotifySimulationEndHandlers()

```
void OnEventManager::NotifySimulationEndHandlers (
    SimulationEvent * se )
```

3.91.3.46 NotifySimulationPausedHandlers()

```
void OnEventManager::NotifySimulationPausedHandlers
(
    SimulationEvent * se )
```

3.91.3.47 NotifySimulationResumeHandlers()

```
void OnEventManager::NotifySimulationResumeHandlers
(
    SimulationEvent * se )
```

3.91.3.48 NotifySimulationStartHandlers()

```
void OnEventManager::NotifySimulationStartHandlers (
    SimulationEvent * se )
```

3.92 Parser_if Class Reference

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

Inheritance diagram for Parser_if:

Public Member Functions

- virtual double parse (const std::string expression)=0
- virtual double parse (const std::string expression, bool ∗success, std::string ∗errorMessage)=0
- virtual std::string ∗ getErrorMessage ()=0
- virtual void setSampler (Sampler_if ∗sampler)=0
- virtual Sampler_if ∗ getSampler () const =0
- virtual genesyspp_driver getParser () const =0

3.92.1 Member Function Documentation

3.92.1.1 getErrorMessage() `virtual std::string * Parser_if::getErrorMessage ( ) [pure virtual]`

Implemented in ParserDefaultImpl1, and ParserDefaultImpl2.

3.92.1.2 getParser() `virtual genesyspp_driver Parser_if::getParser ( ) const [pure virtual]`

Implemented in ParserDefaultImpl2.

3.92.1.3 getSampler() `virtual Sampler_if * Parser_if::getSampler ( ) const [pure virtual]`

Implemented in ParserDefaultImpl2.

3.92.1.4 parse() [1/2]

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

Implemented in ParserDefaultImpl1, and ParserDefaultImpl2.

3.92.1.5 parse() [2/2]

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

Implemented in ParserDefaultImpl1, and ParserDefaultImpl2.

3.92.1.6 setSampler()

```
virtual void Parser_if::setSampler (
    Sampler_if * sampler ) [pure virtual]
```

Implemented in ParserDefaultImpl2.

3.93 ParserChangesInformation Class Reference

```
#include <ParserChangesInformation.h>
```

Public Member Functions

- ParserChangesInformation ()
- virtual ~ParserChangesInformation ()=default
- std::string getincludes () const
- void setIncludes (const std::string &newIncludes)
- std::string gettokens () const
- void setTokens (const std::string &newTokens)
- std::string gettypeObjs () const
- void setTypeObjs (const std::string &newTypeObjs)
- std::string getexpressions () const
- void setExpressions (const std::string &newExpressions)
- std::string getexpressionProductions () const
- void setExpressionProductions (const std::string &newExpressionProductions)
- std::string getassignments () const
- void setAssignments (const std::string &newAssignments)
- std::string getfunctionProdutions () const
- void setFunctionProdutions (const std::string &newFunctionProdutions)

3.93.1 Constructor & Destructor Documentation

3.93.1.1 ParserChangesInformation() `ParserChangesInformation::ParserChangesInformation ( )`

3.93.1.2 ~ParserChangesInformation() `virtual ParserChangesInformation::~ParserChangesInformation ( ) [virtual], [default]`

3.93.2 Member Function Documentation

3.93.2.1 getassignments() `std::string ParserChangesInformation::getassignments ( ) const`

3.93.2.2 getexpressionProductions() `std::string ParserChangesInformation::getexpressionProductions ( ) const`

3.93.2.3 getexpressions() `std::string ParserChangesInformation::getexpressions ( ) const`

3.93.2.4 getfunctionProdutions() `std::string ParserChangesInformation::getfunctionProdutions ( ) const`

3.93.2.5 getincludes() `std::string ParserChangesInformation::getincludes ( ) const`

3.93.2.6 gettokens() `std::string ParserChangesInformation::gettokens ( ) const`

3.93.2.7 gettypeObjs() `std::string ParserChangesInformation::gettypeObjs ( ) const`

3.93.2.8 setAssignments()

```
void ParserChangesInformation::setAssignments (
    const std::string & newAssignments )
```

3.93.2.9 setExpressionProductions()

```
void ParserChangesInformation::setExpressionProductions (
    const std::string & newExpressionProductions )
```

3.93.2.10 setExpressions()

```
void ParserChangesInformation::setExpressions (
    const std::string & newExpressions )
```

3.93.2.11 setFunctionProdutions()

```
void ParserChangesInformation::setFunctionProdutions (
    const std::string & newFunctionProdutions )
```

3.93.2.12 setIncludes()

```
void ParserChangesInformation::setIncludes (
    const std::string & newIncludes )
```

3.93.2.13 setTokens()

```
void ParserChangesInformation::setTokens (
    const std::string & newTokens )
```

3.93.2.14 setTypeObjs()

```
void ParserChangesInformation::setTypeObjs (
    const std::string & newTypeObjs )
```

3.94 ParserDefaultImpl1 Class Reference

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

Inheritance diagram for ParserDefaultImpl1:

Public Member Functions

- ParserDefaultImpl1 (Model ∗model)
- virtual ∼ParserDefaultImpl1 ()=default
- virtual double parse (const std::string expression) override
- virtual double parse (const std::string expression, bool ∗success, std::string ∗errorMessage) override
- virtual std::string ∗ getErrorMessage () override

3.94.1 Constructor & Destructor Documentation

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

3.94.1.2 ∼ParserDefaultImpl1() `virtual ParserDefaultImpl1::~ParserDefaultImpl1 ( ) [virtual], [default]`

3.94.2 Member Function Documentation

3.94.2.1 getErrorMessage() `std::string * ParserDefaultImpl1::getErrorMessage ( ) [override], [virtual]`

Implements Parser_if.

3.94.2.2 parse() [1/2] `double ParserDefaultImpl1::parse ( const std::string expression ) [override], [virtual]`

Implements Parser_if.

3.94.2.3 parse() [2/2] `double ParserDefaultImpl1::parse ( const std::string expression, bool * success, std::string * errorMessage ) [override], [virtual]`

Implements Parser_if.

Here is the call graph for this function:

3.95 ParserDefaultImpl2 Class Reference

```
#include <ParserDefaultImpl2.h>
```

Inheritance diagram for ParserDefaultImpl2:

Public Member Functions

- ParserDefaultImpl2 (Model ∗model, Sampler_if ∗sampler, bool throws=false)
- virtual ∼ParserDefaultImpl2 ()=default
- virtual double parse (const std::string expression) override
- virtual double parse (const std::string expression, bool ∗success, std::string ∗errorMessage) override
- virtual std::string ∗ getErrorMessage () override
- virtual void setSampler (Sampler_if ∗_sampler) override
- virtual Sampler_if ∗ getSampler () const override
- virtual genesyspp_driver getParser () const override

3.95.1 Constructor & Destructor Documentation

3.95.1.1 ParserDefaultImpl2()

```
ParserDefaultImpl2::ParserDefaultImpl2 (
    Model * model,
    Sampler_if * sampler,
    bool throws = false )
```

3.95.1.2 ~ParserDefaultImpl2()

```
virtual ParserDefaultImpl2::~ParserDefaultImpl2 ( ) [virtual], [default]
```

3.95.2 Member Function Documentation

3.95.2.1 getErrorMessage()

```
std::string * ParserDefaultImpl2::getErrorMessage ( ) [override], [virtual]
```

Implements Parser_if.

Here is the call graph for this function:

3.95.2.2 getParser()

```
genesyspp_driver ParserDefaultImpl2::getParser ( ) const [override], [virtual]
```

Implements Parser_if.

3.95.2.3 getSampler()

```
Sampler_if * ParserDefaultImpl2::getSampler ( ) const [override], [virtual]
```

Implements Parser_if.

Here is the call graph for this function:

3.95.2.4 parse() [1/2]

```
double ParserDefaultImpl2::parse (
    const std::string expression ) [override], [virtual]
```

Implements Parser_if.

Here is the call graph for this function:

3.95.2.5 parse() [2/2]

```
double ParserDefaultImpl2::parse (
    const std::string expression,
    bool * success,
    std::string * errorMessage ) [override], [virtual]
```

Implements Parser_if.

Here is the call graph for this function:

3.95.2.6 setSampler()

```
void ParserDefaultImpl2::setSampler (
    Sampler_if * _sampler ) [override], [virtual]
```

Implements Parser_if.

Here is the call graph for this function:

3.96 ParserManager Class Reference

```
#include <ParserManager.h>
```

Classes

- struct GenerateNewParserResult
- struct NewParser

Public Member Functions

- ParserManager ()
- virtual ~ParserManager ()=default
- ParserManager::GenerateNewParserResult generateNewParser (ParserChangesInformation *changes)
- bool connectNewParser (ParserManager::NewParser newParser)

3.96.1 Constructor & Destructor Documentation

3.96.1.1 ParserManager() `ParserManager::ParserManager ( )`

3.96.1.2 ~ParserManager() `virtual ParserManager::~ParserManager ( ) [virtual], [default]`

3.96.2 Member Function Documentation

3.96.2.1 connectNewParser()

```
bool ParserManager::connectNewParser (
        ParserManager::NewParser newParser )
```

3.96.2.2 generateNewParser()

```
ParserManager::GenerateNewParserResult ParserManager::generate↩
NewParser (
        ParserChangesInformation * changes )
```

3.97 PersistenceRecord Class Reference

Serialization/Deserialization structure which behaves like an associative array.

```
#include <Persistence.h>
```

Classes

- struct Entry

  "Raw" entry, users should treat this as an opaque type since payloads could be in any format.

Public Types

- using Payload = std::string
- using Iterator = std::unordered_map< std::string, Entry >::iterator

Public Member Functions

- virtual ~PersistenceRecord ()
- virtual PersistenceRecord * newInstance ()
- virtual std::string loadField (std::string key, std::string defaultValue="")
- virtual double loadField (std::string key, double defaultValue)
- virtual unsigned int loadField (std::string key, unsigned int defaultValue)
- virtual int loadField (std::string key, int defaultValue)
- virtual Util::TimeUnit loadField (std::string key, Util::TimeUnit defaultValue)
- virtual void saveField (std::string key, std::string value, const std::string defaultValue, const bool saveIf↩Default=false)
- virtual void saveField (std::string key, std::string value)
- virtual void saveField (std::string key, double value, const double defaultValue, const bool saveIfDefault=false)
- virtual void saveField (std::string key, unsigned int value, const unsigned int defaultValue, const bool saveIf↩Default=false)
- virtual void saveField (std::string key, unsigned int value)
- virtual void saveField (std::string key, int value, const int defaultValue, const bool saveIfDefault=false)
- virtual void saveField (std::string key, Util::TimeUnit value, const Util::TimeUnit defaultValue, const bool save↩IfDefault=false)
- PersistenceRecord (ModelPersistence_if &config)
- std::size_t size () const

  Returns the number of distinct entries in the record.
- void insert (Entry entry)

  Inserts a raw entry in the record.
- void insert (Iterator it, Iterator end)

  Inserts multiple entries from the given iterators.
- void erase (const std::string &key)

  Removes a specific entry from the record.
- void clear ()

  Removes all entries from this record.
- Iterator find (const std::string &key)

  Returns an iterator over a specific entry, if found.
- Iterator begin ()

  Returns an iterator over raw entries.
- Iterator end ()

  Past-the-end sentinel iterator.

3.97.1 Detailed Description

Serialization/Deserialization structure which behaves like an associative array.

Simulation models/components/definitions use this as a "visitor", which is guided across their innards during ser/deserialization and filled with fields. See `PersistentObject_base`.

When this structure reaches the main model persistence interface, it finishes the serialization process in a format-specific manner. See `ModelPersistence_if` and `ModelSerializer`.

3.97.2 Member Typedef Documentation

3.97.2.1 Iterator

```
using PersistenceRecord::Iterator = std::unordered_map<std::string, Entry>↩
::iterator
```

3.97.2.2 Payload

```
using PersistenceRecord::Payload = std::string
```

3.97.3 Constructor & Destructor Documentation

3.97.3.1 ~PersistenceRecord()

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

3.97.3.2 PersistenceRecord()

```
PersistenceRecord::PersistenceRecord (
    ModelPersistence_if & config ) [explicit]
```

3.97.4 Member Function Documentation

3.97.4.1 begin()

```
PersistenceRecord::Iterator PersistenceRecord::begin ( )
```

Returns an iterator over raw entries.

3.97.4.2 clear()

```
void PersistenceRecord::clear ( )
```

Removes all entries from this record.

3.97.4.3 end() `PersistenceRecord::Iterator PersistenceRecord::end ( )`

Past-the-end sentinel iterator.

3.97.4.4 erase() `void PersistenceRecord::erase ( const std::string & key )`

Removes a specific entry from the record.

3.97.4.5 find() `PersistenceRecord::Iterator PersistenceRecord::find ( const std::string & key )`

Returns an iterator over a specific entry, if found.

3.97.4.6 insert() [1/2] `void PersistenceRecord::insert ( PersistenceRecord::Entry entry )`

Inserts a raw entry in the record.

3.97.4.7 insert() [2/2] `void PersistenceRecord::insert ( Iterator it, Iterator end )`

Inserts multiple entries from the given iterators.

Here is the call graph for this function:

3.97.4.8 loadField() [1/5]

```
double PersistenceRecord::loadField (
    std::string key,
    double defaultValue ) [virtual]
```

3.97.4.9 loadField() [2/5]

```
int PersistenceRecord::loadField (
    std::string key,
    int defaultValue ) [virtual]
```

3.97.4.10 loadField() [3/5]

```
std::string PersistenceRecord::loadField (
    std::string key,
    std::string defaultValue = "" ) [virtual]
```

3.97.4.11 loadField() [4/5]

```
unsigned int PersistenceRecord::loadField (
    std::string key,
    unsigned int defaultValue ) [virtual]
```

3.97.4.12 loadField() [5/5]

```
Util::TimeUnit PersistenceRecord::loadField (
    std::string key,
    Util::TimeUnit defaultValue ) [virtual]
```

3.97.4.13 newInstance()

```
PersistenceRecord * PersistenceRecord::newInstance ( ) [virtual]
```

Here is the call graph for this function:

3.97.4.14 saveField() [1/7]

```
void PersistenceRecord::saveField (
    std::string key,
    double value,
    const double defaultValue,
    const bool saveIfDefault = false ) [virtual]
```

3.97.4.15 saveField() [2/7]

```
void PersistenceRecord::saveField (
    std::string key,
    int value,
    const int defaultValue,
    const bool saveIfDefault = false ) [virtual]
```

3.97.4.16 saveField() [3/7]

```
void PersistenceRecord::saveField (
    std::string key,
    std::string value ) [virtual]
```

3.97.4.17 saveField() [4/7]

```
void PersistenceRecord::saveField (
    std::string key,
    std::string value,
    const std::string defaultValue,
    const bool saveIfDefault = false ) [virtual]
```

Here is the call graph for this function:

3.97.4.18 saveField() [5/7]

```
void PersistenceRecord::saveField (
    std::string key,
    unsigned int value ) [virtual]
```

3.97.4.19 saveField() [6/7]

```
void PersistenceRecord::saveField (
    std::string key,
    unsigned int value,
    const unsigned int defaultValue,
    const bool saveIfDefault = false ) [virtual]
```

3.97.4.20 saveField() [7/7]

```
void PersistenceRecord::saveField (
    std::string key,
    Util::TimeUnit value,
    const Util::TimeUnit defaultValue,
    const bool saveIfDefault = false ) [virtual]
```

3.97.4.21 size()

```
std::size_t PersistenceRecord::size ( ) const
```

Returns the number of distinct entries in the record.

3.98 PersistentObject_base Class Reference

Dynamic serializable interface.

```
#include <Persistence.h>
```

Inheritance diagram for PersistentObject_base:

Public Member Functions

- virtual ~PersistentObject_base ()

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)=0

  Loads object fields from a deserialization structure.
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues=false)=0

  Saves object fields into a serialization structure.

3.98.1 Detailed Description

Dynamic serializable interface.

Any object that's supposed to be saved to and/or loaded from a file must implement this interface.

3.98.2 Constructor & Destructor Documentation

3.98.2.1 ~PersistentObject_base() `virtual PersistentObject_base::~PersistentObject_base ( ) [inline], [virtual]`

3.98.3 Member Function Documentation

3.98.3.1 _loadInstance() `virtual bool PersistentObject_base::_loadInstance ( PersistenceRecord * fields ) [protected], [pure virtual]`

Loads object fields from a deserialization structure.

Parameters

fields	a table-like structure which can be queried for parsed fields.

Returns

a boolean which is true iff the load operation was successful.

Implemented in Attribute, Counter, Entity, EntityType, ModelComponent, ModelDataDefinition, SinkModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, DummyElement, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, SequenceStep, Sequence, Set, SignalData, Station, Storage, and Variable.

3.98.3.2 _saveInstance() `virtual void PersistentObject_base::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues = false ) [protected], [pure virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Implemented in Attribute, Counter, Entity, EntityType, ModelComponent, ModelDataDefinition, SinkModelComponent, SourceModelComponent, StatisticsCollector, Access, Assign, Batch, Buffer, CellularAutomata, Clone, CppForG, Create, Decide, Delay, DiffEquations, Dispose, DropOff, DummyComponent, Enter, Exit, Leave, LSODE, MarkovChain, Match, OLD_ODEelement, PickStation, Process, Record, Release, Remove, Route, Search, Seize, Separate, Signal, Start, Stop, Store, Submodel, Unstore, Wait, Write, CppCompiler, DummyElement, ExtendedFSM, EntityGroup, Failure, File, Formula, Label, Queue, Resource, Schedule, SequenceStep, Sequence, Set, SignalData, Station, Storage, and Variable.

3.99 PickableStationItem Class Reference

```
#include <PickableStationItem.h>
```

Public Member Functions

- PickableStationItem (Model *model, std::string stationName, std::string queueName="", std::string resourceName="")
- PickableStationItem (Model *model, std::string stationName, std::string expression)
- PickableStationItem (Station *station, Queue *queue)
- PickableStationItem (Station *station, Resource *resource)
- PickableStationItem (Station *station, std::string expression)
- PickableStationItem (Station *station)
- virtual ~PickableStationItem ()=default
- std::string getExpression () const
- Queue * getQueue () const
- Resource * getResource () const
- Station * getStation () const
- void setExpression (std::string _expression)
- void setQueue (Queue *_queue)
- void setResource (Resource *_resource)
- void setStation (Station *_station)

3.99.1 Constructor & Destructor Documentation

3.99.1.1 PickableStationItem() [1/6]

```
PickableStationItem::PickableStationItem (
    Model * model,
    std::string stationName,
    std::string queueName = "",
    std::string resourceName = "" )
```

Here is the call graph for this function:

3.99.1.2 PickableStationItem() [2/6]

```
PickableStationItem::PickableStationItem (
    Model * model,
    std::string stationName,
    std::string expression )
```

Here is the call graph for this function:

3.99.1.3 PickableStationItem() [3/6]

```
PickableStationItem::PickableStationItem (
    Station * station,
    Queue * queue )
```

3.99.1.4 PickableStationItem() [4/6]

```
PickableStationItem::PickableStationItem (
    Station * station,
    Resource * resource )
```

3.99.1.5 PickableStationItem() [5/6]

```
PickableStationItem::PickableStationItem (
    Station * station,
    std::string expression )
```

3.99.1.6 PickableStationItem() [6/6]

```
PickableStationItem::PickableStationItem (
    Station * station )
```

3.99.1.7 ~PickableStationItem()

`virtual PickableStationItem::~PickableStationItem ( ) [virtual], [default]`

3.99.2 Member Function Documentation

3.99.2.1 getExpression() `std::string PickableStationItem::getExpression ( ) const`

3.99.2.2 getQueue() `Queue * PickableStationItem::getQueue ( ) const`

3.99.2.3 getResource() `Resource * PickableStationItem::getResource ( ) const`

3.99.2.4 getStation() `Station * PickableStationItem::getStation ( ) const`

3.99.2.5 setExpression() `void PickableStationItem::setExpression ( std::string _expression )`

3.99.2.6 setQueue() `void PickableStationItem::setQueue ( Queue * _queue )`

3.99.2.7 setResource() `void PickableStationItem::setResource ( Resource * _resource )`

3.99.2.8 setStation() `void PickableStationItem::setStation ( Station * _station )`

3.100 PickStation Class Reference

```
#include <PickStation.h>
```

Inheritance diagram for PickStation:

Public Types

- enum class TestCondition : int { MINIMUM = 1 , MAXIMUM = 2 }

Public Member Functions

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

  virtual public methods
- void setTestCondition (PickStation::TestCondition _testCondition)
- PickStation::TestCondition getTestCondition () const
- void setSaveAttribute (std::string _saveAttribute)
- std::string getSaveAttribute () const
- List< PickableStationItem * > * getPickableStationItens () const
- void setPickConditionExpression (bool _pickConditionExpression)
- bool isPickConditionExpression () const
- void setPickConditionNumberInQueue (bool _pickConditionNumberInQueue)
- bool isPickConditionNumberInQueue () const
- void setPickConditionNumberBusyResource (bool _pickConditionNumberBusyResource)
- bool isPickConditionNumberBusyResource () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation * _getParserChangesInformation ()
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- virtual void _addProperty (PropertyBase *property)

Additional Inherited Members

3.100.1 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.

3.100.2 Member Enumeration Documentation

3.100.2.1 TestCondition `enum class PickStation::TestCondition : int [strong]`

Enumerator

MINIMUM	
MAXIMUM	

3.100.3 Constructor & Destructor Documentation

3.100.3.1 PickStation()

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

3.100.3.2 ~PickStation() `virtual PickStation::~PickStation ( ) [virtual], [default]`

3.100.4 Member Function Documentation

3.100.4.1 _addProperty()

```
void PickStation::_addProperty (
    PropertyBase * property ) [protected], [virtual]
```

Reimplemented from ModelDataDefinition.

3.100.4.2 _check() `bool PickStation::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.100.4.3 _createInternalAndAttachedData() `void PickStation::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.100.4.4 _getParserChangesInformation() `ParserChangesInformation * PickStation::_getParserChangesInformation ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.100.4.5 _initBetweenReplications() `void PickStation::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.100.4.6 _loadInstance()

```
bool PickStation::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.100.4.7 _onDispatchEvent()

```
void PickStation::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.100.4.8 _saveInstance()

```
void PickStation::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.100.4.9 getPickableStationItens() `List< PickableStationItem * > * PickStation::getPickable↩ StationItens ( ) const`

3.100.4.10 GetPluginInformation() `PluginInformation * PickStation::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.100.4.11 getSaveAttribute() `std::string PickStation::getSaveAttribute ( ) const`

3.100.4.12 getTestCondition() `PickStation::TestCondition PickStation::getTestCondition ( ) const`

3.100.4.13 isPickConditionExpression()

`bool PickStation::isPickConditionExpression ( ) const`

3.100.4.14 isPickConditionNumberBusyResource()

`bool PickStation::isPickConditionNumberBusy↩ Resource ( ) const`

3.100.4.15 isPickConditionNumberInQueue()

`bool PickStation::isPickConditionNumberInQueue ( ) const`

3.100.4.16 LoadInstance()

```
ModelComponent * PickStation::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.100.4.17 NewInstance()

```
ModelDataDefinition * PickStation::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.100.4.18 setPickConditionExpression()

```
void PickStation::setPickConditionExpression (
    bool _pickConditionExpression )
```

3.100.4.19 setPickConditionNumberBusyResource()

```
void PickStation::setPickConditionNumber↩
BusyResource (
            bool _pickConditionNumberBusyResource )
```

3.100.4.20 setPickConditionNumberInQueue()

```
void PickStation::setPickConditionNumberInQueue (
            bool _pickConditionNumberInQueue )
```

3.100.4.21 setSaveAttribute()

```
void PickStation::setSaveAttribute (
            std::string _saveAttribute )
```

3.100.4.22 setTestCondition()

```
void PickStation::setTestCondition (
            PickStation::TestCondition _testCondition )
```

3.100.4.23 show()

```
std::string PickStation::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.101 PickUp Class Reference

```
#include <PickUp.h>
```

Inheritance diagram for PickUp:

Public Member Functions

- PickUp (Model ∗model, std::string name="")
- virtual ∼PickUp ()=default

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Additional Inherited Members

3.101.1 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.

3.101.2 Constructor & Destructor Documentation

3.101.2.1 PickUp()

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

3.101.2.2 ~PickUp()

```
virtual PickUp::~PickUp ( ) [virtual], [default]
```

3.101.3 Member Function Documentation

3.101.3.1 GetPluginInformation()

```
PluginInformation * PickUp::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.101.3.2 NewInstance()

```
ModelDataDefinition * PickUp::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.102 Plugin Class Reference

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

Public Member Functions

- Plugin (std::string filename_so_dll)
- Plugin (StaticGetPluginInformation getInformation)
- virtual ~Plugin ()=default
- std::string show ()
- bool isIsValidPlugin () const
- PluginInformation * getPluginInfo () const
- ModelDataDefinition * loadNew (Model *model, PersistenceRecord *fields)

  creates a new ModelDataDefinition from fields loaded from a file
- bool loadAndInsertNew (Model *model, PersistenceRecord *fields)
- ModelDataDefinition * newInstance (Model *model, std::string name="")

3.102.1 Detailed Description

A Plugin represents a dynamically linked component class (ModelComponent) or modeldatum class (ModelDataDefinition); 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

3.102.2 Constructor & Destructor Documentation

3.102.2.1 Plugin() [1/2]

```
Plugin::Plugin (
    std::string filename_so_dll )
```

3.102.2.2 Plugin() [2/2]

```
Plugin::Plugin (
    StaticGetPluginInformation getInformation )
```

3.102.2.3 ~Plugin()

```
virtual Plugin::~Plugin ( ) [virtual], [default]
```

3.102.3 Member Function Documentation

3.102.3.1 getPluginInfo()

```
PluginInformation * Plugin::getPluginInfo ( ) const
```

3.102.3.2 isIsValidPlugin()

```
bool Plugin::isIsValidPlugin ( ) const
```

3.102.3.3 loadAndInsertNew()

```
bool Plugin::loadAndInsertNew (
    Model * model,
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.102.3.4 loadNew()

```
ModelDataDefinition * Plugin::loadNew (
    Model * model,
    PersistenceRecord * fields )
```

creates a new ModelDataDefinition from fields loaded from a file

Here is the call graph for this function:

3.102.3.5 newInstance()

```
ModelDataDefinition * Plugin::newInstance (
    Model * model,
    std::string name = "" )
```

Here is the call graph for this function:

3.102.3.6 show()

```
std::string Plugin::show ( )
```

Here is the call graph for this function:

3.103 PluginConnector_if Class Reference

```
#include <PluginConnector_if.h>
```

Inheritance 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

3.103.1 Member Function Documentation

3.103.1.1 check()

```
virtual Plugin * PluginConnector_if::check (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in PluginConnectorDummyImpl1.

3.103.1.2 connect()

```
virtual Plugin * PluginConnector_if::connect (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in PluginConnectorDummyImpl1.

3.103.1.3 disconnect() [1/2]

```
virtual bool PluginConnector_if::disconnect (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in PluginConnectorDummyImpl1.

3.103.1.4 disconnect() [2/2]

```
virtual bool PluginConnector_if::disconnect (
    Plugin * plugin ) [pure virtual]
```

Implemented in PluginConnectorDummyImpl1.

3.104 PluginConnectorDummyImpl1 Class Reference

```
#include <PluginConnectorDummyImpl1.h>
```

Inheritance 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)

3.104.1 Constructor & Destructor Documentation

3.104.1.1 PluginConnectorDummyImpl1() `PluginConnectorDummyImpl1::PluginConnectorDummyImpl1 ( )`

3.104.1.2 ~PluginConnectorDummyImpl1() `virtual PluginConnectorDummyImpl1::~PluginConnector↩ DummyImpl1 ( ) [virtual], [default]`

3.104.2 Member Function Documentation

3.104.2.1 check() `Plugin * PluginConnectorDummyImpl1::check ( const std::string dynamicLibraryFilename ) [virtual]`

@ @TODO:To implement

Implements PluginConnector_if.

3.104.2.2 connect() `Plugin * PluginConnectorDummyImpl1::connect ( const std::string dynamicLibraryFilename ) [virtual]`

Implements PluginConnector_if.

3.104.2.3 disconnect() [1/2] `bool PluginConnectorDummyImpl1::disconnect ( const std::string dynamicLibraryFilename ) [virtual]`

Implements PluginConnector_if.

3.104.2.4 disconnect() [2/2] bool PluginConnectorDummyImpl1::disconnect (Plugin * plugin) [virtual]

Implements PluginConnector_if.

3.105 PluginInformation Class Reference

```
#include <PluginInformation.h>
```

Public Member Functions

- PluginInformation (std::string pluginTypename, StaticLoaderComponentInstance componentloader, Static↩ ConstructorDataDefinitionInstance elementConstructor)
- PluginInformation (std::string pluginTypename, StaticLoaderDataDefinitionInstance elementloader, Static↩ ConstructorDataDefinitionInstance elementConstructor)
- StaticLoaderDataDefinitionInstance getDataDefinitionLoader () const
- StaticLoaderComponentInstance GetComponentLoader () const
- StaticConstructorDataDefinitionInstance getDataDefinitionConstructor () 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
- void setDescriptionHelp (std::string _descriptionHelp)
- std::string getDescriptionHelp () const
- void setFields (std::map< std::string, std::string > *_fiewlds)
- std::map< std::string, std::string > * getFields () const
- void setLanguageTemplate (std::string _languageTemplate)
- std::string getLanguageTemplate () const
- void setCategory (std::string _category)
- std::string getCategory () const

3.105.1 Constructor & Destructor Documentation

3.105.1.1 PluginInformation() [1/2]

```
PluginInformation::PluginInformation (
    std::string pluginTypename,
    StaticLoaderComponentInstance componentloader,
    StaticConstructorDataDefinitionInstance elementConstructor )
```

3.105.1.2 PluginInformation() [2/2]

```
PluginInformation::PluginInformation (
    std::string pluginTypename,
    StaticLoaderDataDefinitionInstance elementloader,
    StaticConstructorDataDefinitionInstance elementConstructor )
```

3.105.2 Member Function Documentation

3.105.2.1 getAuthor()

```
std::string PluginInformation::getAuthor ( ) const
```

3.105.2.2 getCategory()

```
std::string PluginInformation::getCategory ( ) const
```

3.105.2.3 GetComponentLoader()

```
StaticLoaderComponentInstance PluginInformation::GetComponent↩
Loader ( ) const
```

3.105.2.4 getDataDefinitionConstructor()

```
StaticConstructorDataDefinitionInstance PluginInformation↩
::getDataDefinitionConstructor ( ) const
```

3.105.2.5 getDataDefinitionLoader()

```
StaticLoaderDataDefinitionInstance PluginInformation::get↩
DataDefinitionLoader ( ) const
```

3.105.2.6 getDate()

```
std::string PluginInformation::getDate ( ) const
```

3.105.2.7 getDescriptionHelp()

```
std::string PluginInformation::getDescriptionHelp ( ) const
```

3.105.2.8 getDynamicLibFilenameDependencies()

```
std::list< std::string > * PluginInformation↩
::getDynamicLibFilenameDependencies ( ) const
```

3.105.2.9 getFields()

```
std::map< std::string, std::string > * PluginInformation::getFields ( )
const
```

3.105.2.10 getLanguageTemplate()

```
std::string PluginInformation::getLanguageTemplate ( ) const
```

3.105.2.11 getMaximumInputs()

```
unsigned short PluginInformation::getMaximumInputs ( ) const
```

3.105.2.12 getMaximumOutputs()

```
unsigned short PluginInformation::getMaximumOutputs ( ) const
```

3.105.2.13 getMinimumInputs()

```
unsigned short PluginInformation::getMinimumInputs ( ) const
```

3.105.2.14 getMinimumOutputs()

```
unsigned short PluginInformation::getMinimumOutputs ( ) const
```

3.105.2.15 getObservation()

```
std::string PluginInformation::getObservation ( ) const
```

3.105.2.16 getPluginTypename()

```
std::string PluginInformation::getPluginTypename ( ) const
```

3.105.2.17 getVersion()

```
std::string PluginInformation::getVersion ( ) const
```

3.105.2.18 insertDynamicLibFileDependence()

```
void PluginInformation::insertDynamicLibFile↩
Dependence (
    std::string filename )
```

3.105.2.19 isComponent()

```
bool PluginInformation::isComponent ( ) const
```

3.105.2.20 isGenerateReport()

```
bool PluginInformation::isGenerateReport ( ) const
```

3.105.2.21 isReceiveTransfer()

```
bool PluginInformation::isReceiveTransfer ( ) const
```

3.105.2.22 isSendTransfer()

```
bool PluginInformation::isSendTransfer ( ) const
```

3.105.2.23 isSink()

```
bool PluginInformation::isSink ( ) const
```

3.105.2.24 isSource()

```
bool PluginInformation::isSource ( ) const
```

3.105.2.25 setAuthor()

```
void PluginInformation::setAuthor (
    std::string author )
```

3.105.2.26 setCategory()

```
void PluginInformation::setCategory (
    std::string _category )
```

3.105.2.27 setDate()

```
void PluginInformation::setDate (
    std::string date )
```

3.105.2.28 setDescriptionHelp()

```
void PluginInformation::setDescriptionHelp (
    std::string _descriptionHelp )
```

3.105.2.29 setDynamicLibFilenameDependencies()

```
void PluginInformation::setDynamicLibFilenameDependencies (
    std::list< std::string > * dynamicLibFilenameDependencies )
```

3.105.2.30 setFields()

```
void PluginInformation::setFields (
    std::map< std::string, std::string > * _fiewlds )
```

3.105.2.31 setGenerateReport()

```
void PluginInformation::setGenerateReport (
    bool generateReport )
```

3.105.2.32 setLanguageTemplate()

```
void PluginInformation::setLanguageTemplate (
    std::string _languageTemplate )
```

3.105.2.33 setMaximumInputs()

```
void PluginInformation::setMaximumInputs (
    unsigned short _maximumInputs )
```

3.105.2.34 setMaximumOutputs()

```
void PluginInformation::setMaximumOutputs (
    unsigned short _maximumOutputs )
```

3.105.2.35 setMinimumInputs()

```
void PluginInformation::setMinimumInputs (
    unsigned short _minimumInputs )
```

3.105.2.36 setMinimumOutputs()

```
void PluginInformation::setMinimumOutputs (
    unsigned short _minimumOutputs )
```

3.105.2.37 setObservation()

```
void PluginInformation::setObservation (
    std::string observation )
```

3.105.2.38 setReceiveTransfer()

```
void PluginInformation::setReceiveTransfer (
    bool receiveTransfer )
```

3.105.2.39 setSendTransfer()

```
void PluginInformation::setSendTransfer (
    bool sendTransfer )
```

3.105.2.40 setSink()

```
void PluginInformation::setSink (
    bool Sink )
```

3.105.2.41 setSource()

```
void PluginInformation::setSource (
    bool Source )
```

3.105.2.42 setVersion()

```
void PluginInformation::setVersion (
    std::string version )
```

3.106 PluginManager Class Reference

```
#include <PluginManager.h>
```

Public Member Functions

- PluginManager (Simulator *simulator)
- virtual ~PluginManager ()=default
- std::string show ()
- bool completePluginsFieldsAndTemplates ()
- 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)
- bool autoInsertPlugins (const std::string pluginsListFilename)
- Plugin * front ()
- Plugin * next ()
- Plugin * last ()
- unsigned int size ()
- Plugin * getAtRank (unsigned int rank)
- ModelDataDefinition * newInstance (std::string pluginTypename, Model *model, std::string name="")
- template<typename T >
  T * newInstance (Model *model, std::string name="")

3.106.1 Constructor & Destructor Documentation

3.106.1.1 PluginManager() `PluginManager::PluginManager ( Simulator * simulator )`

3.106.1.2 ~PluginManager() `virtual PluginManager::~PluginManager ( ) [virtual], [default]`

3.106.2 Member Function Documentation

3.106.2.1 autoInsertPlugins() `bool PluginManager::autoInsertPlugins ( const std::string pluginsListFilename )`

Here is the call graph for this function:

3.106.2.2 check() `bool PluginManager::check ( const std::string dynamicLibraryFilename )`

Here is the call graph for this function:

3.106.2.3 completePluginsFieldsAndTemplates()

```
bool PluginManager::completePluginsFieldsAndTemplates ( )
```

3.106.2.4 find()

```
Plugin * PluginManager::find (
    std::string pluginTypeName )
```

3.106.2.5 front()

```
Plugin * PluginManager::front ( )
```

3.106.2.6 getAtRank()

```
Plugin * PluginManager::getAtRank (
    unsigned int rank )
```

3.106.2.7 insert()

```
Plugin * PluginManager::insert (
    const std::string dynamicLibraryFilename )
```

Here is the call graph for this function:

3.106.2.8 last()

```
Plugin * PluginManager::last ( )
```

3.106.2.9 newInstance() [1/2] template<typename T >

```
T * PluginManager::newInstance (
            Model * model,
            std::string name = "" )  [inline]
```

innvalid use of incomplete class _simulator->getTracer()->traceError(TraceManager::Level::L1_errorFatal, "Error: Could not find any plugin with Typename \"" + pluginTypename + "\"");Here is the call graph for this function:

3.106.2.10 newInstance() [2/2] ModelDataDefinition * PluginManager::newInstance (

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

Here is the call graph for this function:

3.106.2.11 next() Plugin * PluginManager::next ( )

3.106.2.12 remove() [1/2] bool PluginManager::remove (const std::string *dynamicLibraryFilename*)

Here is the call graph for this function:

3.106.2.13 remove() [2/2] bool PluginManager::remove (Plugin * *plugin*)

Here is the call graph for this function:

3.106.2.14 show() std::string PluginManager::show ( )

3.106.2.15 size() unsigned int PluginManager::size ( )

3.107 ProbabilityDistribution Class Reference

```
#include <ProbabilityDistribution.h>
```

Inheritance diagram for ProbabilityDistribution:

Static Public Member Functions

- static double inverseChi2 (double cumulativeProbability, double degreeFreedom)
- static double inverseFFisherSnedecor (double cumulativeProbability, double d1, double d2)
- static double inverseNormal (double cumulativeProbability, double mean, double stddev)
- static double inverseTStudent (double cumulativeProbability, double mean, double stddev, double degree↩ Freedom)

Additional Inherited Members

3.107.1 Member Function Documentation

3.107.1.1 inverseChi2()

```
double ProbabilityDistribution::inverseChi2 (
    double cumulativeProbability,
    double degreeFreedom ) [static]
```

@TODO: Is there a way to better determine the upper limit?Here is the call graph for this function:

3.107.1.2 inverseFFisherSnedecor()

```
double ProbabilityDistribution::inverseFFisherSnedecor (
    double cumulativeProbability,
    double d1,
    double d2 ) [static]
```

@TODO: Is there a way to better determine the upper limit?Here is the call graph for this function:

3.107.1.3 inverseNormal()

```
double ProbabilityDistribution::inverseNormal (
    double cumulativeProbability,
    double mean,
    double stddev ) [static]
```

@TODO: Could be better (separate into 1,2,3xstddev)

3.107.1.4 inverseTStudent()

```
double ProbabilityDistribution::inverseTStudent (
    double cumulativeProbability,
    double mean,
    double stddev,
    double degreeFreedom ) [static]
```

@TODO: Could be better

3.108 ProbabilityDistributionBase Class Reference

```
#include <ProbabilityDistributionBase.h>
```

Inheritance diagram for ProbabilityDistributionBase:

Static Public Member Functions

- static double beta (double x, double alpha, double beta)
- static double chi2 (double x, double degreeFreedom)
- static double erlang (double x, double shape, double scale)
- static double exponential (double x, double mean)
- static double fisherSnedecor (double x, double d1, double d2)
- static double gamma (double x, double shape, double scale)
- static double logNormal (double x, double mean, double stddev)
- static double normal (double x, double mean, double stddev)
- static double poisson (double x, double mean)
- static double triangular (double x, double min, double mode, double max)
- static double tStudent (double x, double mean, double stddev, double degreeFreedom)
- static double uniform (double x, double min, double max)
- static double weibull (double x, double shape, double scale)

Static Protected Member Functions

- static double _gammaFunction (double x)
- static double _betaFunction (double x, double y)
- static double _factorial (double x)

3.108.1 Member Function Documentation

3.108.1.1 _betaFunction()

```
double ProbabilityDistributionBase::_betaFunction (
    double x,
    double y ) [static], [protected]
```

Here is the call graph for this function:

3.108.1.2 _factorial()

```
double ProbabilityDistributionBase::_factorial (
    double x )  [static], [protected]
```

Here is the call graph for this function:

3.108.1.3 _gammaFunction()

```
double ProbabilityDistributionBase::_gammaFunction (
    double x )  [static], [protected]
```

@TODO: Implement by myself

3.108.1.4 beta()

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

Here is the call graph for this function:

3.108.1.5 chi2()

```
double ProbabilityDistributionBase::chi2 (
    double x,
    double degreeFreedom )  [static]
```

Here is the call graph for this function:

3.108.1.6 erlang()

```
double ProbabilityDistributionBase::erlang (
    double x,
    double shape,
    double scale ) [static]
```

3.108.1.7 exponential()

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

3.108.1.8 fisherSnedecor()

```
double ProbabilityDistributionBase::fisherSnedecor (
    double x,
    double d1,
    double d2 ) [static]
```

Here is the call graph for this function:

3.108.1.9 gamma()

```
double ProbabilityDistributionBase::gamma (
    double x,
    double shape,
    double scale ) [static]
```

Here is the call graph for this function:

3.108.1.10 logNormal()

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

3.108.1.11 normal()

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

3.108.1.12 poisson()

```
double ProbabilityDistributionBase::poisson (
    double x,
    double mean ) [static]
```

Here is the call graph for this function:

3.108.1.13 triangular()

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

3.108.1.14 tStudent()

```
double ProbabilityDistributionBase::tStudent (
    double x,
    double mean,
    double stddev,
    double degreeFreedom ) [static]
```

Here is the call graph for this function:

3.108.1.15 uniform()

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

3.108.1.16 weibull()

```
double ProbabilityDistributionBase::weibull (
    double x,
    double shape,
    double scale ) [static]
```

3.109 Process Class Reference

```
#include <Process.h>
```

Inheritance diagram for Process:

Public Member Functions

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

  virtual public methods
- void setPriority (unsigned short _priority)

- unsigned short getPriority () const
- void setPriorityExpression (std::string priority)
- std::string getPriorityExpression () const
- void setAllocationType (Util::AllocationType _allocationType)
- Util::AllocationType getAllocationType () const
- List< SeizableItem * > * getSeizeRequests () const
- void setQueueableItem (QueueableItem *_queueableItem)
- QueueableItem * getQueueableItem () const
- void setDelayExpression (std::string _delayExpression)
- void setDelayExpression (std::string _delayExpression, Util::TimeUnit _delayTimeUnit)
- std::string delayExpression () const
- void setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)
- Util::TimeUnit delayTimeUnit () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.109.1 Detailed Description

This component ...

3.109.2 Constructor & Destructor Documentation

3.109.2.1 Process()

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

Here is the call graph for this function:

3.109.2.2 ~Process()

```
virtual Process::~Process ( ) [virtual], [default]
```

3.109.3 Member Function Documentation

3.109.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.109.3.2 _createInternalAndAttachedData() `void Process::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.109.3.3 _loadInstance() `bool Process::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.109.3.4 _onDispatchEvent()

```
void Process::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.109.3.5 _saveInstance()

```
void Process::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.109.3.6 delayExpression() `std::string Process::delayExpression ( ) const`

Here is the call graph for this function:

3.109.3.7 delayTimeUnit() `Util::TimeUnit Process::delayTimeUnit ( ) const`

Here is the call graph for this function:

3.109.3.8 getAllocationType() `Util::AllocationType Process::getAllocationType ( ) const`

Here is the call graph for this function:

3.109.3.9 GetPluginInformation() `PluginInformation * Process::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.109.3.10 getPriority() `unsigned short Process::getPriority ( ) const`

Here is the call graph for this function:

3.109.3.11 getPriorityExpression() `std::string Process::getPriorityExpression ( ) const`

Here is the call graph for this function:

3.109.3.12 getQueueableItem() `QueueableItem * Process::getQueueableItem ( ) const`

Here is the call graph for this function:

3.109.3.13 getSeizeRequests() `List< SeizableItem * > * Process::getSeizeRequests ( ) const`

Here is the call graph for this function:

3.109.3.14 LoadInstance() `ModelComponent * Process::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.109.3.15 NewInstance()

```
ModelDataDefinition * Process::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.109.3.16 setAllocationType()

```
void Process::setAllocationType (
    Util::AllocationType _allocationType )
```

Here is the call graph for this function:

3.109.3.17 setDelayExpression() [1/2]

```
void Process::setDelayExpression (
    std::string _delayExpression )
```

Here is the call graph for this function:

3.109.3.18 setDelayExpression() [2/2]

```
void Process::setDelayExpression (
    std::string _delayExpression,
    Util::TimeUnit _delayTimeUnit )
```

Here is the call graph for this function:

3.109.3.19 setDelayTimeUnit()

```
void Process::setDelayTimeUnit (
    Util::TimeUnit _delayTimeUnit )
```

Here is the call graph for this function:

3.109.3.20 setPriority()

```
void Process::setPriority (
    unsigned short _priority )
```

Here is the call graph for this function:

3.109.3.21 setPriorityExpression()

```
void Process::setPriorityExpression (
    std::string priority )
```

Here is the call graph for this function:

3.109.3.22 setQueueableItem()

```
void Process::setQueueableItem (
    QueueableItem * _queueableItem )
```

Here is the call graph for this function:

3.109.3.23 show()

```
std::string Process::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.110 PropertyGenesys Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for PropertyGenesys:

Public Member Functions

- PropertyGenesys (std::string className, std::string elementName, std::string propertyName, std::string whatsThis="", std::string propertyId="", std::string toolTip="", std::string statusTip="")

- virtual ~PropertyGenesys ()=default
- std::list< PropertyGenesys * > subProperties () const
- std::string toolTip () const
- std::string statusTip () const
- std::string propertyName () const
- std::string propertyId () const
- bool isEnabled () const
- bool isModified () const
- bool isSubProperty () const
- bool hasValue () const
- std::string valueText () const
- void setToolTip (const std::string text)
- void setStatusTip (const std::string text)
- void setWhatsThis (const std::string text)
- void setPropertyName (const std::string text)
- void setPropertyId (const std::string text)
- void setEnabled (bool enable)
- void setModified (bool modified)
- virtual bool compare (PropertyGenesys *otherProperty) const
- void addSubProperty (PropertyGenesys *property)
- void insertSubProperty (PropertyGenesys *property, PropertyGenesys *afterProperty)
- void removeSubProperty (PropertyGenesys *property)

Protected Member Functions

- void propertyChanged ()

Protected Attributes

- std::string _className
- std::string _propertyName
- std::string _whatsThis
- std::string _propertyId
- std::string _toolTip
- std::string _statusTip
- std::list< PropertyGenesys * > _subProperties
- bool _isEnabled
- bool _isModified
- bool _hasValue
- bool _isSubProperty

3.110.1 Constructor & Destructor Documentation

3.110.1.1 PropertyGenesys()

```
PropertyGenesys::PropertyGenesys (
    std::string  className,
    std::string  elementName,
    std::string  propertyName,
    std::string  whatsThis = "",
    std::string  propertyId = "",
    std::string  toolTip = "",
    std::string  statusTip = "" )  [inline]
```

Here is the call graph for this function:

3.110.1.2 ~PropertyGenesys()

```
virtual PropertyGenesys::~PropertyGenesys ( )  [virtual], [default]
```

3.110.2 Member Function Documentation

3.110.2.1 addSubProperty()

```
void PropertyGenesys::addSubProperty (
    PropertyGenesys * property )  [inline]
```

3.110.2.2 compare()

```
virtual bool PropertyGenesys::compare (
    PropertyGenesys * otherProperty ) const  [inline], [virtual]
```

3.110.2.3 hasValue()

```
bool PropertyGenesys::hasValue ( ) const  [inline]
```

3.110.2.4 insertSubProperty()

```
void PropertyGenesys::insertSubProperty (
    PropertyGenesys * property,
    PropertyGenesys * afterProperty )  [inline]
```

3.110.2.5 isEnabled() `bool PropertyGenesys::isEnabled ( ) const [inline]`

3.110.2.6 isModified() `bool PropertyGenesys::isModified ( ) const [inline]`

3.110.2.7 isSubProperty() `bool PropertyGenesys::isSubProperty ( ) const [inline]`

3.110.2.8 propertyChanged() `void PropertyGenesys::propertyChanged ( ) [inline], [protected]`

3.110.2.9 propertyId() `std::string PropertyGenesys::propertyId ( ) const [inline]`

3.110.2.10 propertyName() `std::string PropertyGenesys::propertyName ( ) const [inline]`

3.110.2.11 removeSubProperty()

```
void PropertyGenesys::removeSubProperty (
    PropertyGenesys * property ) [inline]
```

3.110.2.12 setEnabled()

```
void PropertyGenesys::setEnabled (
    bool enable )
```

3.110.2.13 setModified()

```
void PropertyGenesys::setModified (
    bool modified )
```

3.110.2.14 setPropertyId()

```
void PropertyGenesys::setPropertyId (
    const std::string text )
```

3.110.2.15 setPropertyName()

```
void PropertyGenesys::setPropertyName (
    const std::string text )
```

3.110.2.16 setStatusTip()

```
void PropertyGenesys::setStatusTip (
    const std::string text )
```

3.110.2.17 setToolTip()

```
void PropertyGenesys::setToolTip (
    const std::string text )
```

3.110.2.18 setWhatsThis()

```
void PropertyGenesys::setWhatsThis (
    const std::string text )
```

3.110.2.19 statusTip()

```
std::string PropertyGenesys::statusTip ( ) const [inline]
```

3.110.2.20 subProperties()

```
std::list< PropertyGenesys * > PropertyGenesys::subProperties ( )
const [inline]
```

3.110.2.21 toolTip()

```
std::string PropertyGenesys::toolTip ( ) const [inline]
```

3.110.2.22 valueText()

```
std::string PropertyGenesys::valueText ( ) const [inline]
```

3.110.3 Member Data Documentation

3.110.3.1 _className

```
std::string PropertyGenesys::_className [protected]
```

3.110.3.2 _hasValue

```
bool PropertyGenesys::_hasValue [protected]
```

3.110.3.3 _isEnabled `bool PropertyGenesys::_isEnabled [protected]`

3.110.3.4 _isModified `bool PropertyGenesys::_isModified [protected]`

3.110.3.5 _isSubProperty `bool PropertyGenesys::_isSubProperty [protected]`

3.110.3.6 _propertyId `std::string PropertyGenesys::_propertyId [protected]`

3.110.3.7 _propertyName `std::string PropertyGenesys::_propertyName [protected]`

3.110.3.8 _statusTip `std::string PropertyGenesys::_statusTip [protected]`

3.110.3.9 _subProperties `std::list<PropertyGenesys*> PropertyGenesys::_subProperties [protected]`

3.110.3.10 _toolTip `std::string PropertyGenesys::_toolTip [protected]`

3.110.3.11 _whatsThis `std::string PropertyGenesys::_whatsThis [protected]`

3.111 PropertyGenesysBase Class Reference

`#include <SimulationControlAndResponse.h>`

Inheritance diagram for PropertyGenesysBase:

Public Member Functions

- PropertyGenesysBase (std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- virtual ~PropertyGenesysBase ()=default
- std::string whatsThis () const
- std::string getClassname () const
- std::string getElementName () const
- std::string getName () const
- std::string propertyType () const
- void setWhatsThis (const std::string text)
- void setName (std::string name)

Protected Attributes

- std::string _elementName
- std::string _className
- std::string _propertyName
- std::string _propertyType
- std::string _whatsThis

3.111.1 Constructor & Destructor Documentation

3.111.1.1 PropertyGenesysBase()

```
PropertyGenesysBase::PropertyGenesysBase (
    std::string  className,
    std::string  elementName,
    std::string  propertyName,
    std::string  whatsThis = "" )  [inline]
```

Here is the call graph for this function:

3.111.1.2 ~PropertyGenesysBase()

```
virtual PropertyGenesysBase::~PropertyGenesysBase ( )  [virtual], [default]
```

3.111.2 Member Function Documentation

3.111.2.1 getClassname() `std::string PropertyGenesysBase::getClassname ( ) const [inline]`

3.111.2.2 getElementName() `std::string PropertyGenesysBase::getElementName ( ) const [inline]`

3.111.2.3 getName() `std::string PropertyGenesysBase::getName ( ) const [inline]`

3.111.2.4 propertyType() `std::string PropertyGenesysBase::propertyType ( ) const [inline]`

3.111.2.5 setName() `void PropertyGenesysBase::setName ( std::string name ) [inline]`

3.111.2.6 setWhatsThis() `void PropertyGenesysBase::setWhatsThis ( const std::string text )`

3.111.2.7 whatsThis() `std::string PropertyGenesysBase::whatsThis ( ) const [inline]`

3.111.3 Member Data Documentation

3.111.3.1 _className `std::string PropertyGenesysBase::_className [protected]`

3.111.3.2 _elementName `std::string PropertyGenesysBase::_elementName [protected]`

3.111.3.3 _propertyName `std::string PropertyGenesysBase::_propertyName [protected]`

3.111.3.4 _propertyType `std::string PropertyGenesysBase::_propertyType [protected]`

3.111.3.5 _whatsThis `std::string PropertyGenesysBase::_whatsThis [protected]`

3.112 Queue Class Reference

`#include <Queue.h>`

Inheritance diagram for Queue:

Public Types

- enum class OrderRule : int { FIFO = 1 , LIFO = 2 , HIGHESTVALUE = 3 , SMALLESTVALUE = 4 }

Public Member Functions

- Queue (Model ∗model, std::string name="")
- virtual ∼Queue ()
- virtual std::string show ()
- void insertElement (Waiting ∗modeldatum)
- void removeElement (Waiting ∗modeldatum)
- 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
- void setOrderRuleInt (int orderRule)
- int getOrderRuleInt () const
- double sumAttributesFromWaiting (Util::identification attributeID)
- double getAttributeFromWaitingRank (unsigned int rank, Util::identification attributeID)

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- virtual ParserChangesInformation ∗ _getParserChangesInformation ()

Additional Inherited Members

3.112.1 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.

3.112.2 Member Enumeration Documentation

3.112.2.1 OrderRule `enum class Queue::OrderRule : int [strong]`

Enumerator

FIFO	
LIFO	
HIGHESTVALUE	
SMALLESTVALUE	

3.112.3 Constructor & Destructor Documentation

3.112.3.1 Queue()

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

Here is the call graph for this function:

3.112.3.2 ~Queue() `Queue::~Queue ( ) [virtual]`

3.112.4 Member Function Documentation

3.112.4.1 _check() `bool Queue::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.112.4.2 _createInternalAndAttachedData() `void Queue::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.112.4.3 _getParserChangesInformation() `ParserChangesInformation * Queue::_getParserChanges↩ Information ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.112.4.4 _initBetweenReplications() `void Queue::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.112.4.5 _loadInstance() `bool Queue::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.112.4.6 _saveInstance() `void Queue::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.112.4.7 first() `Waiting * Queue::first ( )`

3.112.4.8 getAtRank() `Waiting * Queue::getAtRank ( unsigned int rank )`

3.112.4.9 getAttributeFromWaitingRank() `double Queue::getAttributeFromWaitingRank ( unsigned int rank, Util::identification attributeID )`

Here is the call graph for this function:

3.112.4.10 getAttributeName() `std::string Queue::getAttributeName ( ) const`

3.112.4.11 getOrderRule() `Queue::OrderRule Queue::getOrderRule ( ) const`

3.112.4.12 getOrderRuleInt() `int Queue::getOrderRuleInt ( ) const`

3.112.4.13 GetPluginInformation() `PluginInformation * Queue::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.112.4.14 insertElement()

```
void Queue::insertElement (
    Waiting * modeldatum )
```

Here is the call graph for this function:

3.112.4.15 LoadInstance()

```
ModelDataDefinition * Queue::LoadInstance (
    Model * model,
    PersistenceRecord * fields )  [static]
```

Here is the call graph for this function:

3.112.4.16 NewInstance()

```
ModelDataDefinition * Queue::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.112.4.17 removeElement()

```
void Queue::removeElement (
    Waiting * modeldatum )
```

Here is the call graph for this function:

3.112.4.18 setAttributeName()

```
void Queue::setAttributeName (
    std::string _attributeName )
```

3.112.4.19 setOrderRule()

```
void Queue::setOrderRule (
    OrderRule _orderRule )
```

3.112.4.20 setOrderRuleInt() `void Queue::setOrderRuleInt ( int orderRule )`

Here is the call graph for this function:

3.112.4.21 show() `std::string Queue::show ( ) [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.112.4.22 size() `unsigned int Queue::size ( )`

3.112.4.23 sumAttributesFromWaiting() `double Queue::sumAttributesFromWaiting ( Util::identification attributeID )`

3.113 QueueableItem Class Reference

`#include <QueueableItem.h>`

Public Types

- enum class QueueableType : int { QUEUE = 1 , SET = 2 }

Public Member Functions

- QueueableItem (ModelDataDefinition *queueOrSet, QueueableItem::QueueableType queueable↩ Type=QueueableItem::QueueableType::QUEUE, std::string index="0")
- QueueableItem (Model *model, std::string queueName)
- bool loadInstance (PersistenceRecord *fields)
- void saveInstance (PersistenceRecord *fields, bool saveDefaultValues)
- std::string show ()
- void setIndex (std::string index)
- std::string getIndex () const
- std::string getQueueableName () const
- void setQueue (Queue *resource)
- Queue * getQueue () const
- void setSet (Set *set)
- Set * getSet () const
- void setQueueableType (QueueableType queueableType)
- QueueableType getQueueableType () const
- ModelDataDefinition * getQueueable () const
- void setElementManager (ModelDataManager *_modeldataManager)

3.113.1 Member Enumeration Documentation

3.113.1.1 QueueableType `enum class QueueableItem::QueueableType : int [strong]`

Enumerator

QUEUE	
SET	

3.113.2 Constructor & Destructor Documentation

3.113.2.1 QueueableItem() [1/2]

```
QueueableItem::QueueableItem (
    ModelDataDefinition * queueOrSet,
    QueueableItem::QueueableType queueableType = QueueableItem::QueueableType::QUEUE,
    std::string index = "0" )
```

3.113.2.2 QueueableItem() [2/2]

```
QueueableItem::QueueableItem (
    Model * model,
    std::string queueName = "" )
```

Here is the call graph for this function:

3.113.3 Member Function Documentation

3.113.3.1 getIndex() `std::string QueueableItem::getIndex ( ) const`

3.113.3.2 getQueue() `Queue * QueueableItem::getQueue ( ) const`

3.113.3.3 getQueueable() `ModelDataDefinition * QueueableItem::getQueueable ( ) const`

3.113.3.4 getQueueableName() `std::string QueueableItem::getQueueableName ( ) const`

3.113.3.5 getQueueableType() `QueueableItem::QueueableType QueueableItem::getQueueableType ( ) const`

3.113.3.6 getSet() `Set * QueueableItem::getSet ( ) const`

3.113.3.7 loadInstance()

```
bool QueueableItem::loadInstance (
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.113.3.8 saveInstance()

```
void QueueableItem::saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues )
```

Here is the call graph for this function:

3.113.3.9 setElementManager()

```
void QueueableItem::setElementManager (
    ModelDataManager * _modeldataManager )
```

3.113.3.10 setIndex()

```
void QueueableItem::setIndex (
    std::string index )
```

3.113.3.11 setQueue()

```
void QueueableItem::setQueue (
    Queue * resource )
```

Here is the call graph for this function:

3.113.3.12 setQueueableType()

```
void QueueableItem::setQueueableType (
    QueueableItem::QueueableType queueableType )
```

3.113.3.13 setSet()

```
void QueueableItem::setSet (
    Set * set )
```

Here is the call graph for this function:

3.113.3.14 show()

```
std::string QueueableItem::show ( )
```

Here is the call graph for this function:

3.114 Record Class Reference

```
#include <Record.h>
```

Inheritance 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 (const std::string expression)

- std::string getExpression () const
- void setExpressionName (std::string expressionName)
- std::string getExpressionName () const
- StatisticsCollector * getCstatExpression () const
- bool getTimeDependent () const
- void setTimeDependent (bool timeDependent)
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _initBetweenReplications ()
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.114.1 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.

3.114.2 Constructor & Destructor Documentation

3.114.2.1 Record()

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

3.114.2.2 ~Record()

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

Here is the call graph for this function:

3.114.3 Member Function Documentation

3.114.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.114.3.2 _createInternalAndAttachedData() `void Record::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.114.3.3 _initBetweenReplications() `void Record::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.114.3.4 _loadInstance() `bool Record::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.114.3.5 _onDispatchEvent()

```
void Record::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.114.3.6 _saveInstance()

```
void Record::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.114.3.7 getCstatExpression() `StatisticsCollector * Record::getCstatExpression ( ) const`

3.114.3.8 getExpression() `std::string Record::getExpression ( ) const`

3.114.3.9 getExpressionName() `std::string Record::getExpressionName ( ) const`

3.114.3.10 getFileName() `std::string Record::getFileName ( ) const`

3.114.3.11 GetPluginInformation() `PluginInformation * Record::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.114.3.12 getTimeDependent() `bool Record::getTimeDependent ( ) const`

3.114.3.13 LoadInstance()

```
ModelComponent * Record::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.114.3.14 NewInstance()

```
ModelDataDefinition * Record::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.114.3.15 setExpression()

```
void Record::setExpression (
    const std::string expression )
```

3.114.3.16 setExpressionName()

```
void Record::setExpressionName (
    std::string expressionName )
```

Here is the call graph for this function:

3.114.3.17 setFilename()

```
void Record::setFilename (
    std::string filename )
```

3.114.3.18 setTimeDependent()

```
void Record::setTimeDependent (
    bool timeDependent )
```

3.114.3.19 show()

```
std::string Record::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.115 Release Class Reference

```
#include <Release.h>
```

Inheritance diagram for Release:

Public Member Functions

- Release (Model ∗model, std::string name="")
- virtual ∼Release ()=default
- virtual std::string show ()

  virtual public methods
- void setPriority (unsigned short _priority)

- unsigned short priority () const
- List< SeizableItem * > * getReleaseRequests () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _initBetweenReplications ()
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.115.1 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. Release 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.

3.115.2 Constructor & Destructor Documentation

3.115.2.1 Release()

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

3.115.2.2 ~Release()

```
virtual Release::~Release ( ) [virtual], [default]
```

3.115.3 Member Function Documentation

3.115.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.115.3.2 _createInternalAndAttachedData()

```
void Release::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.115.3.3 _initBetweenReplications() `void Release::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.115.3.4 _loadInstance() `bool Release::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.115.3.5 _onDispatchEvent()

```
void Release::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.115.3.6 _saveInstance()

```
void Release::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.115.3.7 GetPluginInformation() `PluginInformation * Release::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.115.3.8 getReleaseRequests() `List< SeizableItem * > * Release::getReleaseRequests ( ) const`

3.115.3.9 LoadInstance() `ModelComponent * Release::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.115.3.10 NewInstance()

```
ModelDataDefinition * Release::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.115.3.11 priority()

```
unsigned short Release::priority ( ) const
```

3.115.3.12 setPriority()

```
void Release::setPriority (
    unsigned short _priority )
```

3.115.3.13 show()

```
std::string Release::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.116 Remove Class Reference

```
#include <Remove.h>
```

Inheritance diagram for Remove:

Public Types

- enum class RemoveFromType : int { QUEUE = 1 , ENTITYGROUP = 2 }

Public Member Functions

- Remove (Model ∗model, std::string name="")
- virtual ∼Remove ()=default
- virtual std::string show ()

  virtual public methods
- void setRemoveStartRank (std::string _removeFromRank)
- std::string getRemoveStartRank () const
- void setRemoveEndRank (std::string _removeEndRank)
- std::string getRemoveEndRank () const
- void setRemoveFromType (Remove::RemoveFromType _removeFromType)
- Remove::RemoveFromType getRemoveFromType () const
- void setRemoveFrom (ModelDataDefinition ∗_removeFrom)
- ModelDataDefinition ∗ getRemoveFrom () const

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.116.1 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. Rank of Entity Rank of the entity to remove from within the queue.

3.116.2 Member Enumeration Documentation

3.116.2.1 RemoveFromType `enum class Remove::RemoveFromType : int [strong]`

Enumerator

QUEUE	
ENTITYGROUP	

3.116.3 Constructor & Destructor Documentation

3.116.3.1 Remove()

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

3.116.3.2 ~Remove()

```
virtual Remove::~Remove ( ) [virtual], [default]
```

3.116.4 Member Function Documentation

3.116.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.116.4.2 _createInternalAndAttachedData() `void Remove::_createInternalAndAttachedData ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.116.4.3 _loadInstance() `bool Remove::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.116.4.4 _onDispatchEvent()

```
void Remove::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.116.4.5 _saveInstance()

```
void Remove::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.116.4.6 GetPluginInformation() `PluginInformation * Remove::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.116.4.7 getRemoveEndRank() `std::string Remove::getRemoveEndRank ( ) const`

3.116.4.8 getRemoveFrom() `ModelDataDefinition * Remove::getRemoveFrom ( ) const`

3.116.4.9 getRemoveFromType() `Remove::RemoveFromType Remove::getRemoveFromType ( ) const`

3.116.4.10 getRemoveStartRank() `std::string Remove::getRemoveStartRank ( ) const`

3.116.4.11 LoadInstance()

```
ModelComponent * Remove::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.116.4.12 NewInstance()

```
ModelDataDefinition * Remove::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.116.4.13 setRemoveEndRank()

```
void Remove::setRemoveEndRank (
    std::string _removeEndRank )
```

3.116.4.14 setRemoveFrom()

```
void Remove::setRemoveFrom (
    ModelDataDefinition * _removeFrom )
```

3.116.4.15 setRemoveFromType()

```
void Remove::setRemoveFromType (
    Remove::RemoveFromType _removeFromType )
```

3.116.4.16 setRemoveStartRank()

```
void Remove::setRemoveStartRank (
    std::string _removeFromRank )
```

3.116.4.17 show() `std::string Remove::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.117 Resource Class Reference

```
#include <Resource.h>
```

Inheritance diagram for Resource:

Public Types

- enum class ResourceState : int {
  IDLE = 1 , BUSY = 2 , FAILED = 3 , INACTIVE = 4 ,
  OTHER = 5 }
- typedef std::function< void(Resource *) > ResourceEventHandler
- typedef std::pair< std::pair< ResourceEventHandler, ModelComponent * >, unsigned int > SortedResourceEventHandler

Public Member Functions

- Resource (Model ∗model, std::string name="")
- virtual ∼Resource ()=default
- virtual std::string show ()
- bool seize (unsigned int quantity, double priority=0)
- void release (unsigned int quantity)
- void insertFailure (Failure ∗failure)
- void removeFailure (Failure ∗failure)
- double getInstantCapacityUtilization () const
- double getCapacityUtilization () const
- double getSeizedUtilization () const
- double getLastTimeSeized () const
- void addReleaseResourceEventHandler (ResourceEventHandler eventHandler, ModelComponent ∗component, unsigned int priority)
- void setResourceState (ResourceState _resourceState)
- Resource::ResourceState getResourceState () const
- void setCapacity (unsigned int capacity)
- unsigned int getCapacity () const
- void setCostBusyTimeUnit (double _costBusyTimeUnit)
- double getCostBusyTimeUnit () const
- void setCostIdleTimeUnit (double _costIdleTimeUnit)
- double getCostIdleTimeUnit () const
- void setCostPerUse (double _costPerUse)
- double getCostPerUse () const
- void setCapacitySchedule (Schedule ∗_capacitySchedule)
- Schedule ∗ getCapacitySchedule () const
- unsigned int getNumberBusy () const

Static Public Member Functions

- template<typename Class >
  static ResourceEventHandler SetResourceEventHandler (void(Class::∗function)(Resource ∗), Class ∗object)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()
- virtual void _initBetweenReplications ()

Friends

- class Failure

Additional Inherited Members

3.117.1 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/TimeUnit Cost per timeUnit 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/timeUnit cost. The busy cost per timeUnit is automatically converted to the appropriate base time unit specified within the Replication Parameters page of the Run > Setup menu item. Idle/TimeUnit Cost per timeUnit 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/timeUnit cost. The idle cost per timeUnit 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.

3.117.2 Member Typedef Documentation

3.117.2.1 ResourceEventHandler `typedef std::function<void(Resource*) > Resource::ResourceEventHandler`

3.117.2.2 SortedResourceEventHandler `typedef std::pair<std::pair<ResourceEventHandler, ModelComponent*>, unsigned int> Resource::SortedResourceEventHandler`

3.117.3 Member Enumeration Documentation

3.117.3.1 ResourceState `enum class Resource::ResourceState : int [strong]`

Enumerator

IDLE	
BUSY	
FAILED	
INACTIVE	
OTHER	

3.117.4 Constructor & Destructor Documentation

3.117.4.1 Resource()

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

Handlers are sorted by priority

3.117.4.2 ~Resource()

```
virtual Resource::~Resource ( ) [virtual], [default]
```

3.117.5 Member Function Documentation

3.117.5.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.117.5.2 _createInternalAndAttachedData()

```
void Resource::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.117.5.3 _initBetweenReplications() `void Resource::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.117.5.4 _loadInstance() `bool Resource::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.117.5.5 _saveInstance() `void Resource::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.117.5.6 addReleaseResourceEventHandler()

```
void Resource::addReleaseResourceEventHandler (
    ResourceEventHandler eventHandler,
    ModelComponent * component,
    unsigned int priority )
```

3.117.5.7 getCapacity()

```
unsigned int Resource::getCapacity ( ) const
```

Here is the call graph for this function:

3.117.5.8 getCapacitySchedule()

```
Schedule * Resource::getCapacitySchedule ( ) const
```

3.117.5.9 getCapacityUtilization()

```
double Resource::getCapacityUtilization ( ) const
```

Here is the call graph for this function:

3.117.5.10 getCostBusyTimeUnit() `double Resource::getCostBusyTimeUnit ( ) const`

3.117.5.11 getCostIdleTimeUnit() `double Resource::getCostIdleTimeUnit ( ) const`

3.117.5.12 getCostPerUse() `double Resource::getCostPerUse ( ) const`

3.117.5.13 getInstantCapacityUtilization() `double Resource::getInstantCapacityUtilization ( ) const`

Here is the call graph for this function:

3.117.5.14 getLastTimeSeized() `double Resource::getLastTimeSeized ( ) const`

3.117.5.15 getNumberBusy() `unsigned int Resource::getNumberBusy ( ) const`

3.117.5.16 GetPluginInformation() `PluginInformation * Resource::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.117.5.17 getResourceState() `Resource::ResourceState Resource::getResourceState ( ) const`

3.117.5.18 getSeizedUtilization() `double Resource::getSeizedUtilization ( ) const`

Here is the call graph for this function:

3.117.5.19 insertFailure() `void Resource::insertFailure ( Failure * failure )`

Here is the call graph for this function:

3.117.5.20 LoadInstance() `ModelDataDefinition * Resource::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.117.5.21 NewInstance()

```
ModelDataDefinition * Resource::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.117.5.22 release()

```
void Resource::release (
    unsigned int quantity )
```

Here is the call graph for this function:

3.117.5.23 removeFailure()

```
void Resource::removeFailure (
    Failure * failure )
```

Here is the call graph for this function:

3.117.5.24 seize()

```
bool Resource::seize (
    unsigned int quantity,
    double priority = 0 )
```

Here is the call graph for this function:

3.117.5.25 setCapacity()

```
void Resource::setCapacity (
    unsigned int capacity )
```

3.117.5.26 setCapacitySchedule()

```
void Resource::setCapacitySchedule (
    Schedule * _capacitySchedule )
```

3.117.5.27 setCostBusyTimeUnit()

```
void Resource::setCostBusyTimeUnit (
    double _costBusyTimeUnit )
```

3.117.5.28 setCostIdleTimeUnit()

```
void Resource::setCostIdleTimeUnit (
    double _costIdleTimeUnit )
```

3.117.5.29 setCostPerUse()

```
void Resource::setCostPerUse (
    double _costPerUse )
```

3.117.5.30 SetResourceEventHandler()

```
template<typename Class >
static ResourceEventHandler Resource::SetResourceEventHandler (
    void(Class::*)(Resource *) function,
    Class * object )  [inline], [static]
```

3.117.5.31 setResourceState() `void Resource::setResourceState ( ResourceState _resourceState )`

3.117.5.32 show() `std::string Resource::show ( ) [virtual]`

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.117.6 Friends And Related Function Documentation

3.117.6.1 Failure `friend class Failure [friend]`

3.118 Sampler_if::RNG_Parameters Struct Reference

`#include <Sampler_if.h>`

Inheritance diagram for Sampler_if::RNG_Parameters:

Public Member Functions

- virtual ~RNG_Parameters ()=default

3.118.1 Detailed Description

class that encapsulates attributes required to generate random numbers, which depends on the generation method used.

3.118.2 Constructor & Destructor Documentation

3.118.2.1 ~RNG_Parameters() `virtual Sampler_if::RNG_Parameters::~RNG_Parameters ( ) [virtual], [default]`

3.119 Route Class Reference

```
#include <Route.h>
```

Inheritance diagram for Route:

Public Types

- enum class DestinationType : int { Station = 0 , Sequence = 1 , Label = 2 }

Public Member Functions

- Route (Model ∗model, std::string name="")

- virtual ~Route ()=default
- virtual std::string show ()

  virtual public methods
- void setStationName (std::string stationName)
- void setStation (Station *_station)
- Station * getStation () const
- void setStationExpression (std::string _stationExpression)
- std::string getStationExpression () const
- void setRouteTimeExpression (std::string _routeTimeExpression)
- void setRouteTimeExpression (std::string _routeTimeExpression, Util::TimeUnit _routeTimeTimeUnit)
- std::string getRouteTimeExpression () const
- void setRouteTimeTimeUnit (Util::TimeUnit _routeTimeTimeUnit)
- Util::TimeUnit getRouteTimeTimeUnit () const
- void setRouteDestinationType (DestinationType _routeDestinationType)
- Route::DestinationType getRouteDestinationType () const
- void setLabel (Label *_label)
- Label * getLabel () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.119.1 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.

3.119.2 Member Enumeration Documentation

3.119.2.1 DestinationType `enum class Route::DestinationType : int [strong]`

Enumerator

Station	
Sequence	
Label	

3.119.3 Constructor & Destructor Documentation

3.119.3.1 Route()

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

3.119.3.2 ~Route() `virtual Route::~Route ( ) [virtual], [default]`

3.119.4 Member Function Documentation

3.119.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.119.4.2 _createInternalAndAttachedData() `void Route::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.119.4.3 _loadInstance() `bool Route::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.119.4.4 _onDispatchEvent()

```
void Route::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.119.4.5 _saveInstance()

```
void Route::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.119.4.6 getLabel()

```
Label * Route::getLabel ( ) const
```

3.119.4.7 GetPluginInformation()

```
PluginInformation * Route::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.119.4.8 getRouteDestinationType()

```
Route::DestinationType Route::getRouteDestinationType ( ) const
```

3.119.4.9 getRouteTimeExpression() `std::string Route::getRouteTimeExpression ( ) const`

3.119.4.10 getRouteTimeTimeUnit() `Util::TimeUnit Route::getRouteTimeTimeUnit ( ) const`

3.119.4.11 getStation() `Station * Route::getStation ( ) const`

3.119.4.12 getStationExpression() `std::string Route::getStationExpression ( ) const`

3.119.4.13 LoadInstance()

```
ModelComponent * Route::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.119.4.14 NewInstance()

```
ModelDataDefinition * Route::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.119.4.15 setLabel()

```
void Route::setLabel (
    Label * _label )
```

3.119.4.16 setRouteDestinationType()

```
void Route::setRouteDestinationType (
    DestinationType _routeDestinationType )
```

3.119.4.17 setRouteTimeExpression() [1/2]

```
void Route::setRouteTimeExpression (
    std::string _routeTimeExpression )
```

3.119.4.18 setRouteTimeExpression() [2/2]

```
void Route::setRouteTimeExpression (
    std::string _routeTimeExpression,
    Util::TimeUnit _routeTimeTimeUnit )
```

3.119.4.19 setRouteTimeTimeUnit()

```
void Route::setRouteTimeTimeUnit (
    Util::TimeUnit _routeTimeTimeUnit )
```

3.119.4.20 setStation()

```
void Route::setStation (
    Station * _station )
```

3.119.4.21 setStationExpression()

```
void Route::setStationExpression (
    std::string _stationExpression )
```

3.119.4.22 setStationName()

```
void Route::setStationName (
    std::string stationName )
```

Here is the call graph for this function:

3.119.4.23 show() `std::string Route::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.120 Sampler_if Class Reference

```
#include <Sampler_if.h>
```

Inheritance 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 sampleBeta (double alpha, double beta)=0
- virtual double sampleErlang (double mean, int M, double offset=0.0)=0
- virtual double sampleExponential (double mean, double offset=0.0)=0
- virtual double sampleGamma (double alpha, double beta, double offset=0.0)=0
- virtual double sampleGumbell (double mode, double scale)=0
- virtual double sampleLogNormal (double mean, double stddev, double offset=0.0)=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 double sampleBinomial (int trials, double p)=0
- virtual double sampleBernoulli (double p)=0
- virtual double sampleDiscrete (double prob, double value,...)=0
- virtual double sampleDiscrete (double *prob, double *value, int size)=0
- virtual double sampleGeometric (double p)=0
- virtual void setRNGparameters (RNG_Parameters *param)=0
- virtual RNG_Parameters * getRNGparameters () const =0

3.120.1 Detailed Description

Interface that describes the methods to be implemented by classes that generate random values that follow a specific probability distribution.

3.120.2 Member Function Documentation

3.120.2.1 getRNGparameters() `virtual RNG_Parameters * Sampler_if::getRNGparameters ( ) const [pure virtual]`

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.2 random() `virtual double Sampler_if::random ( ) [pure virtual]`

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.3 sampleBernoulli() `virtual double Sampler_if::sampleBernoulli ( double p ) [pure virtual]`

Implemented in SamplerDefaultImpl1.

3.120.2.4 sampleBeta() [1/2] `virtual double Sampler_if::sampleBeta ( double alpha, double beta ) [pure virtual]`

Implemented in SamplerDefaultImpl1.

3.120.2.5 sampleBeta() [2/2]

```
virtual double Sampler_if::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit )  [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.6 sampleBinomial()

```
virtual double Sampler_if::sampleBinomial (
    int trials,
    double p )  [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.7 sampleDiscrete() [1/2]

```
virtual double Sampler_if::sampleDiscrete (
    double * prob,
    double * value,
    int size )  [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.8 sampleDiscrete() [2/2]

```
virtual double Sampler_if::sampleDiscrete (
    double prob,
    double value,
     ...  )  [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.9 sampleErlang()

```
virtual double Sampler_if::sampleErlang (
    double mean,
    int M,
    double offset = 0.0 )  [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.10 sampleExponential()

```
virtual double Sampler_if::sampleExponential (
    double mean,
    double offset = 0.0 )  [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.11 sampleGamma()

```
virtual double Sampler_if::sampleGamma (
    double alpha,
    double beta,
    double offset = 0.0 ) [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.12 sampleGeometric()

```
virtual double Sampler_if::sampleGeometric (
    double p ) [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.13 sampleGumbell()

```
virtual double Sampler_if::sampleGumbell (
    double mode,
    double scale ) [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.14 sampleLogNormal()

```
virtual double Sampler_if::sampleLogNormal (
    double mean,
    double stddev,
    double offset = 0.0 ) [pure virtual]
```

Implemented in SamplerDefaultImpl1.

3.120.2.15 sampleNormal()

```
virtual double Sampler_if::sampleNormal (
    double mean,
    double stddev ) [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.16 sampleTriangular()

```
virtual double Sampler_if::sampleTriangular (
    double min,
    double mode,
    double max ) [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.17 sampleUniform()

```
virtual double Sampler_if::sampleUniform (
    double min,
    double max ) [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.18 sampleWeibull()

```
virtual double Sampler_if::sampleWeibull (
    double alpha,
    double scale ) [pure virtual]
```

Implemented in SamplerBoostImpl, and SamplerDefaultImpl1.

3.120.2.19 setRNGparameters()

```
virtual void Sampler_if::setRNGparameters (
    RNG_Parameters * param ) [pure virtual]
```

Implemented in SamplerDefaultImpl1, and SamplerBoostImpl.

3.121 SamplerBoostImpl Class Reference

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

Inheritance 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

3.121.1 Constructor & Destructor Documentation

3.121.1.1 SamplerBoostImpl() `SamplerBoostImpl::SamplerBoostImpl ( )`

3.121.1.2 ~SamplerBoostImpl() `virtual SamplerBoostImpl::~SamplerBoostImpl ( ) [virtual], [default]`

3.121.2 Member Function Documentation

3.121.2.1 getRNGparameters() `Sampler_if::RNG_Parameters * SamplerBoostImpl::getRNGparameters ( ) const [virtual]`

Implements Sampler_if.

3.121.2.2 random() `double SamplerBoostImpl::random ( ) [virtual]`

Implements Sampler_if.

3.121.2.3 reset() `void SamplerBoostImpl::reset ( )`

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

3.121.2.4 sampleBeta()

```
double SamplerBoostImpl::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [virtual]
```

Implements Sampler_if.

3.121.2.5 sampleDiscrete()

```
double SamplerBoostImpl::sampleDiscrete (
    double acumProb,
    double value,
    ... ) [virtual]
```

Implements Sampler_if.

3.121.2.6 sampleErlang()

```
double SamplerBoostImpl::sampleErlang (
    double mean,
    int M ) [virtual]
```

3.121.2.7 sampleExponential()

```
double SamplerBoostImpl::sampleExponential (
    double mean ) [virtual]
```

3.121.2.8 sampleGamma()

```
double SamplerBoostImpl::sampleGamma (
    double mean,
    double alpha ) [virtual]
```

3.121.2.9 sampleLogNormal()

```
double SamplerBoostImpl::sampleLogNormal (
    double mean,
    double stddev ) [virtual]
```

3.121.2.10 sampleNormal()

```
double SamplerBoostImpl::sampleNormal (
    double mean,
    double stddev ) [virtual]
```

Implements Sampler_if.

3.121.2.11 sampleTriangular()

```
double SamplerBoostImpl::sampleTriangular (
    double min,
    double mode,
    double max ) [virtual]
```

Implements Sampler_if.

3.121.2.12 sampleUniform()

```
double SamplerBoostImpl::sampleUniform (
    double min,
    double max ) [virtual]
```

Implements Sampler_if.

3.121.2.13 sampleWeibull()

```
double SamplerBoostImpl::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

Implements Sampler_if.

3.121.2.14 setRNGparameters()

```
void SamplerBoostImpl::setRNGparameters (
    Sampler_if::RNG_Parameters * param ) [virtual]
```

Implements Sampler_if.

3.122 SamplerDefaultImpl1 Class Reference

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

Inheritance 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 sampleBeta (double alpha, double beta)
- virtual double sampleErlang (double mean, int M, double offset=0.0)
- virtual double sampleExponential (double mean, double offset=0.0)
- virtual double sampleGamma (double alpha, double beta, double offset=0.0)
- virtual double sampleGumbell (double mode, double scale)
- virtual double sampleLogNormal (double mean, double stddev, double offset=0.0)
- 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)
- virtual double sampleBinomial (int trials, double p)
- virtual double sampleBernoulli (double p)
- virtual double sampleDiscrete (double prob, double value,...)
- virtual double sampleDiscrete (double *prob, double *value, int size)
- virtual double sampleGeometric (double p)
- 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

3.122.1 Constructor & Destructor Documentation

3.122.1.1 SamplerDefaultImpl1() `SamplerDefaultImpl1::SamplerDefaultImpl1 ( )`

Here is the call graph for this function:

3.122.1.2 ~SamplerDefaultImpl1() `virtual SamplerDefaultImpl1::~SamplerDefaultImpl1 ( ) [virtual], [default]`

3.122.2 Member Function Documentation

3.122.2.1 getRNGparameters() `Sampler_if::RNG_Parameters * SamplerDefaultImpl1::getRNGparameters ( ) const [virtual]`

Implements Sampler_if.

3.122.2.2 random() `double SamplerDefaultImpl1::random ( ) [virtual]`

Implements Sampler_if.

3.122.2.3 reset() `void SamplerDefaultImpl1::reset ( )`

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

3.122.2.4 sampleBernoulli() `double SamplerDefaultImpl1::sampleBernoulli ( double p ) [virtual]`

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.5 sampleBeta() [1/2] `double SamplerDefaultImpl1::sampleBeta ( double alpha, double beta ) [virtual]`

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.6 sampleBeta() [2/2]

```
double SamplerDefaultImpl1::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.7 sampleBinomial()

```
double SamplerDefaultImpl1::sampleBinomial (
    int trials,
    double p ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.8 sampleDiscrete() [1/2]

```
double SamplerDefaultImpl1::sampleDiscrete (
    double * prob,
    double * value,
    int size ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.9 sampleDiscrete() [2/2]

```
double SamplerDefaultImpl1::sampleDiscrete (
    double prob,
    double value,
    ... ) [virtual]
```

Implements Sampler_if.

3.122.2.10 sampleErlang()

```
double SamplerDefaultImpl1::sampleErlang (
    double mean,
    int M,
    double offset = 0.0 ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.11 sampleExponential()

```
double SamplerDefaultImpl1::sampleExponential (
    double mean,
    double offset = 0.0 ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.12 sampleGamma()

```
double SamplerDefaultImpl1::sampleGamma (
    double alpha,
    double beta,
    double offset = 0.0 ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.13 sampleGeometric()

```
double SamplerDefaultImpl1::sampleGeometric (
    double p ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.14 sampleGumbell()

```
double SamplerDefaultImpl1::sampleGumbell (
    double mode,
    double scale ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.15 sampleLogNormal()

```
double SamplerDefaultImpl1::sampleLogNormal (
    double mean,
    double stddev,
    double offset = 0.0 ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.16 sampleNormal()

```
double SamplerDefaultImpl1::sampleNormal (
    double mean,
    double stddev ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.17 sampleTriangular()

```
double SamplerDefaultImpl1::sampleTriangular (
    double min,
    double mode,
    double max ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.18 sampleUniform()

```
double SamplerDefaultImpl1::sampleUniform (
    double min,
    double max ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.19 sampleWeibull()

```
double SamplerDefaultImpl1::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

Implements Sampler_if.

Here is the call graph for this function:

3.122.2.20 setRNGparameters()

```
void SamplerDefaultImpl1::setRNGparameters (
    Sampler_if::RNG_Parameters * param ) [virtual]
```

Implements Sampler_if.

3.123 ScenarioExperiment_if Class Reference

```
#include <ScenarioExperiment_if.h>
```

3.124 SchedulableItem Class Reference

```
#include <Schedule.h>
```

Public Types

- enum class Rule : int { IGNORE = 1 , PREEMPT = 2 , WAIT = 3 }

Public Member Functions

- SchedulableItem (std::string expression, double duration, SchedulableItem::Rule=SchedulableItem::Rule::IGNORE)
- void setDuration (double duration)
- double getDuration () const
- void setExpression (std::string expression)
- std::string getExpression () const
- void setRule (SchedulableItem::Rule rule)
- SchedulableItem::Rule getRule () const

3.124.1 Member Enumeration Documentation

3.124.1.1 Rule `enum class SchedulableItem::Rule : int [strong]`

Enumerator

IGNORE	
PREEMPT	
WAIT	

3.124.2 Constructor & Destructor Documentation

3.124.2.1 SchedulableItem()

```
SchedulableItem::SchedulableItem (
    std::string expression,
    double duration,
    SchedulableItem::Rule = SchedulableItem::Rule::IGNORE ) [inline]
```

3.124.3 Member Function Documentation

3.124.3.1 getDuration() `double SchedulableItem::getDuration ( ) const [inline]`

3.124.3.2 getExpression() `std::string SchedulableItem::getExpression ( ) const [inline]`

3.124.3.3 getRule() `SchedulableItem::Rule SchedulableItem::getRule ( ) const [inline]`

3.124.3.4 setDuration() `void SchedulableItem::setDuration ( double duration ) [inline]`

3.124.3.5 setExpression() `void SchedulableItem::setExpression ( std::string expression ) [inline]`

3.124.3.6 setRule() `void SchedulableItem::setRule ( SchedulableItem::Rule rule ) [inline]`

3.125 Schedule Class Reference

```
#include <Schedule.h>
```

Inheritance diagram for Schedule:

Public Member Functions

- Schedule (Model *model, std::string name="")
- virtual ~Schedule ()=default
- virtual std::string show ()
- std::string getExpression ()
- List< SchedulableItem * > * getSchedulableItems () const
- void setRepeatAfterLast (bool _repeatAfterLast)
- bool isRepeatAfterLast () const

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation ∗ _getParserChangesInformation ()
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()
- virtual void _addProperty (PropertyBase ∗property)

Additional Inherited Members

3.125.1 Constructor & Destructor Documentation

3.125.1.1 Schedule()

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

3.125.1.2 ∼Schedule()

```
virtual Schedule::~Schedule ( ) [virtual], [default]
```

3.125.2 Member Function Documentation

3.125.2.1 _addProperty()

```
void Schedule::_addProperty (
    PropertyBase * property ) [protected], [virtual]
```

Reimplemented from ModelDataDefinition.

3.125.2.2 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.125.2.3 _createInternalAndAttachedData()

```
void Schedule::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.125.2.4 _getParserChangesInformation()

```
ParserChangesInformation * Schedule::_getParserChangesInformation ( ) [protected], [virtual]
```

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.125.2.5 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

3.125.2.6 _loadInstance()

```
bool Schedule::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.125.2.7 _saveInstance()

```
void Schedule::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.125.2.8 getExpression()

```
std::string Schedule::getExpression ( )
```

Here is the call graph for this function:

3.125.2.9 GetPluginInformation() `PluginInformation * Schedule::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.125.2.10 getSchedulableItems() `List< SchedulableItem * > * Schedule::getSchedulableItems ( ) const`

3.125.2.11 isRepeatAfterLast() `bool Schedule::isRepeatAfterLast ( ) const`

3.125.2.12 LoadInstance() `ModelDataDefinition * Schedule::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.125.2.13 NewInstance()

```
ModelDataDefinition * Schedule::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.125.2.14 setRepeatAfterLast()

```
void Schedule::setRepeatAfterLast (
    bool _repeatAfterLast )
```

3.125.2.15 show()

```
std::string Schedule::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.126 Search Class Reference

```
#include <Search.h>
```

Inheritance diagram for Search:

Public Types

- enum class SearchInType : int { QUEUE = 1 , ENTITYGROUP = 2 }

Public Member Functions

- Search (Model ∗model, std::string name="")

- virtual ~Search ()=default
- virtual std::string show ()

  virtual public methods
- void setSaveFounRankAttribute (std::string _saveFounRankAttribute)
- std::string getSaveFounRankAttribute () const
- void setSearchCondition (std::string _searchCondition)
- std::string getSearchCondition () const
- void setEndRank (std::string _endRank)
- std::string getEndRank () const
- void setStartRank (std::string _startRank)
- std::string getStartRank () const
- void setSearchInName (std::string searchInName)
- std::string getSearchInName () const
- void setSearchIn (ModelDataDefinition *_searchIn)
- ModelDataDefinition * getSearchIn () const
- void setSearchInType (Search::SearchInType _searchInType)
- Search::SearchInType getSearchInType () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.126.1 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.

3.126.2 Member Enumeration Documentation

3.126.2.1 SearchInType `enum class Search::SearchInType : int [strong]`

Enumerator

QUEUE	
ENTITYGROUP	

3.126.3 Constructor & Destructor Documentation

3.126.3.1 Search()

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

3.126.3.2 ~Search() `virtual Search::~Search ( ) [virtual], [default]`

3.126.4 Member Function Documentation

3.126.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.126.4.2 _createInternalAndAttachedData() `void Search::_createInternalAndAttachedData ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.126.4.3 _loadInstance() `bool Search::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.126.4.4 _onDispatchEvent()

```
void Search::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.126.4.5 _saveInstance()

```
void Search::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.126.4.6 getEndRank() `std::string Search::getEndRank ( ) const`

3.126.4.7 GetPluginInformation() `PluginInformation * Search::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.126.4.8 getSaveFounRankAttribute() `std::string Search::getSaveFounRankAttribute ( ) const`

3.126.4.9 getSearchCondition() `std::string Search::getSearchCondition ( ) const`

3.126.4.10 getSearchIn() `ModelDataDefinition * Search::getSearchIn ( ) const`

3.126.4.11 getSearchInName() `std::string Search::getSearchInName ( ) const`

Here is the call graph for this function:

3.126.4.12 getSearchInType() `Search::SearchInType Search::getSearchInType ( ) const`

3.126.4.13 getStartRank() `std::string Search::getStartRank ( ) const`

3.126.4.14 LoadInstance() `ModelComponent * Search::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.126.4.15 NewInstance() `ModelDataDefinition * Search::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.126.4.16 setEndRank()

```
void Search::setEndRank (
    std::string _endRank )
```

3.126.4.17 setSaveFounRankAttribute()

```
void Search::setSaveFounRankAttribute (
    std::string _saveFounRankAttribute )
```

3.126.4.18 setSearchCondition()

```
void Search::setSearchCondition (
    std::string _searchCondition )
```

3.126.4.19 setSearchIn()

```
void Search::setSearchIn (
    ModelDataDefinition * _searchIn )
```

3.126.4.20 setSearchInName()

```
void Search::setSearchInName (
    std::string searchInName )
```

Here is the call graph for this function:

3.126.4.21 setSearchInType()

```
void Search::setSearchInType (
    Search::SearchInType _searchInType )
```

3.126.4.22 setStartRank()

```
void Search::setStartRank (
    std::string _startRank )
```

3.126.4.23 show() `std::string Search::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.127 SeizableItem Class Reference

```
#include <SeizableItem.h>
```

Public Types

- enum class SelectionRule : int { CYCLICAL = 1 , RANDOM = 2 , SPECIFICMEMBER = 3 , LARGESTREMAININGCAPACITY = 4 , SMALLESTNUMBERBUSY = 5 , PREFEREDORDER = 6 }
- enum class SeizableType : int { RESOURCE = 1 , SET = 2 }

Public Member Functions

- SeizableItem (ModelDataDefinition *resourceOrSet, std::string quantityExpression="1", SeizableItem::SelectionRule selectionRule=SeizableItem::SelectionRule::LARGESTREMAININGCAPACITY, std::string saveAttribute="", std::string index="0")
- SeizableItem (Model *model, std::string resourceName, std::string quantityExpression="1", SeizableItem::SelectionRule selectionRule=SeizableItem::SelectionRule::LARGESTREMAININGCAPACITY, std::string saveAttribute="", std::string index="0")
- SeizableItem (SeizableItem *original)
- bool loadInstance (PersistenceRecord *fields, unsigned int parentIndex)
- void saveInstance (PersistenceRecord *fields, unsigned int parentIndex, bool saveDefault)
- bool loadInstance (PersistenceRecord *fields)
- void saveInstance (PersistenceRecord *fields, bool saveDefaults)
- std::string show ()
- void setIndex (std::string index)
- std::string 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 setSeizableType (SeizableType resourceType)
- SeizableType getSeizableType () const
- void setLastMemberSeized (unsigned int lastMemberSeized)
- unsigned int getLastMemberSeized () const
- ModelDataDefinition * getSeizable () const
- void setElementManager (ModelDataManager *modeldataManager)
- void setLastPreferedOrder (unsigned int lastPreferedOrder)
- unsigned int getLastPreferedOrder () const

3.127.1 Member Enumeration Documentation

3.127.1.1 SeizableType `enum class SeizableItem::SeizableType : int [strong]`

Enumerator

RESOURCE	
SET	

3.127.1.2 SelectionRule `enum class SeizableItem::SelectionRule : int [strong]`

Enumerator

CYCLICAL	
RANDOM	
SPECIFICMEMBER	
LARGESTREMAININGCAPACITY	
SMALLESTNUMBERBUSY	
PREFEREDORDER	

3.127.2 Constructor & Destructor Documentation

3.127.2.1 SeizableItem() [1/3]

```
SeizableItem::SeizableItem (
    ModelDataDefinition * resourceOrSet,
    std::string quantityExpression = "1",
    SeizableItem::SelectionRule selectionRule = SeizableItem::SelectionRule::LARGESTREMAININGCAPACITY
    std::string saveAttribute = "",
    std::string index = "0" )
```

3.127.2.2 SeizableItem() [2/3]

```
SeizableItem::SeizableItem (
    Model * model,
    std::string resourceName,
    std::string quantityExpression = "1",
    SeizableItem::SelectionRule selectionRule = SeizableItem::SelectionRule::LARGESTREMAININGCAPACITY
    std::string saveAttribute = "",
    std::string index = "0" )
```

Here is the call graph for this function:

3.127.2.3 SeizableItem() [3/3]

```
SeizableItem::SeizableItem (
    SeizableItem * original )
```

Here is the call graph for this function:

3.127.3 Member Function Documentation

3.127.3.1 getIndex() `std::string SeizableItem::getIndex ( ) const`

3.127.3.2 getLastMemberSeized() `unsigned int SeizableItem::getLastMemberSeized ( ) const`

3.127.3.3 getLastPreferedOrder() `unsigned int SeizableItem::getLastPreferedOrder ( ) const`

3.127.3.4 getQuantityExpression() `std::string SeizableItem::getQuantityExpression ( ) const`

3.127.3.5 getResource() `Resource * SeizableItem::getResource ( ) const`

3.127.3.6 getResourceName() `std::string SeizableItem::getResourceName ( ) const`

3.127.3.7 getSaveAttribute() `std::string SeizableItem::getSaveAttribute ( ) const`

3.127.3.8 getSeizable() `ModelDataDefinition * SeizableItem::getSeizable ( ) const`

3.127.3.9 getSeizableType() `SeizableItem::SeizableType SeizableItem::getSeizableType ( ) const`

3.127.3.10 getSelectionRule() `SeizableItem::SelectionRule SeizableItem::getSelectionRule ( ) const`

3.127.3.11 getSet() `Set * SeizableItem::getSet ( ) const`

3.127.3.12 loadInstance() [1/2]

```
bool SeizableItem::loadInstance (
    PersistenceRecord * fields )
```

Here is the call graph for this function:

3.127.3.13 loadInstance() [2/2]

```
bool SeizableItem::loadInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex )
```

Here is the call graph for this function:

3.127.3.14 saveInstance() [1/2]

```
void SeizableItem::saveInstance (
    PersistenceRecord * fields,
    bool saveDefaults )
```

Here is the call graph for this function:

3.127.3.15 saveInstance() [2/2]

```
void SeizableItem::saveInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex,
    bool saveDefault )
```

Here is the call graph for this function:

3.127.3.16 setElementManager()

```
void SeizableItem::setElementManager (
    ModelDataManager * modeldataManager )
```

3.127.3.17 setIndex()

```
void SeizableItem::setIndex (
    std::string index )
```

3.127.3.18 setLastMemberSeized()

```
void SeizableItem::setLastMemberSeized (
    unsigned int lastMemberSeized )
```

3.127.3.19 setLastPreferedOrder()

```
void SeizableItem::setLastPreferedOrder (
    unsigned int lastPreferedOrder )
```

3.127.3.20 setQuantityExpression()

```
void SeizableItem::setQuantityExpression (
    std::string quantityExpression )
```

3.127.3.21 setResource()

```
void SeizableItem::setResource (
    Resource * resource )
```

Here is the call graph for this function:

3.127.3.22 setSaveAttribute()

```
void SeizableItem::setSaveAttribute (
    std::string saveAttribute )
```

3.127.3.23 setSeizableType()

```
void SeizableItem::setSeizableType (
    SeizableItem::SeizableType resourceType )
```

3.127.3.24 setSelectionRule()

```
void SeizableItem::setSelectionRule (
    SeizableItem::SelectionRule selectionRule )
```

3.127.3.25 setSet()

```
void SeizableItem::setSet (
    Set * set )
```

Here is the call graph for this function:

3.127.3.26 show() `std::string SeizableItem::show ( )`

Here is the call graph for this function:

3.128 Seize Class Reference

```
#include <Seize.h>
```

Inheritance diagram for Seize:

Classes

- struct DEFAULT_VALUES

Public Member Functions

- Seize (Model *model, std::string name="")

- virtual ~Seize ()=default
- virtual std::string show ()

  virtual public methods
- void setPriority (unsigned short _priority)
- unsigned short getPriority () const
- void setAllocationType (Util::AllocationType _allocationType)
- Util::AllocationType getAllocationType () const
- void setQueue (Queue *queue)

  Deprected.
- List< SeizableItem * > * getSeizeRequests () const
- void setQueueableItem (QueueableItem *_queueableItem)
- QueueableItem * getQueueableItem () const
- void setPriorityExpression (std::string _priorityExpression)
- std::string getPriorityExpression () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Public Attributes

- const struct Seize::DEFAULT_VALUES DEFAULT

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _initBetweenReplications ()
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.128.1 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.

3.128.2 Constructor & Destructor Documentation

3.128.2.1 Seize()

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

Here is the call graph for this function:

3.128.2.2 ~Seize() `virtual Seize::~Seize ( ) [virtual], [default]`

3.128.3 Member Function Documentation

3.128.3.1 _check() `bool Seize::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.128.3.2 _createInternalAndAttachedData() `void Seize::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.128.3.3 _initBetweenReplications() `void Seize::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.128.3.4 _loadInstance() `bool Seize::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.128.3.5 _onDispatchEvent()

```
void Seize::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.128.3.6 _saveInstance()

```
void Seize::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.128.3.7 getAllocationType() `Util::AllocationType Seize::getAllocationType ( ) const`

3.128.3.8 GetPluginInformation() `PluginInformation * Seize::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.128.3.9 getPriority() `unsigned short Seize::getPriority ( ) const`

3.128.3.10 getPriorityExpression() `std::string Seize::getPriorityExpression ( ) const`

3.128.3.11 getQueueableItem() `QueueableItem * Seize::getQueueableItem ( ) const`

3.128.3.12 getSeizeRequests() `List< SeizableItem * > * Seize::getSeizeRequests ( ) const`

3.128.3.13 LoadInstance()

```
ModelComponent * Seize::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.128.3.14 NewInstance()

```
ModelDataDefinition * Seize::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.128.3.15 setAllocationType()

```
void Seize::setAllocationType (
    Util::AllocationType _allocationType )
```

3.128.3.16 setPriority()

```
void Seize::setPriority (
    unsigned short _priority )
```

3.128.3.17 setPriorityExpression()

```
void Seize::setPriorityExpression (
    std::string _priorityExpression )
```

3.128.3.18 setQueue()

```
void Seize::setQueue (
    Queue * queue )
```

Deprected.

Here is the call graph for this function:

3.128.3.19 setQueueableItem()

```
void Seize::setQueueableItem (
    QueueableItem * _queueableItem )
```

3.128.3.20 show()

```
std::string Seize::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.128.4 Member Data Documentation

3.128.4.1 DEFAULT `const struct Seize::DEFAULT_VALUES Seize::DEFAULT`

3.129 Separate Class Reference

```
#include <Separate.h>
```

Inheritance diagram for Separate:

Public Member Functions

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

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.129.1 Detailed Description

Separate module DESCRIPTION This module can be used to either copy an incoming entity into multiple entities or to split a previously batched entity. Rules for allocating costs and times to the duplicate are also specified. Rules for attribute assignment to member entities are specified as well. When splitting existing batches, the temporary representative entity that was formed is disposed and the original entities that formed the group are recovered. The entities proceed sequentially from the module in the same order in which they originally were added to the batch. When duplicating entities, the specified number of copies is made and sent from the module. The original incoming entity also leaves the module. TYPICAL USES Send individual entities to represent boxes removed from a container Send an order both to fulfillment and billing for parallel processing Separate a previously batched set of documents PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of separating the incoming entity. Duplicate Original will simply take the original entity and make some number of identical duplicates. Split Existing Batch requires that the incoming entity be a temporarily batched entity using the Batch module. The original entities from the batch will be split. Percent Cost to Duplicates Allocation of costs and times of the incoming entity to the outgoing duplicates. This value is specified as a percentage of the original entity's costs and times (between 0-100). The percentage specified will be split evenly between the duplicates, while the original entity will retain any remaining cost/time percentage. Visible only when Type is Duplicate Original.

3.129.2 of Duplicates Number of outgoing entities that will leave the module, in

addition to the original incoming entity. Applies only when Type is Duplicate Original. Member Attributes Method of determining how to assign the representative entity attribute values to the original entities.These options relate to six of the special-purpose attributes (Entity.Type, Entity.Picture, Entity.Sequence, Entity.Station, Entity.Jobstep, and Entity.HoldCostRate) and all user-defined attributes. Applies only when Type is Split Existing Batch. Attribute Name Name of representative entity attribute(s) that are assigned to original entities of the group. Applies only when Member Attributes is Take Specific Representative Values.

3.129.3 Constructor & Destructor Documentation

3.129.3.1 Separate()

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

3.129.3.2 ~Separate()

```
virtual Separate::~Separate ( ) [virtual], [default]
```

3.129.4 Member Function Documentation

3.129.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.129.4.2 _loadInstance()

```
bool Separate::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.129.4.3 _onDispatchEvent()

```
void Separate::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.129.4.4 _saveInstance()

```
void Separate::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.129.4.5 GetPluginInformation() `PluginInformation * Separate::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.129.4.6 LoadInstance() `ModelComponent * Separate::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.129.4.7 NewInstance() `ModelDataDefinition * Separate::NewInstance ( Model * model, std::string name = "" ) [static]`

Here is the call graph for this function:

3.129.4.8 show() `std::string Separate::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.130 Sequence Class Reference

```
#include <Sequence.h>
```

Inheritance 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 ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.130.1 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

3.130.2 Constructor & Destructor Documentation

3.130.2.1 Sequence()

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

3.130.2.2 ~Sequence()

```
virtual Sequence::~Sequence ( ) [virtual], [default]
```

3.130.3 Member Function Documentation

3.130.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.130.3.2 _loadInstance()

```
bool Sequence::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.130.3.3 _saveInstance()

```
void Sequence::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.130.3.4 GetPluginInformation() `PluginInformation * Sequence::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.130.3.5 getSteps() `List< SequenceStep * > * Sequence::getSteps ( ) const`

3.130.3.6 LoadInstance() `ModelDataDefinition * Sequence::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.130.3.7 NewInstance()

```
ModelDataDefinition * Sequence::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.130.3.8 show()

```
std::string Sequence::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.131 SequenceStep Class Reference

```
#include <Sequence.h>
```

Inheritance diagram for SequenceStep:

Public Member Functions

- SequenceStep (Station ∗station, std::list< Assignment ∗ > ∗assignments=nullptr)
- SequenceStep (Label ∗label, std::list< Assignment ∗ > ∗assignments=nullptr)
- SequenceStep (Model ∗model, std::string stationOrLabelName, bool isStation=true, std::list< Assignment ∗ > ∗assignments=nullptr)
- virtual bool _loadInstance (PersistenceRecord ∗fields, unsigned int parentIndex)
- virtual void _saveInstance (PersistenceRecord ∗fields, unsigned int parentIndex, bool saveDefaultValues)
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  Loads object fields from a deserialization structure.
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- std::list< Assignment ∗ > ∗ getAssignments () const
- void setStation (Station ∗_station)
- Station ∗ getStation () const
- void setElementManager (ModelDataManager ∗_modeldataManager)
- void setLabel (Label ∗_label)
- Label ∗ getLabel () const

Additional Inherited Members

3.131.1 Constructor & Destructor Documentation

3.131.1.1 SequenceStep() [1/3]

```
SequenceStep::SequenceStep (
    Station * station,
    std::list< Assignment * > * assignments = nullptr )
```

3.131.1.2 SequenceStep() [2/3]

```
SequenceStep::SequenceStep (
    Label * label,
    std::list< Assignment * > * assignments = nullptr )
```

3.131.1.3 SequenceStep() [3/3]

```
SequenceStep::SequenceStep (
    Model * model,
    std::string stationOrLabelName,
    bool isStation = true,
    std::list< Assignment * > * assignments = nullptr )
```

Here is the call graph for this function:

3.131.2 Member Function Documentation

3.131.2.1 _loadInstance() [1/2]

```
bool SequenceStep::_loadInstance (
    PersistenceRecord * fields ) [virtual]
```

Loads object fields from a deserialization structure.

Parameters

fields	a table-like structure which can be queried for parsed fields.

Returns

a boolean which is true iff the load operation was successful.

Implements PersistentObject_base.

3.131.2.2 _loadInstance() [2/2]

```
bool SequenceStep::_loadInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex ) [virtual]
```

Here is the call graph for this function:

3.131.2.3 _saveInstance() [1/2]

```
void SequenceStep::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Implements PersistentObject_base.

3.131.2.4 _saveInstance() [2/2]

```
void SequenceStep::_saveInstance (
    PersistenceRecord * fields,
    unsigned int parentIndex,
    bool saveDefaultValues ) [virtual]
```

Here is the call graph for this function:

3.131.2.5 getAssignments()

```
std::list< Assignment * > * SequenceStep::getAssignments ( ) const
```

3.131.2.6 getLabel()

```
Label * SequenceStep::getLabel ( ) const
```

3.131.2.7 getStation()

```
Station * SequenceStep::getStation ( ) const
```

3.131.2.8 setElementManager()

```
void SequenceStep::setElementManager (
    ModelDataManager * _modeldataManager )
```

3.131.2.9 setLabel()

```
void SequenceStep::setLabel (
    Label * _label )
```

3.131.2.10 setStation()

```
void SequenceStep::setStation (
    Station * _station )
```

3.132 Set Class Reference

```
#include <Set.h>
```

Inheritance 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< ModelDataDefinition * > * getElementSet () const

Static Public Member Functions

- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation * _getParserChangesInformation ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.132.1 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.

3.132.2 Constructor & Destructor Documentation

3.132.2.1 Set()

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

3.132.2.2 ~Set() `virtual Set::~Set ( ) [virtual], [default]`

3.132.3 Member Function Documentation

3.132.3.1 _check() `bool Set::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.132.3.2 _createInternalAndAttachedData() `void Set::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.132.3.3 _getParserChangesInformation()

```
ParserChangesInformation * Set::_getParserChanges↩
Information ( ) [protected], [virtual]
```

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.132.3.4 _loadInstance()

```
bool Set::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.132.3.5 _saveInstance()

```
void Set::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.132.3.6 getElementSet() `List< ModelDataDefinition * > * Set::getElementSet ( ) const`

3.132.3.7 GetPluginInformation() `PluginInformation * Set::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.132.3.8 getSetOfType() `std::string Set::getSetOfType ( ) const`

3.132.3.9 LoadInstance() `ModelDataDefinition * Set::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.132.3.10 NewInstance()

```
ModelDataDefinition * Set::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.132.3.11 setSetOfType()

```
void Set::setSetOfType (
    std::string _setOfType )
```

3.132.3.12 show()

```
std::string Set::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.133 Signal Class Reference

```
#include <Signal.h>
```

Inheritance diagram for Signal:

Public Member Functions

- Signal (Model ∗model, std::string name="")
- virtual ∼Signal ()=default
- virtual std::string show ()

  virtual public methods
- void setSignalData (SignalData ∗signal)

- const std::string & limitExpression () const
- void setLimitExpression (const std::string &newLimitExpression)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.133.1 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 PROMPTS 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.

3.133.2 Constructor & Destructor Documentation

3.133.2.1 Signal()

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

3.133.2.2 ~Signal()

```
virtual Signal::~Signal ( ) [virtual], [default]
```

3.133.3 Member Function Documentation

3.133.3.1 _check() `bool Signal::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.133.3.2 _createInternalAndAttachedData() `void Signal::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.133.3.3 _initBetweenReplications() `void Signal::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.133.3.4 _loadInstance()

```
bool Signal::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.133.3.5 _onDispatchEvent()

```
void Signal::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.133.3.6 _saveInstance()

```
void Signal::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.133.3.7 GetPluginInformation() `PluginInformation * Signal::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.133.3.8 limitExpression() `const std::string & Signal::limitExpression ( ) const`

3.133.3.9 LoadInstance() `ModelComponent * Signal::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.133.3.10 NewInstance()

```
ModelDataDefinition * Signal::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.133.3.11 setLimitExpression()

```
void Signal::setLimitExpression (
    const std::string & newLimitExpression )
```

3.133.3.12 setSignalData()

```
void Signal::setSignalData (
    SignalData * signal )
```

3.133.3.13 show()

```
std::string Signal::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.134 SignalData Class Reference

```
#include <SignalData.h>
```

Inheritance diagram for SignalData:

Public Types

- typedef std::function< unsigned int(SignalData *) > SignalDataEventHandler
- typedef std::pair< SignalDataEventHandler, ModelComponent * > PairSignalDataEventHandler

Public Member Functions

- SignalData (Model *model, std::string name="")
- virtual ~SignalData ()=default
- virtual std::string show ()
- unsigned int generateSignal (double signalValue, unsigned int limit)
- void addSignalDataEventHandler (SignalDataEventHandler eventHandler, ModelComponent *component)
- unsigned int remainsToLimit () const
- void decreaseRemainLimit ()

Static Public Member Functions

- template<typename Class >
  static SignalDataEventHandler SetSignalDataEventHandler (unsigned int(Class::*function)(SignalData *), Class *object)
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()

Additional Inherited Members

3.134.1 Member Typedef Documentation

3.134.1.1 PairSignalDataEventHandler `typedef std::pair<SignalDataEventHandler, ModelComponent*> SignalData::PairSignalDataEventHandler`

3.134.1.2 SignalDataEventHandler `typedef std::function<unsigned int(SignalData*) > SignalData::SignalDataEven`

3.134.2 Constructor & Destructor Documentation

3.134.2.1 SignalData()

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

3.134.2.2 ~SignalData()

```
virtual SignalData::~SignalData ( ) [virtual], [default]
```

3.134.3 Member Function Documentation

3.134.3.1 _check()

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

could be overriden by derived classes

@TODO

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.134.3.2 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

3.134.3.3 _loadInstance()

```
bool SignalData::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.134.3.4 _saveInstance()

```
void SignalData::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.134.3.5 addSignalDataEventHandler()

```
void SignalData::addSignalDataEventHandler (
    SignalDataEventHandler eventHandler,
    ModelComponent * component )
```

3.134.3.6 decreaseRemainLimit()

```
void SignalData::decreaseRemainLimit ( )
```

3.134.3.7 generateSignal()

```
unsigned int SignalData::generateSignal (
    double signalValue,
    unsigned int limit )
```

3.134.3.8 GetPluginInformation()

```
PluginInformation * SignalData::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.134.3.9 LoadInstance()

```
ModelDataDefinition * SignalData::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.134.3.10 NewInstance()

```
ModelDataDefinition * SignalData::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.134.3.11 remainsToLimit()

```
unsigned int SignalData::remainsToLimit ( ) const
```

3.134.3.12 SetSignalDataEventHandler()

```
template<typename Class >
static SignalDataEventHandler SignalData::SetSignalDataEventHandler (
    unsigned int(Class::*)(SignalData *) function,
    Class * object ) [inline], [static]
```

3.134.3.13 show()

```
std::string SignalData::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.135 SimulationControl Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControl:

Public Member Functions

- SimulationControl (std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- std::string show () const
- bool isReadOnly () const
- virtual std::string getValue () const =0
- virtual void setValue (std::string value)=0

Protected Attributes

- bool _readonly

3.135.1 Constructor & Destructor Documentation

3.135.1.1 SimulationControl()

```
SimulationControl::SimulationControl (
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" )  [inline]
```

3.135.2 Member Function Documentation

3.135.2.1 getValue() `virtual std::string SimulationControl::getValue ( ) const [pure virtual]`

Implemented in SimulationControlString, SimulationControlDouble, SimulationControlBool, SimulationControlUInt, SimulationControlUShort, SimulationControlInt, and SimulationControlTimeUnit.

3.135.2.2 isReadOnly() `bool SimulationControl::isReadOnly ( ) const [inline]`

3.135.2.3 setValue() `virtual void SimulationControl::setValue ( std::string value ) [pure virtual]`

Implemented in SimulationControlString, SimulationControlDouble, SimulationControlBool, SimulationControlUInt, SimulationControlUShort, SimulationControlInt, and SimulationControlTimeUnit.

3.135.2.4 show() `std::string SimulationControl::show ( ) const [inline]`

Here is the call graph for this function:

3.135.3 Member Data Documentation

3.135.3.1 _readonly `bool SimulationControl::_readonly [protected]`

3.136 SimulationControlBool Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlBool:

Public Member Functions

- SimulationControlBool (GetterBool getter, SetterBool setter, std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")

- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.136.1 Constructor & Destructor Documentation

3.136.1.1 SimulationControlBool()

```
SimulationControlBool::SimulationControlBool (
    GetterBool getter,
    SetterBool setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.136.2 Member Function Documentation

3.136.2.1 getValue()

```
virtual std::string SimulationControlBool::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.136.2.2 setValue()

```
virtual void SimulationControlBool::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.137 SimulationControlDouble Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlDouble:

Public Member Functions

- SimulationControlDouble (GetterDouble getter, SetterDouble setter, std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.137.1 Constructor & Destructor Documentation

3.137.1.1 SimulationControlDouble()

```
SimulationControlDouble::SimulationControlDouble (
    GetterDouble getter,
    SetterDouble setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.137.2 Member Function Documentation

3.137.2.1 getValue()

```
virtual std::string SimulationControlDouble::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.137.2.2 setValue()

```
virtual void SimulationControlDouble::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.138 SimulationControlInt Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlInt:

Public Member Functions

- SimulationControlInt (GetterInt getter, SetterInt setter, std::string className, std::string elementName, std↩ ::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.138.1 Constructor & Destructor Documentation

3.138.1.1 SimulationControlInt()

```
SimulationControlInt::SimulationControlInt (
    GetterInt getter,
    SetterInt setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.138.2 Member Function Documentation

3.138.2.1 getValue()

```
virtual std::string SimulationControlInt::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.138.2.2 setValue()

```
virtual void SimulationControlInt::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.139 SimulationControlString Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlString:

Public Member Functions

- SimulationControlString (GetterString getter, SetterString setter, std::string className, std::string element↩ Name, std::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.139.1 Constructor & Destructor Documentation

3.139.1.1 SimulationControlString()

```
SimulationControlString::SimulationControlString (
    GetterString getter,
    SetterString setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.139.2 Member Function Documentation

3.139.2.1 getValue()

```
virtual std::string SimulationControlString::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.139.2.2 setValue()

```
virtual void SimulationControlString::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.140 SimulationControlTimeUnit Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlTimeUnit:

Public Member Functions

- SimulationControlTimeUnit (GetterTimeUnit getter, SetterTimeUnit setter, std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.140.1 Constructor & Destructor Documentation

3.140.1.1 SimulationControlTimeUnit()

```
SimulationControlTimeUnit::SimulationControlTimeUnit (
    GetterTimeUnit getter,
    SetterTimeUnit setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.140.2 Member Function Documentation

3.140.2.1 getValue()

```
virtual std::string SimulationControlTimeUnit::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.140.2.2 setValue()

```
virtual void SimulationControlTimeUnit::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.141 SimulationControlUInt Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlUInt:

Public Member Functions

- SimulationControlUInt (GetterUInt getter, SetterUInt setter, std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.141.1 Constructor & Destructor Documentation

3.141.1.1 SimulationControlUInt()

```
SimulationControlUInt::SimulationControlUInt (
    GetterUInt getter,
    SetterUInt setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.141.2 Member Function Documentation

3.141.2.1 getValue()

```
virtual std::string SimulationControlUInt::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.141.2.2 setValue()

```
virtual void SimulationControlUInt::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.142 SimulationControlUShort Class Reference

```
#include <SimulationControlAndResponse.h>
```

Inheritance diagram for SimulationControlUShort:

Public Member Functions

- SimulationControlUShort (GetterUShort getter, SetterUShort setter, std::string className, std::string elementName, std::string propertyName, std::string whatsThis="")
- virtual std::string getValue () const override
- virtual void setValue (std::string value) override

Additional Inherited Members

3.142.1 Constructor & Destructor Documentation

3.142.1.1 SimulationControlUShort()

```
SimulationControlUShort::SimulationControlUShort (
    GetterUShort getter,
    SetterUShort setter,
    std::string className,
    std::string elementName,
    std::string propertyName,
    std::string whatsThis = "" ) [inline]
```

3.142.2 Member Function Documentation

3.142.2.1 getValue()

```
virtual std::string SimulationControlUShort::getValue ( ) const [inline], [override], [virtual]
```

Implements SimulationControl.

3.142.2.2 setValue()

```
virtual void SimulationControlUShort::setValue (
    std::string value ) [inline], [override], [virtual]
```

Implements SimulationControl.

3.143 SimulationEvent Class Reference

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

Public Member Functions

- void setSimulatedTime (double simulatedTime)
- double getSimulatedTime () const
- void setStopRequested (bool stopRequested)
- bool isStopRequested () const
- void setPauseRequested (bool pauseRequested)
- bool isPauseRequested () const
- void setPaused (bool Paused)
- bool isPaused () const
- void setRunning (bool Running)
- bool isRunning () const
- void setCustomObject (void *customObject)
- void * getCustomObject () const
- void setCurrentReplicationNumber (unsigned int currentReplicationNumber)
- unsigned int getCurrentReplicationNumber () const
- void setCurrentEvent (Event *currentEvent)
- Event * getCurrentEvent () const
- void setEntityCreated (Entity *entityCreated)
- Entity * getEntityCreated () const
- void setDestinationComponent (ModelComponent *destinationComponent)
- ModelComponent * getDestinationComponent () const
- void setEntityMoveTimeDelay (double entityMoveTimeDelay)
- double getEntityMoveTimeDelay () const

Friends

- class ModelSimulation

3.143.1 Detailed Description

Stores an event that happened on a specific replication

3.143.2 Member Function Documentation

3.143.2.1 getCurrentEvent() `Event * SimulationEvent::getCurrentEvent ( ) const [inline]`

3.143.2.2 getCurrentReplicationNumber() `unsigned int SimulationEvent::getCurrentReplication↩ Number ( ) const [inline]`

3.143.2.3 getCustomObject() `void * SimulationEvent::getCustomObject ( ) const [inline]`

3.143.2.4 getDestinationComponent() `ModelComponent * SimulationEvent::getDestinationComponent ( ) const [inline]`

3.143.2.5 getEntityCreated() `Entity * SimulationEvent::getEntityCreated ( ) const [inline]`

3.143.2.6 getEntityMoveTimeDelay() `double SimulationEvent::getEntityMoveTimeDelay ( ) const [inline]`

3.143.2.7 getSimulatedTime() `double SimulationEvent::getSimulatedTime ( ) const [inline]`

3.143.2.8 isPaused() `bool SimulationEvent::isPaused ( ) const [inline]`

3.143.2.9 isPauseRequested() `bool SimulationEvent::isPauseRequested ( ) const [inline]`

3.143.2.10 isRunning() `bool SimulationEvent::isRunning ( ) const [inline]`

3.143.2.11 isStopRequested() `bool SimulationEvent::isStopRequested ( ) const [inline]`

3.143.2.12 setCurrentEvent() `void SimulationEvent::setCurrentEvent ( Event * currentEvent ) [inline]`

3.143.2.13 setCurrentReplicationNumber() `void SimulationEvent::setCurrentReplicationNumber ( unsigned int currentReplicationNumber ) [inline]`

3.143.2.14 setCustomObject() `void SimulationEvent::setCustomObject ( void * customObject ) [inline]`

3.143.2.15 setDestinationComponent() `void SimulationEvent::setDestinationComponent ( ModelComponent * destinationComponent ) [inline]`

3.143.2.16 setEntityCreated() `void SimulationEvent::setEntityCreated ( Entity * entityCreated ) [inline]`

3.143.2.17 setEntityMoveTimeDelay() `void SimulationEvent::setEntityMoveTimeDelay ( double entityMoveTimeDelay ) [inline]`

3.143.2.18 setPaused() `void SimulationEvent::setPaused ( bool Paused ) [inline]`

3.143.2.19 setPauseRequested() `void SimulationEvent::setPauseRequested ( bool pauseRequested ) [inline]`

3.143.2.20 setRunning() `void SimulationEvent::setRunning ( bool Running ) [inline]`

3.143.2.21 setSimulatedTime() `void SimulationEvent::setSimulatedTime ( double simulatedTime ) [inline]`

3.143.2.22 setStopRequested() `void SimulationEvent::setStopRequested ( bool stopRequested ) [inline]`

3.143.3 Friends And Related Function Documentation

3.143.3.1 ModelSimulation `friend class ModelSimulation [friend]`

3.144 SimulationExperiment Class Reference

```
#include <SimulationExperiment.h>
```

Public Member Functions

- SimulationExperiment ()
- virtual ~SimulationExperiment ()=default

3.144.1 Constructor & Destructor Documentation

3.144.1.1 SimulationExperiment() `SimulationExperiment::SimulationExperiment ( )`

3.144.1.2 ~SimulationExperiment() `virtual SimulationExperiment::~SimulationExperiment ( ) [virtual], [default]`

3.145 SimulationReporter_if Class Reference

`#include <SimulationReporter_if.h>`

Inheritance diagram for SimulationReporter_if:

Public Member Functions

- virtual void showReplicationStatistics ()=0
- virtual void showSimulationStatistics ()=0
- virtual void showSimulationResponses ()=0
- virtual void showSimulationControls ()=0

3.145.1 Member Function Documentation

3.145.1.1 showReplicationStatistics() `virtual void SimulationReporter_if::showReplicationStatistics ( ) [pure virtual]`

Implemented in SimulationReporterDefaultImpl1.

3.145.1.2 showSimulationControls() `virtual void SimulationReporter_if::showSimulationControls ( ) [pure virtual]`

Implemented in SimulationReporterDefaultImpl1.

3.145.1.3 showSimulationResponses() `virtual void SimulationReporter_if::showSimulation↩ Responses ( ) [pure virtual]`

Implemented in SimulationReporterDefaultImpl1.

3.145.1.4 showSimulationStatistics() `virtual void SimulationReporter_if::showSimulationStatistics ( ) [pure virtual]`

Implemented in SimulationReporterDefaultImpl1.

3.146 SimulationReporterDefaultImpl1 Class Reference

```
#include <SimulationReporterDefaultImpl1.h>
```

Inheritance diagram for SimulationReporterDefaultImpl1:

Public Member Functions

- SimulationReporterDefaultImpl1 (ModelSimulation *simulation, Model *model, List< ModelDataDefinition * > *statsCountersSimulation)
- virtual ~SimulationReporterDefaultImpl1 ()=default
- virtual void showReplicationStatistics ()
- virtual void showSimulationStatistics ()
- virtual void showSimulationResponses ()
- virtual void showSimulationControls ()

3.146.1 Detailed Description

Class that implements SimulationReporter_if interface and is responsible for building and showing replication and simulation reports

3.146.2 Constructor & Destructor Documentation

3.146.2.1 SimulationReporterDefaultImpl1()

```
SimulationReporterDefaultImpl1::SimulationReporter↩
DefaultImpl1 (
            ModelSimulation * simulation,
            Model * model,
            List< ModelDataDefinition * > * statsCountersSimulation )
```

3.146.2.2 ~SimulationReporterDefaultImpl1()

```
virtual SimulationReporterDefaultImpl1::~Simulation↩
ReporterDefaultImpl1 ( ) [virtual], [default]
```

3.146.3 Member Function Documentation

3.146.3.1 showReplicationStatistics()

```
void SimulationReporterDefaultImpl1::showReplication↩
Statistics ( ) [virtual]
```

Implements SimulationReporter_if.

Here is the call graph for this function:

3.146.3.2 showSimulationControls() `void SimulationReporterDefaultImpl1::showSimulationControls ( ) [virtual]`

Implements SimulationReporter_if.

Here is the call graph for this function:

3.146.3.3 showSimulationResponses() `void SimulationReporterDefaultImpl1::showSimulation↩ Responses ( ) [virtual]`

Implements SimulationReporter_if.

Here is the call graph for this function:

3.146.3.4 showSimulationStatistics() `void SimulationReporterDefaultImpl1::showSimulationStatistics ( ) [virtual]`

@TODO: USE REFERENCE TO MAPITEM TO AVOID COPY

Implements SimulationReporter_if.

Here is the call graph for this function:

3.147 SimulationScenario Class Reference

```
#include <SimulationScenario.h>
```

Public Member Functions

- SimulationScenario ()
- virtual ~SimulationScenario ()=default
- bool startSimulation (Simulator *sim, std::string *errorMessage)
- std::list< std::pair< std::string, double > * > * getResponseValues () const
- std::list< std::pair< std::string, double > * > * getControlValues () const
- double getResponseValue (const 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::string > * getSelectedControls () const
- double getControlValue (const std::string &controlName)
- std::list< std::string > * getSelectedResponses () const
- void setSelectedControls (std::list< std::string > *selectedControls)
- void setControl (std::string name, double value) const

3.147.1 Detailed Description

Represents a scenario where a specific model (defined my ModelFilename) will be simulated. To each scenario will be associated a set of SimulationControl and SimulationResponse, and their values are set to the scenario by the ProcessAnalyser.

3.147.2 Constructor & Destructor Documentation

3.147.2.1 SimulationScenario() `SimulationScenario::SimulationScenario ( ) [default]`

3.147.2.2 ~SimulationScenario() `virtual SimulationScenario::~SimulationScenario ( ) [virtual], [default]`

3.147.3 Member Function Documentation

3.147.3.1 getControlValue() `double SimulationScenario::getControlValue ( const std::string & controlName )`

3.147.3.2 getControlValues() `std::list< std::pair< std::string, double > * > * SimulationScenario::getControlValues ( ) const`

3.147.3.3 getModelFilename() `std::string SimulationScenario::getModelFilename ( ) const`

3.147.3.4 getResponseValue()

```
double SimulationScenario::getResponseValue (
    const std::string & responseName )
```

3.147.3.5 getResponseValues()

```
std::list< std::pair< std::string, double > * > * SimulationScenario::getResponseValues ( ) const
```

The final result of the simulationScenario

3.147.3.6 getScenarioDescription()

```
std::string SimulationScenario::getScenarioDescription ( ) const
```

3.147.3.7 getScenarioName()

```
std::string SimulationScenario::getScenarioName ( ) const
```

3.147.3.8 getSelectedControls()

```
std::list< std::string > * SimulationScenario::getSelectedControls ( ) const
```

3.147.3.9 getSelectedResponses()

```
std::list< std::string > * SimulationScenario::getSelectedResponses ( ) const
```

3.147.3.10 setControl()

```
void SimulationScenario::setControl (
    std::string name,
    double value ) const
```

3.147.3.11 setModelFilename()

```
void SimulationScenario::setModelFilename (
    std::string _modelFilename )
```

3.147.3.12 setScenarioDescription()

```
void SimulationScenario::setScenarioDescription (
    std::string _scenarioDescription )
```

3.147.3.13 setScenarioName()

```
void SimulationScenario::setScenarioName (
    std::string _name )
```

3.147.3.14 setSelectedControls()

```
void SimulationScenario::setSelectedControls (
    std::list< std::string > * selectedControls )
```

3.147.3.15 startSimulation()

```
bool SimulationScenario::startSimulation (
    Simulator * sim,
    std::string * errorMessage )
```

3.148 Simulator Class Reference

```
#include <Simulator.h>
```

Public Member Functions

- Simulator ()
- virtual ~Simulator ()=default
- std::string getVersion () const
- unsigned int getVersionNumber () const
- std::string getName () const
- LicenceManager * getLicence () const
- PluginManager * getPlugins () const
- ModelManager * getModels () const
- TraceManager * getTracer () const
- ParserManager * getParser () const
- ExperimentManager * getExperimenter () const

Friends

- class PluginManager

3.148.1 Detailed Description

The main class of the Genesys KERNEL simulation. It gives access to simulation models and tools. Simulation is the top level class and is supossed to be available to application as a dynamic linked library.

3.148.2 Constructor & Destructor Documentation

3.148.2.1 Simulator() `Simulator::Simulator ( )`

Simulator constructor basically constructs inner classes (managers) and shows basic information on the console. This is the only method on the entire simulator thats directlly plots something to the output Here is the call graph for this function:

3.148.2.2 ~Simulator() `virtual Simulator::~Simulator ( ) [virtual], [default]`

3.148.3 Member Function Documentation

3.148.3.1 getExperimenter() `ExperimentManager * Simulator::getExperimenter ( ) const`

3.148.3.2 getLicence() `LicenceManager * Simulator::getLicence ( ) const`

3.148.3.3 getModels() `ModelManager * Simulator::getModels ( ) const`

3.148.3.4 getName() `std::string Simulator::getName ( ) const`

3.148.3.5 getParser() `ParserManager * Simulator::getParser ( ) const`

3.148.3.6 getPlugins() `PluginManager * Simulator::getPlugins ( ) const`

3.148.3.7 getTracer() `TraceManager * Simulator::getTracer ( ) const`

3.148.3.8 getVersion() `std::string Simulator::getVersion ( ) const`

3.148.3.9 getVersionNumber() `unsigned int Simulator::getVersionNumber ( ) const`

3.148.4 Friends And Related Function Documentation

3.148.4.1 PluginManager `friend class PluginManager [friend]`

3.149 SinkModelComponent Class Reference

```
#include <SinkModelComponent.h>
```

Inheritance diagram for SinkModelComponent:

Public Member Functions

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

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

virtual protected method that must be overriden

- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.

- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.149.1 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.

3.149.2 Constructor & Destructor Documentation

3.149.2.1 SinkModelComponent()

```
SinkModelComponent::SinkModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "" )
```

3.149.2.2 ~SinkModelComponent()

```
virtual SinkModelComponent::~SinkModelComponent ( ) [virtual], [default]
```

3.149.3 Member Function Documentation

3.149.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Reimplemented in Dispose.

3.149.3.2 _loadInstance()

```
bool SinkModelComponent::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Reimplemented in Dispose.

Here is the call graph for this function:

3.149.3.3 _saveInstance()

```
void SinkModelComponent::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Reimplemented in Dispose.

Here is the call graph for this function:

3.150 Solver_if Class Reference

```
#include <Solver_if.h>
```

Inheritance diagram for Solver_if:

Public Member Functions

- virtual void setPrecision (double e)=0
- virtual double getPrecision ()=0
- virtual void setMaxSteps (double steps)=0
- virtual double getMaxSteps ()=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
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double), double p2)=0
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double), double p2, double p3)=0
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double, double), double p2, double p3, double p4)=0
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double, double, double), double p2, double p3, double p4, double p5)=0

3.150.1 Detailed Description

Interface used by classes that perform the numerical integration and derivation of functions with from one up to four parameters. It is mainly used for calculating the probability of theoretical distributions, from its probability distribution functions. p1 is the value where function is being evaluated and p2, ... are the function parameters

3.150.2 Member Function Documentation

3.150.2.1 derivate() [1/4]

```
virtual double Solver_if::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double) f,
    double p2 )  [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.2 derivate() [2/4]

```
virtual double Solver_if::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double) f,
    double p2,
    double p3 )  [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.3 derivate() [3/4]

```
virtual double Solver_if::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 )  [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.4 derivate() [4/4]

```
virtual double Solver_if::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double, double, double) f,
    double p2,
    double p3,
    double p4,
    double p5 ) [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.5 getMaxSteps()

```
virtual double Solver_if::getMaxSteps ( ) [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.6 getPrecision()

```
virtual double Solver_if::getPrecision ( ) [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.7 integrate() [1/4]

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

Implemented in SolverDefaultImpl1.

3.150.2.8 integrate() [2/4]

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

Implemented in SolverDefaultImpl1.

3.150.2.9 integrate() [3/4]

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

Implemented in SolverDefaultImpl1.

3.150.2.10 integrate() [4/4]

```
virtual double Solver_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 SolverDefaultImpl1.

3.150.2.11 setMaxSteps()

```
virtual void Solver_if::setMaxSteps (
    double steps )  [pure virtual]
```

Implemented in SolverDefaultImpl1.

3.150.2.12 setPrecision()

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

Implemented in SolverDefaultImpl1.

3.151 SolverDefaultImpl1 Class Reference

```
#include <SolverDefaultImpl1.h>
```

Inheritance diagram for SolverDefaultImpl1:

Public Member Functions

- SolverDefaultImpl1 (double precision=1e-6, unsigned int steps=1e3)
- virtual ~SolverDefaultImpl1 ()=default
- virtual void setPrecision (double e)
- virtual double getPrecision ()
- virtual void setMaxSteps (double steps)

- virtual double getMaxSteps ()
- 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)
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double), double p2)
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double), double p2, double p3)
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double, double), double p2, double p3, double p4)
- virtual double derivate (double initPoint, double initValue, double(∗f)(double, double, double, double, double), double p2, double p3, double p4, double p5)

3.151.1 Constructor & Destructor Documentation

3.151.1.1 SolverDefaultImpl1()

```
SolverDefaultImpl1::SolverDefaultImpl1 (
    double precision = 1e-6,
    unsigned int steps = 1e3 )
```

3.151.1.2 ∼SolverDefaultImpl1()

```
virtual SolverDefaultImpl1::~SolverDefaultImpl1 ( ) [virtual], [default]
```

3.151.2 Member Function Documentation

3.151.2.1 derivate() [1/4]

```
double SolverDefaultImpl1::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double) f,
    double p2 ) [virtual]
```

Implements Solver_if.

3.151.2.2 derivate() [2/4]

```
double SolverDefaultImpl1::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double) f,
    double p2,
    double p3 ) [virtual]
```

Implements Solver_if.

3.151.2.3 derivate() [3/4]

```
double SolverDefaultImpl1::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 ) [virtual]
```

Implements Solver_if.

3.151.2.4 derivate() [4/4]

```
double SolverDefaultImpl1::derivate (
    double initPoint,
    double initValue,
    double(*)(double, double, double, double, double) f,
    double p2,
    double p3,
    double p4,
    double p5 ) [virtual]
```

Implements Solver_if.

3.151.2.5 getMaxSteps()

```
double SolverDefaultImpl1::getMaxSteps ( ) [virtual]
```

Implements Solver_if.

3.151.2.6 getPrecision()

```
double SolverDefaultImpl1::getPrecision ( ) [virtual]
```

Implements Solver_if.

3.151.2.7 integrate() [1/4]

```
double SolverDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double) f,
    double p2 ) [virtual]
```

Implements Solver_if.

3.151.2.8 integrate() [2/4]

```
double SolverDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double) f,
    double p2,
    double p3 ) [virtual]
```

Implements Solver_if.

3.151.2.9 integrate() [3/4]

```
double SolverDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 ) [virtual]
```

Implements Solver_if.

3.151.2.10 integrate() [4/4]

```
double SolverDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double, double, double) f,
    double p2,
    double p3,
    double p4,
    double p5 ) [virtual]
```

Implements Solver_if.

3.151.2.11 setMaxSteps()

```
void SolverDefaultImpl1::setMaxSteps (
    double steps ) [virtual]
```

Implements Solver_if.

3.151.2.12 setPrecision()

```
void SolverDefaultImpl1::setPrecision (
    double e ) [virtual]
```

Implements Solver_if.

3.152 SortFile Class Reference

```
#include <SorttFile.h>
```

Public Member Functions

- SortFile ()
- ~SortFile ()=default
- bool sort ()
- void setDataFilename (std::string filename)

3.152.1 Constructor & Destructor Documentation

3.152.1.1 SortFile() `SortFile::SortFile ( )`

3.152.1.2 ~SortFile() `SortFile::~SortFile ( ) [default]`

3.152.2 Member Function Documentation

3.152.2.1 setDataFilename() `void SortFile::setDataFilename ( std::string filename )`

3.152.2.2 sort() `bool SortFile::sort ( )`

3.153 SourceModelComponent Class Reference

```
#include <SourceModelComponent.h>
```

Inheritance diagram for SourceModelComponent:

Classes

- struct DEFAULT_VALUES

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)
- void setEntityTypeName (std::string entityTypeName)
- EntityType * getEntityType () const
- void setTimeUnit (Util::TimeUnit _timeUnit)
- Util::TimeUnit getTimeUnit () const
- void setTimeBetweenCreationsExpression (std::string _timeBetweenCreations, Util::TimeUnit _time↩Unit=Util::TimeUnit::unknown)
- std::string getTimeBetweenCreationsExpression () const
- void setMaxCreations (unsigned long _maxCreations)
- 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 ()

  virtual public methods

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _initBetweenReplications ()
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Protected Attributes

- EntityType * _entityType = nullptr
- const struct SourceModelComponent::DEFAULT_VALUES DEFAULT
- double _firstCreation = DEFAULT.firstCreation
- unsigned int _entitiesPerCreation = DEFAULT.entitiesPerCreation
- std::string _maxCreationsExpression = DEFAULT.maxCreationsExpression
- std::string _timeBetweenCreationsExpression = DEFAULT.timeBetweenCreationsExpression
- Util::TimeUnit _timeBetweenCreationsTimeUnit = DEFAULT.timeBetweenCreationsTimeUnit
- unsigned int _entitiesCreatedSoFar = 0

Additional Inherited Members

3.153.1 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

3.153.2 Constructor & Destructor Documentation

3.153.2.1 SourceModelComponent()

```
SourceModelComponent::SourceModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "" )
```

Here is the call graph for this function:

3.153.2.2 ~SourceModelComponent()

```
virtual SourceModelComponent::~SourceModelComponent ( )
[virtual], [default]
```

3.153.3 Member Function Documentation

3.153.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Reimplemented in Create.

Here is the call graph for this function:

3.153.3.2 _createInternalAndAttachedData()

```
void SourceModelComponent::_createInternalAndAttached↩
Data ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Reimplemented in Create.

Here is the call graph for this function:

3.153.3.3 _initBetweenReplications() `void SourceModelComponent::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

Reimplemented in Create.

Here is the call graph for this function:

3.153.3.4 _loadInstance() `bool SourceModelComponent::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Reimplemented in Create.

Here is the call graph for this function:

3.153.3.5 _saveInstance() `void SourceModelComponent::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Reimplemented in Create.

Here is the call graph for this function:

3.153.3.6 getEntitiesCreated()

```
unsigned int SourceModelComponent::getEntitiesCreated ( ) const
```

3.153.3.7 getEntitiesPerCreation()

```
unsigned int SourceModelComponent::getEntitiesPerCreation ( ) const
```

3.153.3.8 getEntityType()

```
EntityType * SourceModelComponent::getEntityType ( ) const
```

3.153.3.9 getFirstCreation()

```
double SourceModelComponent::getFirstCreation ( ) const
```

3.153.3.10 getMaxCreations()

```
std::string SourceModelComponent::getMaxCreations ( ) const
```

3.153.3.11 getTimeBetweenCreationsExpression()

```
std::string SourceModelComponent::getTime↩
BetweenCreationsExpression ( ) const
```

3.153.3.12 getTimeUnit()

```
Util::TimeUnit SourceModelComponent::getTimeUnit ( ) const
```

3.153.3.13 setEntitiesCreated()

```
void SourceModelComponent::setEntitiesCreated (
    unsigned int _entitiesCreated )
```

3.153.3.14 setEntitiesPerCreation()

```
void SourceModelComponent::setEntitiesPerCreation (
    unsigned int _entitiesPerCreation )
```

3.153.3.15 setEntityType()

```
void SourceModelComponent::setEntityType (
    EntityType * _entityType )
```

3.153.3.16 setEntityTypeName()

```
void SourceModelComponent::setEntityTypeName (
    std::string entityTypeName )
```

Here is the call graph for this function:

3.153.3.17 setFirstCreation()

```
void SourceModelComponent::setFirstCreation (
    double _firstCreation )
```

3.153.3.18 setMaxCreations() [1/2]

```
void SourceModelComponent::setMaxCreations (
    std::string _maxCreationsExpression )
```

3.153.3.19 setMaxCreations() [2/2]

```
void SourceModelComponent::setMaxCreations (
    unsigned long _maxCreations )
```

3.153.3.20 setTimeBetweenCreationsExpression()

```
void SourceModelComponent::setTimeBetween↩
CreationsExpression (
    std::string _timeBetweenCreations,
    Util::TimeUnit _timeUnit = Util::TimeUnit::unknown )
```

3.153.3.21 setTimeUnit()

```
void SourceModelComponent::setTimeUnit (
    Util::TimeUnit _timeUnit )
```

3.153.3.22 show()

```
std::string SourceModelComponent::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Reimplemented in Create.

Here is the call graph for this function:

3.153.4 Member Data Documentation

3.153.4.1 _entitiesCreatedSoFar

```
unsigned int SourceModelComponent::_entitiesCreatedSoFar = 0 [protected]
```

3.153.4.2 _entitiesPerCreation

```
unsigned int SourceModelComponent::_entitiesPerCreation = DEFAULT.↩
entitiesPerCreation [protected]
```

3.153.4.3 _entityType

```
EntityType* SourceModelComponent::_entityType = nullptr [protected]
```

3.153.4.4 _firstCreation

```
double SourceModelComponent::_firstCreation = DEFAULT.firstCreation [protected]
```

3.153.4.5 _maxCreationsExpression

```
std::string SourceModelComponent::_maxCreationsExpression = DEFAULT.maxCreationsExpression [protected]
```

3.153.4.6 _timeBetweenCreationsExpression

`std::string SourceModelComponent::_timeBetween↩ CreationsExpression = DEFAULT.timeBetweenCreationsExpression [protected]`

3.153.4.7 _timeBetweenCreationsTimeUnit

`Util::TimeUnit SourceModelComponent::_timeBetween↩ CreationsTimeUnit = DEFAULT.timeBetweenCreationsTimeUnit [protected]`

3.153.4.8 DEFAULT

`const struct SourceModelComponent::DEFAULT_VALUES SourceModelComponent::↩ DEFAULT [protected]`

3.154 Start Class Reference

```
#include <Start.h>
```

Inheritance diagram for Start:

Public Member Functions

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

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.154.1 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.

3.154.2 Constructor & Destructor Documentation

3.154.2.1 Start()

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

3.154.2.2 ~Start()

```
virtual Start::~Start ( ) [virtual], [default]
```

3.154.3 Member Function Documentation

3.154.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.154.3.2 _loadInstance()

```
bool Start::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.154.3.3 _onDispatchEvent()

```
void Start::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.154.3.4 _saveInstance()

```
void Start::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.154.3.5 GetPluginInformation() `PluginInformation * Start::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.154.3.6 LoadInstance() `ModelComponent * Start::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.154.3.7 NewInstance()

```
ModelDataDefinition * Start::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.154.3.8 show()

```
std::string Start::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.155 Station Class Reference

```
#include <Station.h>
```

Inheritance 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 ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.155.1 Constructor & Destructor Documentation

3.155.1.1 Station()

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

3.155.1.2 ~Station()

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

3.155.2 Member Function Documentation

3.155.2.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.155.2.2 _createInternalAndAttachedData() `void Station::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.155.2.3 _loadInstance() `bool Station::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.155.2.4 _saveInstance() `void Station::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.155.2.5 enter()

```
void Station::enter (
    Entity * entity )
```

Here is the call graph for this function:

3.155.2.6 getEnterIntoStationComponent()

```
ModelComponent * Station::getEnterIntoStationComponent
( ) const
```

3.155.2.7 GetPluginInformation() `PluginInformation * Station::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.155.2.8 initBetweenReplications() `void Station::initBetweenReplications ( )`

Here is the call graph for this function:

3.155.2.9 leave() `void Station::leave ( Entity * entity )`

Here is the call graph for this function:

3.155.2.10 LoadInstance()

```
ModelDataDefinition * Station::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.155.2.11 NewInstance()

```
ModelDataDefinition * Station::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.155.2.12 setEnterIntoStationComponent()

```
void Station::setEnterIntoStationComponent ( ModelComponent * _enterIntoStationComponent )
```

3.155.2.13 show()

```
std::string Station::show ( ) [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.156 Statistics_if Class Reference

```
#include <Statistics_if.h>
```

Inheritance diagram for Statistics_if:

Public Member Functions

- virtual Collector_if ∗ getCollector () const =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 double confidenceLevel ()=0
- virtual unsigned int newSampleSize (double halfWidth)=0
- virtual void setConfidenceLevel (double confidencelevel)=0

3.156.1 Detailed Description

Interface for statisct 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.

3.156.2 Member Function Documentation

3.156.2.1 average() `virtual double Statistics_if::average ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.2 confidenceLevel() `virtual double Statistics_if::confidenceLevel ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.3 getCollector() `virtual Collector_if * Statistics_if::getCollector ( ) const [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.4 halfWidthConfidenceInterval() `virtual double Statistics_if::halfWidthConfidence↩ Interval ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.5 max() `virtual double Statistics_if::max ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.6 min() `virtual double Statistics_if::min ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.7 newSampleSize() `virtual unsigned int Statistics_if::newSampleSize ( double halfWidth ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.8 numElements() `virtual unsigned int Statistics_if::numElements ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.9 setCollector() `virtual void Statistics_if::setCollector ( Collector_if * collector ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.10 setConfidenceLevel() `virtual void Statistics_if::setConfidenceLevel ( double confidencelevel ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.11 stddeviation() `virtual double Statistics_if::stddeviation ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.12 variance() `virtual double Statistics_if::variance ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.156.2.13 variationCoef() `virtual double Statistics_if::variationCoef ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1, and StatisticsDefaultImpl1.

3.157 StatisticsCollector Class Reference

```
#include <StatisticsCollector.h>
```

Inheritance diagram for StatisticsCollector:

Public Member Functions

- StatisticsCollector (Model *model, std::string name="", ModelDataDefinition *parent=nullptr, bool insertInto↩ Model=true)

- virtual ~StatisticsCollector ()=default
- virtual std::string show ()
- ModelDataDefinition * getParent () const
- Statistics_if * getStatistics () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.157.1 Detailed Description

The StatisticsCollector is the ModelDataDefinition responsible for collecting data from the model (using the Collector) and simultaneouly keeping statistics updated (using the Statistics)

3.157.2 Constructor & Destructor Documentation

3.157.2.1 StatisticsCollector()

```
StatisticsCollector::StatisticsCollector (
    Model * model,
    std::string name = "",
    ModelDataDefinition * parent = nullptr,
    bool insertIntoModel = true )
```

Here is the call graph for this function:

3.157.2.2 ~StatisticsCollector() `virtual StatisticsCollector::~StatisticsCollector ( ) [virtual], [default]`

3.157.3 Member Function Documentation

3.157.3.1 _check() `bool StatisticsCollector::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.157.3.2 _loadInstance() `bool StatisticsCollector::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.157.3.3 _saveInstance() `void StatisticsCollector::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.157.3.4 getParent() `ModelDataDefinition * StatisticsCollector::getParent ( ) const`

3.157.3.5 GetPluginInformation() `PluginInformation * StatisticsCollector::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.157.3.6 getStatistics() `Statistics_if * StatisticsCollector::getStatistics ( ) const`

3.157.3.7 LoadInstance() `ModelDataDefinition * StatisticsCollector::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.157.3.8 NewInstance()

```
ModelDataDefinition * StatisticsCollector::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.157.3.9 show()

```
std::string StatisticsCollector::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.158 StatisticsDatafile_if Class Reference

```
#include <StatisticsDataFile_if.h>
```

Inheritance 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

3.158.1 Member Function Documentation

3.158.1.1 centil() `virtual double StatisticsDatafile_if::centil ( unsigned short num ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.2 decil() `virtual double StatisticsDatafile_if::decil ( unsigned short num ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.3 histogramClassFrequency() `virtual unsigned int StatisticsDatafile_if::histogram↩ ClassFrequency ( unsigned short classNum ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.4 histogramClassLowerLimit() `virtual double StatisticsDatafile_if::histogramClass↩ LowerLimit ( unsigned short classNum ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.5 histogramNumClasses() `virtual unsigned short StatisticsDatafile_if::histogramNum↩ Classes ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.6 mediane() `virtual double StatisticsDatafile_if::mediane ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.7 mode() `virtual double StatisticsDatafile_if::mode ( ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.8 quartil() `virtual double StatisticsDatafile_if::quartil ( unsigned short num ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.158.1.9 setHistogramNumClasses() `virtual void StatisticsDatafile_if::setHistogramNumClasses ( unsigned short num ) [pure virtual]`

Implemented in StatisticsDatafileDefaultImpl1.

3.159 StatisticsDatafileDefaultImpl1 Class Reference

```
#include <StatisticsDataFileDefaultImpl.h>
```

Inheritance diagram for StatisticsDatafileDefaultImpl1:

Public Member Functions

- StatisticsDatafileDefaultImpl1 ()
- virtual ~StatisticsDatafileDefaultImpl1 ()=default
- virtual Collector_if * getCollector () const override
- virtual void setCollector (Collector_if *collector) override
- virtual unsigned int numElements () override

- virtual double min () override
- virtual double max () override
- virtual double average () override
- virtual double variance () override
- virtual double stddeviation () override
- virtual double variationCoef () override
- virtual double halfWidthConfidenceInterval () override
- virtual double confidenceLevel () override
- virtual unsigned int newSampleSize (double halfWidth) override
- virtual void setConfidenceLevel (double confidencelevel) override
- virtual double mode () override
- virtual double mediane () override
- virtual double quartil (unsigned short num) override
- virtual double decil (unsigned short num) override
- virtual double centil (unsigned short num) override
- virtual void setHistogramNumClasses (unsigned short num) override
- virtual unsigned short histogramNumClasses () override
- virtual double histogramClassLowerLimit (unsigned short classNum) override
- virtual unsigned int histogramClassFrequency (unsigned short classNum) override

3.159.1 Constructor & Destructor Documentation

3.159.1.1 StatisticsDatafileDefaultImpl1() `StatisticsDatafileDefaultImpl1::StatisticsDatafile↩ DefaultImpl1 ( )`

3.159.1.2 ~StatisticsDatafileDefaultImpl1() `virtual StatisticsDatafileDefaultImpl1::~Statistics↩ DatafileDefaultImpl1 ( ) [virtual], [default]`

3.159.2 Member Function Documentation

3.159.2.1 average() `double StatisticsDatafileDefaultImpl1::average ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.2 centil() `double StatisticsDatafileDefaultImpl1::centil ( unsigned short num ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.3 confidenceLevel() `double StatisticsDatafileDefaultImpl1::confidenceLevel ( ) [override], [virtual]`

Implements Statistics_if.

3.159.2.4 decil() `double StatisticsDatafileDefaultImpl1::decil ( unsigned short num ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.5 getCollector() `Collector_if * StatisticsDatafileDefaultImpl1::getCollector ( ) const [override], [virtual]`

Implements Statistics_if.

3.159.2.6 halfWidthConfidenceInterval()

`double StatisticsDatafileDefaultImpl1::halfWidthConfidence↩ Interval ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.7 histogramClassFrequency()

`unsigned int StatisticsDatafileDefaultImpl1::histogram↩ ClassFrequency ( unsigned short *classNum* ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.8 histogramClassLowerLimit()

`double StatisticsDatafileDefaultImpl1::histogramClass↩ LowerLimit ( unsigned short *classNum* ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.9 histogramNumClasses() `unsigned short StatisticsDatafileDefaultImpl1::histogramNum↩ Classes ( ) [override], [virtual]`

Implements StatisticsDatafile_if.

3.159.2.10 max() `double StatisticsDatafileDefaultImpl1::max ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.11 mediane() `double StatisticsDatafileDefaultImpl1::mediane ( ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.12 min() `double StatisticsDatafileDefaultImpl1::min ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.13 mode() `double StatisticsDatafileDefaultImpl1::mode ( ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.14 newSampleSize() `unsigned int StatisticsDatafileDefaultImpl1::newSampleSize ( double halfWidth ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.15 numElements() `unsigned int StatisticsDatafileDefaultImpl1::numElements ( ) [override], [virtual]`

Implements Statistics_if.

3.159.2.16 quartil() `double StatisticsDatafileDefaultImpl1::quartil ( unsigned short num ) [override], [virtual]`

Implements StatisticsDatafile_if.

Here is the call graph for this function:

3.159.2.17 setCollector() `void StatisticsDatafileDefaultImpl1::setCollector ( Collector_if * collector ) [override], [virtual]`

Implements Statistics_if.

3.159.2.18 setConfidenceLevel() `void StatisticsDatafileDefaultImpl1::setConfidenceLevel ( double confidencelevel ) [override], [virtual]`

Implements Statistics_if.

3.159.2.19 setHistogramNumClasses() `void StatisticsDatafileDefaultImpl1::setHistogramNumClasses ( unsigned short num ) [override], [virtual]`

Implements StatisticsDatafile_if.

3.159.2.20 stddeviation() `double StatisticsDatafileDefaultImpl1::stddeviation ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.21 variance() `double StatisticsDatafileDefaultImpl1::variance ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.159.2.22 variationCoef() `double StatisticsDatafileDefaultImpl1::variationCoef ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.160 StatisticsDefaultImpl1 Class Reference

```
#include <StatisticsDefaultImpl1.h>
```

Inheritance 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 () const override
- virtual void setCollector (Collector_if *collector) override
- virtual unsigned int numElements () override
- virtual double min () override
- virtual double max () override
- virtual double average () override
- virtual double variance () override
- virtual double stddeviation () override
- virtual double variationCoef () override
- virtual double halfWidthConfidenceInterval () override
- virtual double confidenceLevel () override
- virtual unsigned int newSampleSize (double halfWidth) override
- virtual void setConfidenceLevel (double confidencelevel) override

3.160.1 Constructor & Destructor Documentation

3.160.1.1 StatisticsDefaultImpl1() [1/2] `StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( )`

When constructor is invoked without a Collector, it is taken from Traits<Statistics_if>::CollectorImplementation configuration.

Here is the call graph for this function:

3.160.1.2 StatisticsDefaultImpl1() [2/2] `StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( Collector_if * collector )`

Here is the call graph for this function:

3.160.1.3 ~StatisticsDefaultImpl1() `virtual StatisticsDefaultImpl1::~StatisticsDefaultImpl1 ( ) [virtual], [default]`

3.160.2 Member Function Documentation

3.160.2.1 average() `double StatisticsDefaultImpl1::average ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.2 confidenceLevel() `double StatisticsDefaultImpl1::confidenceLevel ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.3 getCollector() `Collector_if * StatisticsDefaultImpl1::getCollector ( ) const [override], [virtual]`

Implements Statistics_if.

3.160.2.4 halfWidthConfidenceInterval() `double StatisticsDefaultImpl1::halfWidthConfidence↩ Interval ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.5 max() `double StatisticsDefaultImpl1::max ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.6 min() `double StatisticsDefaultImpl1::min ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.7 newSampleSize()

`unsigned int StatisticsDefaultImpl1::newSampleSize ( double halfWidth ) [override], [virtual]`

Implements Statistics_if.

3.160.2.8 numElements()

`unsigned int StatisticsDefaultImpl1::numElements ( ) [override], [virtual]`

Implements Statistics_if.

Here is the call graph for this function:

3.160.2.9 setCollector()

`void StatisticsDefaultImpl1::setCollector ( Collector_if * collector ) [override], [virtual]`

Implements Statistics_if.

3.160.2.10 setConfidenceLevel()

`void StatisticsDefaultImpl1::setConfidenceLevel ( double confidencelevel ) [override], [virtual]`

Implements Statistics_if.

3.160.2.11 stddeviation()

`double StatisticsDefaultImpl1::stddeviation ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.12 variance()

`double StatisticsDefaultImpl1::variance ( ) [override], [virtual]`

Implements Statistics_if.

3.160.2.13 variationCoef() `double StatisticsDefaultImpl1::variationCoef ( ) [override], [virtual]`

Implements Statistics_if.

3.161 Stop Class Reference

`#include <Stop.h>`

Inheritance diagram for Stop:

Public Member Functions

- Stop (Model ∗model, std::string name="")
- virtual ∼Stop ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.161.1 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.

3.161.2 Constructor & Destructor Documentation

3.161.2.1 Stop()

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

3.161.2.2 ∼Stop()

```
virtual Stop::~Stop ( ) [virtual], [default]
```

3.161.3 Member Function Documentation

3.161.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.161.3.2 _loadInstance()

```
bool Stop::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.161.3.3 _onDispatchEvent()

```
void Stop::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.161.3.4 _saveInstance()

```
void Stop::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.161.3.5 GetPluginInformation() `PluginInformation * Stop::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.161.3.6 LoadInstance() `ModelComponent * Stop::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.161.3.7 NewInstance()

```
ModelDataDefinition * Stop::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.161.3.8 show()

```
std::string Stop::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.162 Storage Class Reference

```
#include <Storage.h>
```

Inheritance diagram for Storage:

Public Member Functions

- Storage (Model *model, std::string name="")
- virtual ~Storage ()=default
- virtual std::string show ()
- void setTotalArea (double _totalArea)
- double getTotalArea () const

- void setCapacity (unsigned int _capacity)
- unsigned int getCapacity () const
- void setUnitsPerArea (double _unitsPerArea)
- double getUnitsPerArea () const

Static Public Member Functions

- static ModelDataDefinition ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static PluginInformation ∗ GetPluginInformation ()
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord ∗fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes
- virtual ParserChangesInformation ∗ _getParserChangesInformation ()

Additional Inherited Members

3.162.1 Constructor & Destructor Documentation

3.162.1.1 Storage()

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

Here is the call graph for this function:

3.162.1.2 ~Storage() `virtual Storage::~Storage ( ) [virtual], [default]`

3.162.2 Member Function Documentation

3.162.2.1 _check() `bool Storage::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.162.2.2 _getParserChangesInformation() `ParserChangesInformation * Storage::_getParserChanges↩ Information ( ) [protected], [virtual]`

This method returns all changes in the parser that are needed by plugins of this ModelDatas. When connecting a new plugin, ParserChangesInformation are used to change parser source code, whch is after compiled and dinamically linked to to simulator kernel to reflect the changes

Reimplemented from ModelDataDefinition.

3.162.2.3 _loadInstance() `bool Storage::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.162.2.4 _saveInstance() `void Storage::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.162.2.5 getCapacity() `unsigned int Storage::getCapacity ( ) const`

3.162.2.6 GetPluginInformation() `PluginInformation * Storage::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.162.2.7 getTotalArea() `double Storage::getTotalArea ( ) const`

3.162.2.8 getUnitsPerArea() `double Storage::getUnitsPerArea ( ) const`

3.162.2.9 LoadInstance() `ModelDataDefinition * Storage::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.162.2.10 NewInstance()

```
ModelDataDefinition * Storage::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.162.2.11 setCapacity()

```
void Storage::setCapacity (
    unsigned int _capacity )
```

3.162.2.12 setTotalArea()

```
void Storage::setTotalArea (
    double _totalArea )
```

3.162.2.13 setUnitsPerArea()

```
void Storage::setUnitsPerArea (
    double _unitsPerArea )
```

3.162.2.14 show()

```
std::string Storage::show ( )  [virtual]
```

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.163 Store Class Reference

```
#include <Store.h>
```

Inheritance diagram for Store:

Public Member Functions

- Store (Model *model, std::string name="")

- virtual ~Store ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.163.1 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.

3.163.2 Constructor & Destructor Documentation

3.163.2.1 Store()

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

3.163.2.2 ~Store() `virtual Store::~Store ( ) [virtual], [default]`

3.163.3 Member Function Documentation

3.163.3.1 _check() `bool Store::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.163.3.2 _loadInstance() `bool Store::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.163.3.3 _onDispatchEvent() `void Store::_onDispatchEvent ( Entity * entity, unsigned int inputPortNumber ) [protected], [virtual]`

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.163.3.4 _saveInstance()

```
void Store::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.163.3.5 GetPluginInformation()

```
PluginInformation * Store::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.163.3.6 LoadInstance()

```
ModelComponent * Store::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.163.3.7 NewInstance()

```
ModelDataDefinition * Store::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.163.3.8 show()

```
std::string Store::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.164 Submodel Class Reference

```
#include <Submodel.h>
```

Inheritance diagram for Submodel:

Public Member Functions

- Submodel (Model ∗model, std::string name="")
- virtual ∼Submodel ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.164.1 Detailed Description

This component ...

3.164.2 Constructor & Destructor Documentation

3.164.2.1 Submodel()

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

3.164.2.2 ~Submodel()

```
virtual Submodel::~Submodel ( ) [virtual], [default]
```

3.164.3 Member Function Documentation

3.164.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.164.3.2 _loadInstance()

```
bool Submodel::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.164.3.3 _onDispatchEvent()

```
void Submodel::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.164.3.4 _saveInstance()

```
void Submodel::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.164.3.5 GetPluginInformation() `PluginInformation * Submodel::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.164.3.6 LoadInstance() `ModelComponent * Submodel::LoadInstance ( Model * model, PersistenceRecord * fields ) [static]`

Here is the call graph for this function:

3.164.3.7 NewInstance()

```
ModelDataDefinition * Submodel::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.164.3.8 show()

```
std::string Submodel::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.165 Token Struct Reference

Public Types

- enum Kind {
  eof , null , f , t ,
  number , string , lbracket , rbracket ,
  lbrace , rbrace , comma , colon }

Public Attributes

- enum Token::Kind kind
- std::string str {}

3.165.1 Member Enumeration Documentation

3.165.1.1 Kind

```
enum Token::Kind
```

Enumerator

eof	
null	
f	
t	
number	
string	
lbracket	
rbracket	
lbrace	
rbrace	
comma	
colon	

3.165.2 Member Data Documentation

3.165.2.1 kind `enum Token::Kind Token::kind`

3.165.2.2 str `std::string Token::str {}`

3.166 TraceErrorEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceErrorEvent:

Public Member Functions

- TraceErrorEvent (std::string text, std::exception e)
- std::exception getException () const

3.166.1 Constructor & Destructor Documentation

3.166.1.1 TraceErrorEvent()

```
TraceErrorEvent::TraceErrorEvent (
    std::string text,
    std::exception e ) [inline]
```

3.166.2 Member Function Documentation

3.166.2.1 getException()

```
std::exception TraceErrorEvent::getException ( ) const [inline]
```

3.167 TraceEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceEvent:

Public Member Functions

- TraceEvent (TraceManager::Level level, std::string text)
- TraceEvent (std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)
- TraceManager::Level getTracelevel () const
- std::string getText () const

3.167.1 Constructor & Destructor Documentation

3.167.1.1 TraceEvent() [1/2]

```
TraceEvent::TraceEvent (
    TraceManager::Level level,
    std::string text ) [inline]
```

3.167.1.2 TraceEvent() [2/2]

```
TraceEvent::TraceEvent (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed ) [inline]
```

3.167.2 Member Function Documentation

3.167.2.1 getText() `std::string TraceEvent::getText ( ) const [inline]`

3.167.2.2 getTracelevel() `TraceManager::Level TraceEvent::getTracelevel ( ) const [inline]`

3.168 TraceManager Class Reference

```
#include <TraceManager.h>
```

Public Types

- enum class Level : int {
  L0_noTraces = 0 , L1_errorFatal = 1 , L2_results = 2 , L3_errorRecover = 3 ,
  L4_warning = 4 , L5_event = 5 , L6_arrival = 6 , L7_internal = 7 ,
  L8_detailed = 8 , L9_mostDetailed = 9 }

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 addTraceSimulationExceptionRuleModelData (void *thisobject)
- void trace (TraceManager::Level level, std::string text)
- void traceSimulation (void *thisobject, TraceManager::Level level, double time, Entity *entity, ModelComponent *component, std::string text)
- void traceSimulation (void *thisobject, TraceManager::Level level, std::string text)
- void trace (std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)
- void traceError (std::string text, std::exception e)
- void traceError (std::string text, TraceManager::Level level=TraceManager::Level::L1_errorFatal)
- void traceReport (std::string text, TraceManager::Level level=TraceManager::Level::L2_results)
- void traceSimulation (void *thisobject, double time, Entity *entity, ModelComponent *component, std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)
- void traceSimulation (void *thisobject, std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)
- List< std::string > * errorMessages () const
- void setTraceLevel (TraceManager::Level _traceLevel)
- TraceManager::Level getTraceLevel () const
- Simulator * getParentSimulator () const
- void setTraceSimulationRuleAllAllowed (bool _traceSimulationRuleAllAllowed)
- bool isTraceSimulationRuleAllAllowed () const

3.168.1 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.

3.168.2 Member Enumeration Documentation

3.168.2.1 Level `enum class TraceManager::Level : int [strong]`

Enumerator

L0_noTraces	
L1_errorFatal	
L2_results	
L3_errorRecover	
L4_warning	
L5_event	
L6_arrival	
L7_internal	
L8_detailed	
L9_mostDetailed	

3.168.3 Constructor & Destructor Documentation

3.168.3.1 TraceManager()

```
TraceManager::TraceManager (
    Simulator * simulator )
```

3.168.3.2 ~TraceManager() `virtual TraceManager::~TraceManager ( ) [virtual], [default]`

3.168.4 Member Function Documentation

3.168.4.1 addTraceErrorHandler() [1/2]

```
template<typename Class >
void TraceManager::addTraceErrorHandler (
    Class * object,
    void(Class::*)(TraceErrorEvent) function )
```

3.168.4.2 addTraceErrorHandler() [2/2]

```
void TraceManager::addTraceErrorHandler (
            traceErrorListener traceErrorListener )
```

3.168.4.3 addTraceHandler() [1/2]

```
template<typename Class >
void TraceManager::addTraceHandler (
            Class * object,
            void(Class::*)(TraceEvent) function )
```

3.168.4.4 addTraceHandler() [2/2]

```
void TraceManager::addTraceHandler (
            traceListener traceListener )
```

3.168.4.5 addTraceReportHandler() [1/2]

```
template<typename Class >
void TraceManager::addTraceReportHandler (
            Class * object,
            void(Class::*)(TraceEvent) function )
```

3.168.4.6 addTraceReportHandler() [2/2]

```
void TraceManager::addTraceReportHandler (
            traceListener traceReportListener )
```

3.168.4.7 addTraceSimulationExceptionRuleModelData()

```
void TraceManager::addTraceSimulation↩
ExceptionRuleModelData (
            void * thisobject )
```

3.168.4.8 addTraceSimulationHandler() [1/2]

```
template<typename Class >
void TraceManager::addTraceSimulationHandler (
            Class * object,
            void(Class::*)(TraceSimulationEvent) function )
```

3.168.4.9 addTraceSimulationHandler() [2/2]

```
void TraceManager::addTraceSimulationHandler (
            traceSimulationListener traceSimulationListener )
```

3.168.4.10 errorMessages()

```
List< std::string > * TraceManager::errorMessages ( ) const
```

3.168.4.11 getParentSimulator()

```
Simulator * TraceManager::getParentSimulator ( ) const
```

3.168.4.12 getTraceLevel()

```
TraceManager::Level TraceManager::getTraceLevel ( ) const
```

3.168.4.13 isTraceSimulationRuleAllAllowed()

```
bool TraceManager::isTraceSimulationRuleAllAllowed
( ) const
```

3.168.4.14 setTraceLevel()

```
void TraceManager::setTraceLevel (
    TraceManager::Level _traceLevel )
```

3.168.4.15 setTraceSimulationRuleAllAllowed()

```
void TraceManager::setTraceSimulationRuleAll↩
Allowed (
    bool _traceSimulationRuleAllAllowed )
```

3.168.4.16 trace() [1/2]

```
void TraceManager::trace (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed )
```

Here is the call graph for this function:

3.168.4.17 trace() [2/2]

```
void TraceManager::trace (
    TraceManager::Level level,
    std::string text )
```

Here is the call graph for this function:

3.168.4.18 traceError() [1/2]

```
void TraceManager::traceError (
    std::string text,
    std::exception e )
```

Here is the call graph for this function:

3.168.4.19 traceError() [2/2]

```
void TraceManager::traceError (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L1_errorFatal )
```

Here is the call graph for this function:

3.168.4.20 traceReport()

```
void TraceManager::traceReport (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L2_results )
```

Here is the call graph for this function:

3.168.4.21 traceSimulation() [1/4]

```
void TraceManager::traceSimulation (
    void * thisobject,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed )
```

Here is the call graph for this function:

3.168.4.22 traceSimulation() [2/4]

```
void TraceManager::traceSimulation (
    void * thisobject,
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed )
```

Here is the call graph for this function:

3.168.4.23 traceSimulation() [3/4]

```
void TraceManager::traceSimulation (
    void * thisobject,
    TraceManager::Level level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text )
```

Here is the call graph for this function:

3.168.4.24 traceSimulation() [4/4]

```
void TraceManager::traceSimulation (
    void * thisobject,
    TraceManager::Level level,
    std::string text )
```

Here is the call graph for this function:

3.169 TraceSimulationEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationEvent:

Public Member Functions

- TraceSimulationEvent (TraceManager::Level level, double time, Entity *entity, ModelComponent *component, std::string text)
- ModelComponent * getComponent () const
- Entity * getEntity () const
- double getTime () const

3.169.1 Constructor & Destructor Documentation

3.169.1.1 TraceSimulationEvent()

```
TraceSimulationEvent::TraceSimulationEvent (
    TraceManager::Level level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text ) [inline]
```

3.169.2 Member Function Documentation

3.169.2.1 getComponent() `ModelComponent * TraceSimulationEvent::getComponent ( ) const [inline]`

3.169.2.2 getEntity() `Entity * TraceSimulationEvent::getEntity ( ) const [inline]`

3.169.2.3 getTime() `double TraceSimulationEvent::getTime ( ) const [inline]`

3.170 TraceSimulationProcess Class Reference

`#include <TraceManager.h>`

Inheritance diagram for TraceSimulationProcess:

Public Member Functions

- TraceSimulationProcess (std::string text, TraceManager::Level level=TraceManager::Level::L8_detailed)

3.170.1 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)

3.170.2 Constructor & Destructor Documentation

3.170.2.1 TraceSimulationProcess()

```
TraceSimulationProcess::TraceSimulationProcess (
    std::string text,
    TraceManager::Level level = TraceManager::Level::L8_detailed ) [inline]
```

3.171 TraitsApp< T > Struct Template Reference

```
#include <TraitsApp.h>
```

3.172 TraitsApp< GenesysApplication_if > Struct Reference

```
#include <TraitsApp.h>
```

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L9_mostDetailed
- static const bool runTests = false
- static const bool runGraphicalUserInterface = false

  If false, a terminal application will be compiled and executed (See TraitsTerminalApp.h). If true, a GUI for a general simulator will be executed.

3.172.1 Detailed Description

Configure the Genesys Application

3.172.2 Member Data Documentation

3.172.2.1 runGraphicalUserInterface

```
const bool TraitsApp< GenesysApplication_if >::runGraphical↩
UserInterface = false [static]
```

If false, a terminal application will be compiled and executed (See TraitsTerminalApp.h). If true, a GUI for a general simulator will be executed.

3.172.2.2 runTests

```
const bool TraitsApp< GenesysApplication_if >::runTests = false [static]
```

3.172.2.3 traceLevel

`const TraceManager::Level TraitsApp< GenesysApplication_if >::traceLevel = TraceManager::Level::L9_mostDetailed [static]`

3.173 TraitsKernel< T > Struct Template Reference

`#include <TraitsKernel.h>`

3.174 TraitsKernel< Collector_if > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef CollectorDatafileDefaultImpl1 Implementation
- typedef double DataType

3.174.1 Member Typedef Documentation

3.174.1.1 DataType

`typedef double TraitsKernel< Collector_if >::DataType`

3.174.1.2 Implementation

`typedef CollectorDatafileDefaultImpl1 TraitsKernel< Collector_if >↩ ::Implementation`

3.175 TraitsKernel< Model > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation
- typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation

Static Public Attributes

- static constexpr bool automaticallyCreatesModelData = true
- static const TraceManager::Level traceLevel = TraceManager::Level::L5_event

3.175.1 Member Typedef Documentation

3.175.1.1 StatisticsCollector_CollectorImplementation

`typedef CollectorDefaultImpl1 TraitsKernel< Model >::StatisticsCollector_CollectorImplementation`

3.175.1.2 StatisticsCollector_StatisticsImplementation

`typedef StatisticsDefaultImpl1 TraitsKernel< Model >::StatisticsCollector_StatisticsImplementation`

3.175.2 Member Data Documentation

3.175.2.1 automaticallyCreatesModelData

`constexpr bool TraitsKernel< Model >::automatically↩ CreatesModelData = true [static], [constexpr]`

3.175.2.2 traceLevel

`const TraceManager::Level TraitsKernel< Model >::traceLevel = TraceManager::Level::L5_eve [static]`

3.176 TraitsKernel< ModelChecker_if > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef ModelCheckerDefaultImpl1 Implementation

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L2_results

3.176.1 Member Typedef Documentation

3.176.1.1 Implementation

`typedef ModelCheckerDefaultImpl1 TraitsKernel< ModelChecker_if >::↩ Implementation`

3.176.2 Member Data Documentation

3.176.2.1 traceLevel

`const TraceManager::Level TraitsKernel< ModelChecker_if >::traceLevel = TraceManager::Level::L2_results [static]`

3.177 TraitsKernel< ModelComponent > Struct Reference

`#include <TraitsKernel.h>`

Static Public Attributes

- static constexpr bool reportStatistics = true
- static const TraceManager::Level traceLevel = TraceManager::Level::L2_results

3.177.1 Member Data Documentation

3.177.1.1 reportStatistics

`constexpr bool TraitsKernel< ModelComponent >::reportStatistics = true [static], [constexpr]`

3.177.1.2 traceLevel

`const TraceManager::Level TraitsKernel< ModelComponent >::traceLevel = TraceManager::Level::L2_results [static]`

3.178 TraitsKernel< ModelDataDefinition > Struct Reference

`#include <TraitsKernel.h>`

Static Public Attributes

- static constexpr bool reportStatistics = true
- static const TraceManager::Level traceLevel = TraceManager::Level::L2_results

3.178.1 Member Data Documentation

3.178.1.1 reportStatistics

`constexpr bool TraitsKernel< ModelDataDefinition >::reportStatistics = true [static], [constexpr]`

3.178.1.2 traceLevel

`const TraceManager::Level TraitsKernel< ModelDataDefinition >::traceLevel = TraceManager::Level::L2_results [static]`

3.179 TraitsKernel< ModelPersistence_if > Struct Reference

```
#include <TraitsKernel.h>
```

Public Types

- typedef ModelPersistenceDefaultImpl2 Implementation

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L2_results

3.179.1 Member Typedef Documentation

3.179.1.1 Implementation `typedef ModelPersistenceDefaultImpl2 TraitsKernel< ModelPersistence_if >::Implementation`

3.179.2 Member Data Documentation

3.179.2.1 traceLevel `const TraceManager::Level TraitsKernel< ModelPersistence_if >::traceLevel = TraceManager::Level::L2_results [static]`

3.180 TraitsKernel< Parser_if > Struct Reference

```
#include <TraitsKernel.h>
```

Public Types

- typedef ParserDefaultImpl2 Implementation

3.180.1 Member Typedef Documentation

3.180.1.1 Implementation `typedef ParserDefaultImpl2 TraitsKernel< Parser_if >::Implementation`

3.181 TraitsKernel< PluginConnector_if > Struct Reference

```
#include <TraitsKernel.h>
```

Public Types

- typedef PluginConnectorDummyImpl1 Implementation

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L4_warning

3.181.1 Member Typedef Documentation

3.181.1.1 Implementation `typedef PluginConnectorDummyImpl1 TraitsKernel< PluginConnector_if >::Implementation`

3.181.2 Member Data Documentation

3.181.2.1 traceLevel `const TraceManager::Level TraitsKernel< PluginConnector_if >::traceLevel = TraceManager::Level::L4_warning [static]`

3.182 TraitsKernel< Sampler_if > Struct Reference

```
#include <TraitsKernel.h>
```

Public Types

- typedef SamplerDefaultImpl1 Implementation
- typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters

3.182.1 Member Typedef Documentation

3.182.1.1 Implementation `typedef SamplerDefaultImpl1 TraitsKernel< Sampler_if >::Implementation`

3.182.1.2 Parameters

`typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters TraitsKernel< Sampler_if >::Parameters`

3.183 TraitsKernel< SimulationReporter_if > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef SimulationReporterDefaultImpl1 Implementation
- typedef Counter CounterImplementation

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L2_results

3.183.1 Member Typedef Documentation

3.183.1.1 CounterImplementation

`typedef Counter TraitsKernel< SimulationReporter_if >::↩ CounterImplementation`

3.183.1.2 Implementation

`typedef SimulationReporterDefaultImpl1 TraitsKernel< SimulationReporter_if >::Implementation`

3.183.2 Member Data Documentation

3.183.2.1 traceLevel

`const TraceManager::Level TraitsKernel< SimulationReporter_if >::trace↩ Level = TraceManager::Level::L2_results [static]`

3.184 TraitsKernel< Simulator > Struct Reference

`#include <TraitsKernel.h>`

Static Public Attributes

- static const TraceManager::Level traceLevel = TraceManager::Level::L9_mostDetailed

3.184.1 Member Data Documentation

3.184.1.1 traceLevel `const TraceManager::Level TraitsKernel< Simulator >::traceLevel = TraceManager::Level::L`
`[static]`

3.185 TraitsKernel< Statistics_if > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef double DataType

Static Public Attributes

- static constexpr double SignificanceLevel = 0.05

3.185.1 Member Typedef Documentation

3.185.1.1 DataType `typedef double TraitsKernel< Statistics_if >::DataType`

3.185.2 Member Data Documentation

3.185.2.1 SignificanceLevel `constexpr double TraitsKernel< Statistics_if >::SignificanceLevel = 0.05 [static], [constexpr]`

3.186 TraitsKernel< StatisticsDatafile_if > Struct Reference

`#include <TraitsKernel.h>`

Public Types

- typedef StatisticsDatafileDefaultImpl1 Implementation
- typedef CollectorDatafileDefaultImpl1 CollectorImplementation

Static Public Attributes

- static constexpr double SignificanceLevel = 0.05

3.186.1 Member Typedef Documentation

3.186.1.1 CollectorImplementation

`typedef CollectorDatafileDefaultImpl1 TraitsKernel< StatisticsDatafile_if >::CollectorImplementation`

3.186.1.2 Implementation

`typedef StatisticsDatafileDefaultImpl1 TraitsKernel< StatisticsDatafile_if >::Implementation`

3.186.2 Member Data Documentation

3.186.2.1 SignificanceLevel

`constexpr double TraitsKernel< StatisticsDatafile_if >::SignificanceLevel = 0.05 [static], [constexpr]`

3.187 TraitsTerminalApp< T > Struct Template Reference

```
#include <TraitsTerminalApp.h>
```

3.188 TraitsTerminalApp< GenesysApplication_if > Struct Reference

```
#include <TraitsTerminalApp.h>
```

Public Types

- typedef GenesysShell Application

3.188.1 Detailed Description

Configure the Genesys Application to be compiled and executed

3.188.2 Member Typedef Documentation

3.188.2.1 Application `typedef GenesysShell TraitsTerminalApp< GenesysApplication_if >::↩ Application`

3.189 TraitsTools< T > Struct Template Reference

`#include <TraitsTools.h>`

3.190 TraitsTools< Fitter_if > Struct Reference

`#include <TraitsTools.h>`

Public Types

- typedef FitterDummyImpl Implementation

3.190.1 Detailed Description

Configure the Fitter to be used

3.190.2 Member Typedef Documentation

3.190.2.1 Implementation `typedef FitterDummyImpl TraitsTools< Fitter_if >::Implementation`

3.191 TraitsTools< HypothesisTester_if > Struct Reference

`#include <TraitsTools.h>`

Public Types

- typedef HypothesisTesterDefaultImpl1 Implementation

Static Public Attributes

- static constexpr unsigned int ConfidenceLevel = 95

3.191.1 Detailed Description

Configure the Hypothesis Tester to be used

3.191.2 Member Typedef Documentation

3.191.2.1 Implementation `typedef HypothesisTesterDefaultImpl1 TraitsTools< HypothesisTester_if >::Implementation`

3.191.3 Member Data Documentation

3.191.3.1 ConfidenceLevel `constexpr unsigned int TraitsTools< HypothesisTester_if >::Confidence↩ Level = 95 [static], [constexpr]`

3.192 TraitsTools< Solver_if > Struct Reference

```
#include <TraitsTools.h>
```

Public Types

- typedef SolverDefaultImpl1 Implementation

Static Public Attributes

- static constexpr double Precision = 1e-5
- static constexpr unsigned int MaxSteps = 1e2

3.192.1 Detailed Description

Configure the Solver to be used

3.192.2 Member Typedef Documentation

3.192.2.1 Implementation `typedef SolverDefaultImpl1 TraitsTools< Solver_if >::Implementation`

3.192.3 Member Data Documentation

3.192.3.1 MaxSteps `constexpr unsigned int TraitsTools< Solver_if >::MaxSteps = 1e2 [static], [constexpr]`

3.192.3.2 Precision `constexpr double TraitsTools< Solver_if >::Precision = 1e-5 [static], [constexpr]`

3.193 FSM_Transition::TransitionType Struct Reference

`#include <EFSM.h>`

Public Attributes

- bool isdefault = false
- bool nondeterministic = false
- bool immediatbe = false
- bool preemptive = false
- bool history = false
- bool error = false
- bool termination = false

3.193.1 Member Data Documentation

3.193.1.1 error `bool FSM_Transition::TransitionType::error = false`

3.193.1.2 history `bool FSM_Transition::TransitionType::history = false`

3.193.1.3 immediatbe `bool FSM_Transition::TransitionType::immediatbe = false`

3.193.1.4 isdefault `bool FSM_Transition::TransitionType::isdefault = false`

3.193.1.5 nondeterministic `bool FSM_Transition::TransitionType::nondeterministic = false`

3.193.1.6 preemptive `bool FSM_Transition::TransitionType::preemptive = false`

3.193.1.7 termination `bool FSM_Transition::TransitionType::termination = false`

3.194 Unstore Class Reference

`#include <Unstore.h>`

Inheritance diagram for Unstore:

Public Member Functions

- Unstore (Model ∗model, std::string name="")
- virtual ∼Unstore ()=default
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation ∗ GetPluginInformation ()
- static ModelComponent ∗ LoadInstance (Model ∗model, PersistenceRecord ∗fields)
- static ModelDataDefinition ∗ NewInstance (Model ∗model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity ∗entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord ∗fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord ∗fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string ∗errorMessage)

  could be overriden by derived classes

Additional Inherited Members

3.194.1 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 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.

3.194.2 Constructor & Destructor Documentation

3.194.2.1 Unstore()

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

3.194.2.2 ~Unstore() `virtual Unstore::~Unstore ( ) [virtual], [default]`

3.194.3 Member Function Documentation

3.194.3.1 _check() `bool Unstore::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.194.3.2 _loadInstance() `bool Unstore::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.194.3.3 _onDispatchEvent() `void Unstore::_onDispatchEvent ( Entity * entity, unsigned int inputPortNumber ) [protected], [virtual]`

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.194.3.4 _saveInstance()

```
void Unstore::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.194.3.5 GetPluginInformation()

```
PluginInformation * Unstore::GetPluginInformation ( ) [static]
```

Here is the call graph for this function:

3.194.3.6 LoadInstance()

```
ModelComponent * Unstore::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.194.3.7 NewInstance()

```
ModelDataDefinition * Unstore::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.194.3.8 show()

```
std::string Unstore::show ( )  [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.195 Util Class Reference

```
#include <Util.h>
```

Public Types

- enum class TimeUnit : int {
  unknown = 0 , picosecond = 1 , nanosecond = 2 , microsecond = 3 ,
  milisecond = 4 , second = 5 , minute = 6 , hour = 7 ,
  day = 8 , week = 9 }
- enum class AllocationType : int {
  ValueAdded = 1 , NonValueAdded = 2 , Transfer = 3 , Wait = 4 ,
  Others = 5 }
- typedef unsigned long identification
- typedef unsigned int rank

Static Public Member Functions

- static std::string StrTimeUnitShort (Util::TimeUnit timeUnit)
- static std::string StrTimeUnitLong (Util::TimeUnit timeUnit)
- static std::string StrAllocation (Util::AllocationType allocation)
- static void SetIndent (const unsigned short indent)
- static void IncIndent ()
- static void DecIndent ()
- 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 StrTruncIfInt (double value)
- static std::string StrTruncIfInt (std::string strValue)
- static std::string Trim (std::string str)
- static std::string StrReplace (std::string text, std::string searchFor, std::string replaceBy)
- static std::string StrReplaceSpecialChars (std::string text)
- static std::string StrIndex (int index)

  returns a string in the form "[<index>] for array indexes"
- static void Trimwithin (std::string &str)
- static std::string Map2str (std::map< std::string, std::string > *mapss)
- static std::string Map2str (std::map< std::string, double > *mapss)
- static std::string List2str (std::list< unsigned int > *list)
- 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 void ResetAllIds ()
- static double TimeUnitConvert (Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2)
- static char DirSeparator ()
- static std::string FilenameFromFullFilename (const std::string &s)
- static void FileDelete (const std::string &filename)
- static std::string PathFromFullFilename (const std::string &s)
- static std::string RunningPath ()
- static std::vector< std::string > ListFiles (std::string dir, std::string fileFilter="", mode_t attribFilter=S_IFREG &S_IFDIR)
- static bool FileExists (const std::string &name)
- template<typename T >
  static std::string TypeOf ()
- template<class T >
  static Util::identification GenerateNewIdOfType ()

3.195.1 Member Typedef Documentation

3.195.1.1 identification `typedef unsigned long Util::identification`

3.195.1.2 rank `typedef unsigned int Util::rank`

3.195.2 Member Enumeration Documentation

3.195.2.1 AllocationType `enum class Util::AllocationType : int [strong]`

Enumerator

ValueAdded	
NonValueAdded	
Transfer	
Wait	
Others	

3.195.2.2 TimeUnit `enum class Util::TimeUnit : int [strong]`

Enumerator

unknown	
picosecond	
nanosecond	
microsecond	
milisecond	
second	
minute	
hour	
day	
week	

3.195.3 Member Function Documentation

3.195.3.1 DecIndent() `void Util::DecIndent ( ) [static]`

3.195.3.2 DirSeparator() `char Util::DirSeparator ( ) [static]`

3.195.3.3 FileDelete() `void Util::FileDelete ( const std::string & filename ) [static]`

3.195.3.4 FileExists() `bool Util::FileExists ( const std::string & name ) [static]`

3.195.3.5 FilenameFromFullFilename()

```
std::string Util::FilenameFromFullFilename (
    const std::string & s ) [static]
```

Here is the call graph for this function:

3.195.3.6 GenerateNewId()

```
Util::identification Util::GenerateNewId ( ) [static]
```

3.195.3.7 GenerateNewIdOfType() [1/2]

```
template<class T >
static Util::identification Util::GenerateNewIdOfType ( ) [inline], [static]
```

Every component or modeldatum has a unique ID for its class, but not unique for other classes. IDs are generated sequentially for each class.

3.195.3.8 GenerateNewIdOfType() [2/2]

```
Util::identification Util::GenerateNewIdOfType (
    std::string objtype ) [static]
```

3.195.3.9 GetLastIdOfType()

```
Util::identification Util::GetLastIdOfType (
    std::string objtype ) [static]
```

3.195.3.10 IncIndent()

```
void Util::IncIndent ( ) [static]
```

3.195.3.11 Indent()

```
std::string Util::Indent ( ) [static]
```

3.195.3.12 List2str()

```
std::string Util::List2str (
    std::list< unsigned int > * list ) [static]
```

3.195.3.13 ListFiles()

```
std::vector< std::string > Util::ListFiles (
    std::string dir,
    std::string fileFilter = "",
    mode_t attribFilter = S_IFREG & S_IFDIR ) [static]
```

3.195.3.14 Map2str() [1/2]

```
std::string Util::Map2str (
    std::map< std::string, double > * mapss ) [static]
```

Here is the call graph for this function:

3.195.3.15 Map2str() [2/2]

```
std::string Util::Map2str (
    std::map< std::string, std::string > * mapss ) [static]
```

3.195.3.16 PathFromFullFilename()

```
std::string Util::PathFromFullFilename (
    const std::string & s ) [inline], [static]
```

Here is the call graph for this function:

3.195.3.17 ResetAllIds()

```
void Util::ResetAllIds ( ) [static]
```

3.195.3.18 ResetIdOfType()

```
void Util::ResetIdOfType (
    std::string objtype )  [static]
```

3.195.3.19 RunningPath()

```
std::string Util::RunningPath ( )  [static]
```

Here is the call graph for this function:

3.195.3.20 SepKeyVal()

```
void Util::SepKeyVal (
    std::string str,
    std::string & key,
    std::string & value )  [static]
```

3.195.3.21 SetIndent()

```
void Util::SetIndent (
    const unsigned short indent )  [static]
```

3.195.3.22 SetW()

```
std::string Util::SetW (
    std::string text,
    unsigned short width )  [static]
```

3.195.3.23 StrAllocation()

```
std::string Util::StrAllocation (
    Util::AllocationType allocation )  [static]
```

3.195.3.24 StrIndex()

```
std::string Util::StrIndex (
    int index )  [static]
```

returns a string in the form "[<index>] for array indexes"

3.195.3.25 StrReplace()

```
std::string Util::StrReplace (
    std::string text,
    std::string searchFor,
    std::string replaceBy ) [static]
```

3.195.3.26 StrReplaceSpecialChars()

```
std::string Util::StrReplaceSpecialChars (
    std::string text ) [static]
```

Here is the call graph for this function:

3.195.3.27 StrTimeUnitLong()

```
std::string Util::StrTimeUnitLong (
    Util::TimeUnit timeUnit ) [static]
```

3.195.3.28 StrTimeUnitShort()

```
std::string Util::StrTimeUnitShort (
    Util::TimeUnit timeUnit ) [static]
```

3.195.3.29 StrTruncIfInt() [1/2]

```
std::string Util::StrTruncIfInt (
    double value ) [static]
```

3.195.3.30 StrTruncIfInt() [2/2]

```
std::string Util::StrTruncIfInt (
    std::string strValue ) [static]
```

3.195.3.31 TimeUnitConvert()

```
double Util::TimeUnitConvert (
    Util::TimeUnit timeUnit1,
    Util::TimeUnit timeUnit2 ) [static]
```

3.195.3.32 Trim()

```
std::string Util::Trim (
    std::string str )  [static]
```

3.195.3.33 Trimwithin()

```
void Util::Trimwithin (
    std::string & str )  [static]
```

3.195.3.34 TypeOf()

```
template<typename T >
static std::string Util::TypeOf ( )  [inline], [static]
```

Return the name of the class used as T.

3.196 Variable Class Reference

```
#include <Variable.h>
```

Inheritance diagram for Variable:

Public Member Functions

- Variable (Model ∗model, std::string name="")
- virtual ∼Variable ()=default
- virtual std::string show ()
- double getValue ()
- void setValue (double value)

- double getValue (std::string index)
- void setValue (std::string index, double value)
- double getInitialValue (std::string index="")
- void setInitialValue (double value, std::string index="")
- void insertDimentionSize (unsigned int size)
- std::list< unsigned int > * getDimensionSizes () const
- std::map< std::string, double > * getValues () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelDataDefinition * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  must be overriden by derived classes
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()

Additional Inherited Members

3.196.1 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.

3.196.2 Constructor & Destructor Documentation

3.196.2.1 Variable()

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

Here is the call graph for this function:

3.196.2.2 ~Variable()

```
virtual Variable::~Variable ( ) [virtual], [default]
```

3.196.3 Member Function Documentation

3.196.3.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.196.3.2 _initBetweenReplications() `void Variable::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.196.3.3 _loadInstance() `bool Variable::_loadInstance ( PersistenceRecord * fields ) [protected], [virtual]`

must be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.196.3.4 _saveInstance() `void Variable::_saveInstance ( PersistenceRecord * fields, bool saveDefaultValues ) [protected], [virtual]`

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.196.3.5 getDimensionSizes() `std::list< unsigned int > * Variable::getDimensionSizes ( ) const`

3.196.3.6 getInitialValue() `double Variable::getInitialValue ( std::string index = "" )`

3.196.3.7 GetPluginInformation() `PluginInformation * Variable::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.196.3.8 getValue() [1/2] `double Variable::getValue ( )`

Here is the call graph for this function:

3.196.3.9 getValue() [2/2] `double Variable::getValue ( std::string index )`

3.196.3.10 getValues()

```
std::map< std::string, double > * Variable::getValues ( ) const
```

3.196.3.11 insertDimentionSize()

```
void Variable::insertDimentionSize (
    unsigned int size )
```

3.196.3.12 LoadInstance()

```
ModelDataDefinition * Variable::LoadInstance (
    Model * model,
    PersistenceRecord * fields )  [static]
```

Here is the call graph for this function:

3.196.3.13 NewInstance()

```
ModelDataDefinition * Variable::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.196.3.14 setInitialValue()

```
void Variable::setInitialValue (
    double value,
    std::string index = "" )
```

3.196.3.15 setValue() [1/2] void Variable::setValue (double *value*)

Here is the call graph for this function:

3.196.3.16 setValue() [2/2] void Variable::setValue (std::string *index*, double *value*)

3.196.3.17 show() std::string Variable::show ( ) [virtual]

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.197 Wait Class Reference

```
#include <Wait.h>
```

Inheritance diagram for Wait:

Public Types

- enum class WaitType : int { WaitForSignal = 1 , InfiniteHold = 2 , ScanForCondition = 3 }

Public Member Functions

- Wait (Model ∗model, std::string name="")

- virtual ~Wait ()=default
- virtual std::string show ()

  virtual public methods
- void setSignalData (SignalData *signal)
- void setWaitType (WaitType _watType)
- Wait::WaitType getWaitType () const
- void setCondition (std::string _condition)
- std::string getCondition () const
- Queue * getQueue () const
- void setQueue (Queue *queue)
- std::string getlimitExpression () const
- void setLimitExpression (const std::string &newLimitExpression)

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()
- virtual void _createInternalAndAttachedData ()

Additional Inherited Members

3.197.1 Detailed Description

Wait 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 Wait 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.

3.197.2 Member Enumeration Documentation

3.197.2.1 WaitType `enum class Wait::WaitType : int [strong]`

Enumerator

WaitForSignal	
InfiniteHold	
ScanForCondition	

3.197.3 Constructor & Destructor Documentation

3.197.3.1 Wait()

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

3.197.3.2 ~Wait() `virtual Wait::~Wait ( ) [virtual], [default]`

3.197.4 Member Function Documentation

3.197.4.1 _check() `bool Wait::_check ( std::string * errorMessage ) [protected], [virtual]`

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.197.4.2 _createInternalAndAttachedData() `void Wait::_createInternalAndAttachedData ( ) [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 modeldatum must exist before checking such expression

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.197.4.3 _initBetweenReplications() `void Wait::_initBetweenReplications ( ) [protected], [virtual]`

Reimplemented from ModelDataDefinition.

3.197.4.4 _loadInstance()

```
bool Wait::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.197.4.5 _onDispatchEvent()

```
void Wait::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.197.4.6 _saveInstance()

```
void Wait::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.197.4.7 getCondition() `std::string Wait::getCondition ( ) const`

3.197.4.8 getlimitExpression() `std::string Wait::getlimitExpression ( ) const`

3.197.4.9 GetPluginInformation() `PluginInformation * Wait::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.197.4.10 getQueue() `Queue * Wait::getQueue ( ) const`

3.197.4.11 getWaitType() `Wait::WaitType Wait::getWaitType ( ) const`

3.197.4.12 LoadInstance()

```
ModelComponent * Wait::LoadInstance (
    Model * model,
    PersistenceRecord * fields )  [static]
```

Here is the call graph for this function:

3.197.4.13 NewInstance()

```
ModelDataDefinition * Wait::NewInstance (
    Model * model,
    std::string name = "" )  [static]
```

Here is the call graph for this function:

3.197.4.14 setCondition()

```
void Wait::setCondition (
    std::string _condition )
```

3.197.4.15 setLimitExpression()

```
void Wait::setLimitExpression (
    const std::string & newLimitExpression )
```

3.197.4.16 setQueue()

```
void Wait::setQueue (
    Queue * queue )
```

3.197.4.17 setSignalData()

```
void Wait::setSignalData (
    SignalData * signal )
```

3.197.4.18 setWaitType()

```
void Wait::setWaitType (
    WaitType _watType )
```

3.197.4.19 show()

```
std::string Wait::show ( ) [virtual]
```

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.198 Waiting Class Reference

```
#include <Queue.h>
```

Inheritance diagram for Waiting:

Public Member Functions

- Waiting (Entity ∗entity, double timeStartedWaiting, ModelComponent ∗thisComponent, unsigned int this↩ ComponentOutputPort=0)
- virtual ∼Waiting ()=default
- virtual std::string show ()
- double getTimeStartedWaiting () const
- ModelComponent ∗ geComponent () const
- Entity ∗ getEntity () const
- unsigned int geComponentOutputPort () const

3.198.1 Constructor & Destructor Documentation

3.198.1.1 Waiting()

```
Waiting::Waiting (
    Entity * entity,
    double timeStartedWaiting,
    ModelComponent * thisComponent,
    unsigned int thisComponentOutputPort = 0 )  [inline]
```

3.198.1.2 ~Waiting() `virtual Waiting::~Waiting ( ) [virtual], [default]`

3.198.2 Member Function Documentation

3.198.2.1 geComponent() `ModelComponent * Waiting::geComponent ( ) const [inline]`

3.198.2.2 geComponentOutputPort() `unsigned int Waiting::geComponentOutputPort ( ) const [inline]`

3.198.2.3 getEntity() `Entity * Waiting::getEntity ( ) const [inline]`

3.198.2.4 getTimeStartedWaiting() `double Waiting::getTimeStartedWaiting ( ) const [inline]`

3.198.2.5 show() `virtual std::string Waiting::show ( ) [inline], [virtual]`

Reimplemented in WaitingResource.

Here is the call graph for this function:

3.199 WaitingResource Class Reference

```
#include <Seize.h>
```

Inheritance diagram for WaitingResource:

Public Member Functions

- WaitingResource (Entity *entity, double timeStartedWaiting, unsigned int quantity, ModelComponent *this↩ Component, unsigned int thisComponentOutputPort=0)
- WaitingResource (const WaitingResource &orig)
- virtual ~WaitingResource ()=default
- virtual std::string show ()
- unsigned int getQuantity () const

3.199.1 Constructor & Destructor Documentation

3.199.1.1 WaitingResource() [1/2]

```
WaitingResource::WaitingResource (
    Entity * entity,
    double timeStartedWaiting,
    unsigned int quantity,
    ModelComponent * thisComponent,
    unsigned int thisComponentOutputPort = 0 ) [inline]
```

3.199.1.2 WaitingResource() [2/2]

```
WaitingResource::WaitingResource (
    const WaitingResource & orig ) [inline]
```

3.199.1.3 ~WaitingResource()

```
virtual WaitingResource::~WaitingResource ( ) [virtual], [default]
```

3.199.2 Member Function Documentation

3.199.2.1 getQuantity()

```
unsigned int WaitingResource::getQuantity ( ) const [inline]
```

3.199.2.2 show()

```
virtual std::string WaitingResource::show ( ) [inline], [virtual]
```

Reimplemented from Waiting.

Here is the call graph for this function:

3.200 Write Class Reference

```
#include <Write.h>
```

Inheritance diagram for Write:

Public Types

- enum class WriteToType : int { SCREEN = 1 , FILE = 2 }

Public Member Functions

- Write (Model *model, std::string name="")
- virtual ~Write ()=default
- void insertText (std::list< std::string > texts)
- void setFilename (std::string _filename)
- std::string filename () const
- void setWriteToType (WriteToType _writeToType)
- Write::WriteToType writeToType () const
- virtual std::string show ()

  virtual public methods

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, PersistenceRecord *fields)
- static ModelDataDefinition * NewInstance (Model *model, std::string name="")

Protected Member Functions

- virtual void _onDispatchEvent (Entity *entity, unsigned int inputPortNumber)

  new virtual methods for all ModelComponents
- virtual bool _loadInstance (PersistenceRecord *fields)

  virtual protected method that must be overriden
- virtual void _saveInstance (PersistenceRecord *fields, bool saveDefaultValues)

  Saves object fields into a serialization structure.
- virtual bool _check (std::string *errorMessage)

  could be overriden by derived classes
- virtual void _initBetweenReplications ()

Additional Inherited Members

3.200.1 Detailed Description

This component ...

3.200.2 Member Enumeration Documentation

3.200.2.1 WriteToType `enum class Write::WriteToType : int [strong]`

Enumerator

SCREEN	
FILE	

3.200.3 Constructor & Destructor Documentation

3.200.3.1 Write()

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

3.200.3.2 ~Write()

```
virtual Write::~Write ( ) [virtual], [default]
```

3.200.4 Member Function Documentation

3.200.4.1 _check()

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

could be overriden by derived classes

Reimplemented from ModelDataDefinition.

3.200.4.2 _initBetweenReplications()

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

Reimplemented from ModelDataDefinition.

Here is the call graph for this function:

3.200.4.3 _loadInstance()

```
bool Write::_loadInstance (
    PersistenceRecord * fields ) [protected], [virtual]
```

virtual protected method that must be overriden

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.200.4.4 _onDispatchEvent()

```
void Write::_onDispatchEvent (
    Entity * entity,
    unsigned int inputPortNumber ) [protected], [virtual]
```

new virtual methods for all ModelComponents

This method is only for ModelComponents, not ModelDataElements

Implements ModelComponent.

Here is the call graph for this function:

3.200.4.5 _saveInstance()

```
void Write::_saveInstance (
    PersistenceRecord * fields,
    bool saveDefaultValues ) [protected], [virtual]
```

Saves object fields into a serialization structure.

Parameters

fields	a table-like structure used for serialization.
saveDefaultValues	indicates whether or not to save fields which are set to their default value.

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.200.4.6 filename() `std::string Write::filename ( ) const`

3.200.4.7 GetPluginInformation() `PluginInformation * Write::GetPluginInformation ( ) [static]`

Here is the call graph for this function:

3.200.4.8 insertText() `void Write::insertText ( std::list< std::string > texts )`

3.200.4.9 LoadInstance()

```
ModelComponent * Write::LoadInstance (
    Model * model,
    PersistenceRecord * fields ) [static]
```

Here is the call graph for this function:

3.200.4.10 NewInstance()

```
ModelDataDefinition * Write::NewInstance (
    Model * model,
    std::string name = "" ) [static]
```

Here is the call graph for this function:

3.200.4.11 setFilename()

```
void Write::setFilename (
    std::string _filename )
```

3.200.4.12 setWriteToType()

```
void Write::setWriteToType (
    WriteToType _writeToType )
```

3.200.4.13 show() `std::string Write::show ( ) [virtual]`

virtual public methods

Reimplemented from ModelComponent.

Here is the call graph for this function:

3.200.4.14 writeToType() `Write::WriteToType Write::writeToType ( ) const`

3.201 XmlSerializer Class Reference

`#include <XmlSerializer.h>`

Inheritance diagram for XmlSerializer:

Public Member Functions

- XmlSerializer (Model *model)
- PersistenceRecord * newPersistenceRecord () override

  Creates an empty persistence record.
- bool dump (std::ostream &output) override

  Writes formatted contents to a given output stream.
- bool load (std::istream &input) override

  Attempts to load contents from a given input stream.
- bool get (const std::string &name, PersistenceRecord *entry) override

  Tries to fetch an entry with the given label as a key.
- bool put (const std::string name, const std::string type, const Util::identification id, PersistenceRecord *fields) override

  Upserts an entry into this structure.
- int for_each (std::function< int(const std::string &) > delegate) override

  Iterates over this structure's entries, ordered by ID.

3.201.1 Constructor & Destructor Documentation

3.201.1.1 XmlSerializer()

```
XmlSerializer::XmlSerializer (
    Model * model ) [explicit]
```

3.201.2 Member Function Documentation

3.201.2.1 dump()

```
bool XmlSerializer::dump (
    std::ostream & output ) [override], [virtual]
```

Writes formatted contents to a given output stream.

Parameters

output	output stream to write to.

Returns

whether the current state of this structure's contents was deemed valid for serialization.

Implements ModelSerializer.

Here is the call graph for this function:

3.201.2.2 for_each()

```
int XmlSerializer::for_each (
    std::function< int(const std::string &) > delegate )  [override], [virtual]
```

Iterates over this structure's entries, ordered by ID.

Parameters

delegate	delegate which is applied to every entry-identifying key, until it returns a non-zero value.

Returns

either zero, or the first non-zero value returned by the delegate.

Implements ModelSerializer.

3.201.2.3 get()

```
bool XmlSerializer::get (
    const std::string & name,
    PersistenceRecord * entry )  [override], [virtual]
```

Tries to fetch an entry with the given label as a key.

Parameters

name	a label which uniquely identifies an entry.
entry	(out-param) a pointer to a caller-owned entry, which is filled on success.

Returns

whether the entry was found and the out-param was actually written to.

Implements ModelSerializer.

Here is the call graph for this function:

3.201.2.4 load() `bool XmlSerializer::load ( std::istream & input ) [override], [virtual]`

Attempts to load contents from a given input stream.

Returns

whether the input was well-formatted and loading was successful.

Implements ModelSerializer.

Here is the call graph for this function:

3.201.2.5 newPersistenceRecord() `PersistenceRecord * XmlSerializer::newPersistenceRecord ( ) [override], [virtual]`

Creates an empty persistence record.

Although the returned record can be freed normally with `delete`, care must be taken to only use records which come from a implementation, when configuring that implementation's contents.

Implements ModelSerializer.

Here is the call graph for this function:

3.201.2.6 put()

```
bool XmlSerializer::put (
    const std::string name,
    const std::string type,
    const Util::identification id,
    PersistenceRecord * fields ) [override], [virtual]
```

Upserts an entry into this structure.

Parameters

name	a label which uniquely identifies the entry.
type	a type name.
id	a component/datadef ID number.
fields	a persistence record for the entry, whose contents will be copied if valid.

Returns

whether the entry was valid and was actually copied into this structure.

Implements ModelSerializer.

Here is the call graph for this function: