

OS
trunk

Generated by Doxygen 1.8.1.2

Mon Mar 16 2015 21:53:54

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	1
2.1	Class Hierarchy	1
3	Class Index	13
3.1	Class List	13
4	File Index	28
4.1	File List	28
5	Namespace Documentation	30
5.1	Coin Namespace Reference	30
5.2	Couenne Namespace Reference	30
6	Class Documentation	30
6.1	Base64 Class Reference	30
6.1.1	Detailed Description	31
6.1.2	Constructor & Destructor Documentation	31
6.1.3	Member Function Documentation	31
6.2	BaseMatrix Class Reference	32
6.2.1	Detailed Description	34
6.2.2	Constructor & Destructor Documentation	34
6.2.3	Member Function Documentation	34
6.2.4	Member Data Documentation	35
6.3	BasisStatus Class Reference	36
6.3.1	Detailed Description	38
6.3.2	Constructor & Destructor Documentation	38
6.3.3	Member Function Documentation	38
6.3.4	Member Data Documentation	40
6.4	BonminProblem Class Reference	41
6.4.1	Detailed Description	43
6.4.2	Constructor & Destructor Documentation	44
6.4.3	Member Function Documentation	44
6.4.4	Member Data Documentation	46
6.5	BonminSolver Class Reference	46

6.5.1	Detailed Description	48
6.5.2	Constructor & Destructor Documentation	48
6.5.3	Member Function Documentation	49
6.5.4	Member Data Documentation	49
6.6	BranchingWeight Class Reference	50
6.6.1	Detailed Description	51
6.6.2	Constructor & Destructor Documentation	51
6.6.3	Member Function Documentation	51
6.6.4	Member Data Documentation	52
6.7	CoinSolver Class Reference	52
6.7.1	Detailed Description	55
6.7.2	Constructor & Destructor Documentation	55
6.7.3	Member Function Documentation	55
6.7.4	Member Data Documentation	57
6.8	CompletelyPositiveMatricesCone Class Reference	57
6.8.1	Detailed Description	58
6.8.2	Constructor & Destructor Documentation	59
6.8.3	Member Function Documentation	59
6.9	Cone Class Reference	60
6.9.1	Detailed Description	63
6.9.2	Constructor & Destructor Documentation	63
6.9.3	Member Function Documentation	63
6.9.4	Member Data Documentation	64
6.10	Cones Class Reference	65
6.10.1	Detailed Description	66
6.10.2	Constructor & Destructor Documentation	66
6.10.3	Member Function Documentation	66
6.10.4	Member Data Documentation	67
6.11	ConReferenceMatrixElement Class Reference	67
6.11.1	Detailed Description	68
6.11.2	Constructor & Destructor Documentation	68
6.11.3	Member Function Documentation	68
6.11.4	Member Data Documentation	68
6.12	ConReferenceMatrixElements Class Reference	69
6.12.1	Detailed Description	70
6.12.2	Constructor & Destructor Documentation	71
6.12.3	Member Function Documentation	71

6.12.4 Member Data Documentation	72
6.13 ConReferenceMatrixValues Class Reference	73
6.13.1 Detailed Description	74
6.13.2 Constructor & Destructor Documentation	74
6.13.3 Member Function Documentation	74
6.13.4 Member Data Documentation	74
6.14 ConstantMatrixElements Class Reference	75
6.14.1 Detailed Description	77
6.14.2 Constructor & Destructor Documentation	77
6.14.3 Member Function Documentation	77
6.14.4 Member Data Documentation	78
6.15 ConstantMatrixValues Class Reference	79
6.15.1 Detailed Description	80
6.15.2 Constructor & Destructor Documentation	80
6.15.3 Member Function Documentation	80
6.15.4 Member Data Documentation	81
6.16 Constraint Class Reference	81
6.16.1 Detailed Description	82
6.16.2 Constructor & Destructor Documentation	82
6.16.3 Member Function Documentation	82
6.16.4 Member Data Documentation	82
6.17 ConstraintOption Class Reference	83
6.17.1 Detailed Description	84
6.17.2 Constructor & Destructor Documentation	84
6.17.3 Member Function Documentation	84
6.17.4 Member Data Documentation	85
6.18 Constraints Class Reference	86
6.18.1 Detailed Description	87
6.18.2 Constructor & Destructor Documentation	87
6.18.3 Member Function Documentation	87
6.18.4 Member Data Documentation	87
6.19 ConstraintSolution Class Reference	88
6.19.1 Detailed Description	89
6.19.2 Constructor & Destructor Documentation	89
6.19.3 Member Function Documentation	89
6.19.4 Member Data Documentation	90
6.20 ContactOption Class Reference	90

6.20.1 Detailed Description	91
6.20.2 Constructor & Destructor Documentation	92
6.20.3 Member Function Documentation	92
6.20.4 Member Data Documentation	93
6.21 CopositiveMatricesCone Class Reference	93
6.21.1 Detailed Description	94
6.21.2 Constructor & Destructor Documentation	95
6.21.3 Member Function Documentation	95
6.22 CouenneSolver Class Reference	96
6.22.1 Detailed Description	98
6.22.2 Constructor & Destructor Documentation	99
6.22.3 Member Function Documentation	99
6.22.4 Member Data Documentation	99
6.23 CPUNumber Class Reference	100
6.23.1 Detailed Description	101
6.23.2 Constructor & Destructor Documentation	102
6.23.3 Member Function Documentation	102
6.23.4 Member Data Documentation	103
6.24 CPUSpeed Class Reference	103
6.24.1 Detailed Description	104
6.24.2 Constructor & Destructor Documentation	104
6.24.3 Member Function Documentation	105
6.24.4 Member Data Documentation	105
6.25 CsdpSolver Class Reference	105
6.25.1 Detailed Description	108
6.25.2 Constructor & Destructor Documentation	108
6.25.3 Member Function Documentation	109
6.25.4 Member Data Documentation	109
6.26 DefaultSolver Class Reference	110
6.26.1 Detailed Description	112
6.26.2 Constructor & Destructor Documentation	113
6.26.3 Member Function Documentation	113
6.26.4 Member Data Documentation	113
6.27 DirectoriesAndFiles Class Reference	114
6.27.1 Detailed Description	116
6.27.2 Constructor & Destructor Documentation	116
6.27.3 Member Function Documentation	116

6.27.4	Member Data Documentation	117
6.28	DoubleVector Class Reference	117
6.28.1	Detailed Description	118
6.28.2	Constructor & Destructor Documentation	118
6.28.3	Member Function Documentation	118
6.28.4	Member Data Documentation	118
6.29	DualCone Class Reference	118
6.29.1	Detailed Description	120
6.29.2	Constructor & Destructor Documentation	120
6.29.3	Member Function Documentation	120
6.29.4	Member Data Documentation	121
6.30	DualVariableValues Class Reference	122
6.30.1	Detailed Description	123
6.30.2	Constructor & Destructor Documentation	123
6.30.3	Member Function Documentation	124
6.30.4	Member Data Documentation	124
6.31	DualVarValue Class Reference	124
6.31.1	Detailed Description	125
6.31.2	Constructor & Destructor Documentation	126
6.31.3	Member Function Documentation	126
6.31.4	Member Data Documentation	126
6.32	ErrorClass Class Reference	127
6.32.1	Detailed Description	128
6.32.2	Constructor & Destructor Documentation	128
6.32.3	Member Data Documentation	128
6.33	ExpandedMatrixBlocks Class Reference	128
6.33.1	Detailed Description	130
6.33.2	Constructor & Destructor Documentation	130
6.33.3	Member Function Documentation	130
6.33.4	Member Data Documentation	130
6.34	ExprNode Class Reference	132
6.34.1	Detailed Description	134
6.34.2	Constructor & Destructor Documentation	134
6.34.3	Member Function Documentation	134
6.34.4	Member Data Documentation	136
6.35	FileUtil Class Reference	137
6.35.1	Detailed Description	138

6.35.2	Constructor & Destructor Documentation	138
6.35.3	Member Function Documentation	138
6.36	GeneralFileHeader Class Reference	140
6.36.1	Detailed Description	141
6.36.2	Constructor & Destructor Documentation	141
6.36.3	Member Function Documentation	141
6.36.4	Member Data Documentation	142
6.37	GeneralMatrixElements Class Reference	143
6.37.1	Detailed Description	145
6.37.2	Constructor & Destructor Documentation	145
6.37.3	Member Function Documentation	145
6.37.4	Member Data Documentation	146
6.38	GeneralMatrixValues Class Reference	147
6.38.1	Detailed Description	148
6.38.2	Constructor & Destructor Documentation	148
6.38.3	Member Function Documentation	148
6.38.4	Member Data Documentation	149
6.39	GeneralOption Class Reference	149
6.39.1	Detailed Description	151
6.39.2	Constructor & Destructor Documentation	152
6.39.3	Member Function Documentation	152
6.39.4	Member Data Documentation	152
6.40	GeneralResult Class Reference	153
6.40.1	Detailed Description	155
6.40.2	Constructor & Destructor Documentation	155
6.40.3	Member Function Documentation	156
6.40.4	Member Data Documentation	156
6.41	GeneralSparseMatrix Class Reference	157
6.41.1	Detailed Description	158
6.41.2	Constructor & Destructor Documentation	158
6.41.3	Member Function Documentation	158
6.41.4	Member Data Documentation	159
6.42	GeneralStatus Class Reference	160
6.42.1	Detailed Description	161
6.42.2	Constructor & Destructor Documentation	161
6.42.3	Member Function Documentation	162
6.42.4	Member Data Documentation	162

6.43	GeneralSubstatus Class Reference	162
6.43.1	Detailed Description	163
6.43.2	Constructor & Destructor Documentation	164
6.43.3	Member Function Documentation	164
6.43.4	Member Data Documentation	164
6.44	IndexStringPair Struct Reference	165
6.44.1	Detailed Description	165
6.44.2	Member Data Documentation	165
6.45	IndexValuePair Struct Reference	166
6.45.1	Detailed Description	166
6.45.2	Member Data Documentation	166
6.46	InitBasStatus Class Reference	166
6.46.1	Detailed Description	167
6.46.2	Constructor & Destructor Documentation	168
6.46.3	Member Function Documentation	168
6.46.4	Member Data Documentation	169
6.47	InitConstraintValues Class Reference	169
6.47.1	Detailed Description	171
6.47.2	Constructor & Destructor Documentation	171
6.47.3	Member Function Documentation	171
6.47.4	Member Data Documentation	173
6.48	InitConValue Class Reference	173
6.48.1	Detailed Description	174
6.48.2	Constructor & Destructor Documentation	174
6.48.3	Member Function Documentation	175
6.48.4	Member Data Documentation	175
6.49	InitDualVariableValues Class Reference	176
6.49.1	Detailed Description	177
6.49.2	Constructor & Destructor Documentation	177
6.49.3	Member Function Documentation	177
6.49.4	Member Data Documentation	179
6.50	InitDualVarValue Class Reference	179
6.50.1	Detailed Description	181
6.50.2	Constructor & Destructor Documentation	181
6.50.3	Member Function Documentation	181
6.50.4	Member Data Documentation	182
6.51	InitialBasisStatus Class Reference	182

6.51.1 Detailed Description	184
6.51.2 Constructor & Destructor Documentation	184
6.51.3 Member Function Documentation	184
6.51.4 Member Data Documentation	185
6.52 InitObjBound Class Reference	185
6.52.1 Detailed Description	187
6.52.2 Constructor & Destructor Documentation	187
6.52.3 Member Function Documentation	187
6.52.4 Member Data Documentation	188
6.53 InitObjectiveBounds Class Reference	188
6.53.1 Detailed Description	190
6.53.2 Constructor & Destructor Documentation	190
6.53.3 Member Function Documentation	190
6.53.4 Member Data Documentation	192
6.54 InitObjectiveValues Class Reference	192
6.54.1 Detailed Description	194
6.54.2 Constructor & Destructor Documentation	194
6.54.3 Member Function Documentation	194
6.54.4 Member Data Documentation	196
6.55 InitObjValue Class Reference	196
6.55.1 Detailed Description	197
6.55.2 Constructor & Destructor Documentation	198
6.55.3 Member Function Documentation	198
6.55.4 Member Data Documentation	199
6.56 InitVariableValues Class Reference	199
6.56.1 Detailed Description	201
6.56.2 Constructor & Destructor Documentation	201
6.56.3 Member Function Documentation	201
6.56.4 Member Data Documentation	203
6.57 InitVariableValuesString Class Reference	203
6.57.1 Detailed Description	205
6.57.2 Constructor & Destructor Documentation	205
6.57.3 Member Function Documentation	205
6.57.4 Member Data Documentation	207
6.58 InitVarValue Class Reference	207
6.58.1 Detailed Description	208
6.58.2 Constructor & Destructor Documentation	209

6.58.3	Member Function Documentation	209
6.58.4	Member Data Documentation	210
6.59	InitVarValueString Class Reference	210
6.59.1	Detailed Description	212
6.59.2	Constructor & Destructor Documentation	212
6.59.3	Member Function Documentation	212
6.59.4	Member Data Documentation	213
6.60	InstanceData Class Reference	213
6.60.1	Detailed Description	214
6.60.2	Constructor & Destructor Documentation	215
6.60.3	Member Function Documentation	215
6.60.4	Member Data Documentation	215
6.61	InstanceLocationOption Class Reference	216
6.61.1	Detailed Description	217
6.61.2	Constructor & Destructor Documentation	217
6.61.3	Member Function Documentation	218
6.61.4	Member Data Documentation	218
6.62	IntegerVariableBranchingWeights Class Reference	218
6.62.1	Detailed Description	220
6.62.2	Constructor & Destructor Documentation	220
6.62.3	Member Function Documentation	220
6.62.4	Member Data Documentation	222
6.63	IntersectionCone Class Reference	222
6.63.1	Detailed Description	224
6.63.2	Constructor & Destructor Documentation	224
6.63.3	Member Function Documentation	224
6.63.4	Member Data Documentation	225
6.64	Interval Class Reference	226
6.64.1	Detailed Description	226
6.65	IntVector Class Reference	226
6.65.1	Detailed Description	228
6.65.2	Constructor & Destructor Documentation	228
6.65.3	Member Function Documentation	228
6.65.4	Member Data Documentation	229
6.66	IpoptProblem Class Reference	230
6.66.1	Detailed Description	232
6.66.2	Constructor & Destructor Documentation	232

6.66.3	Member Function Documentation	232
6.66.4	Member Data Documentation	233
6.67	IpoptSolver Class Reference	233
6.67.1	Detailed Description	236
6.67.2	Constructor & Destructor Documentation	236
6.67.3	Member Function Documentation	237
6.67.4	Member Data Documentation	237
6.68	JobDependencies Class Reference	238
6.68.1	Detailed Description	239
6.68.2	Constructor & Destructor Documentation	239
6.68.3	Member Function Documentation	239
6.68.4	Member Data Documentation	240
6.69	JobOption Class Reference	240
6.69.1	Detailed Description	242
6.69.2	Constructor & Destructor Documentation	242
6.69.3	Member Function Documentation	242
6.69.4	Member Data Documentation	243
6.70	JobResult Class Reference	244
6.70.1	Detailed Description	246
6.70.2	Constructor & Destructor Documentation	246
6.70.3	Member Function Documentation	247
6.70.4	Member Data Documentation	247
6.71	KnitroProblem Class Reference	248
6.71.1	Detailed Description	250
6.71.2	Constructor & Destructor Documentation	250
6.71.3	Member Function Documentation	250
6.71.4	Member Data Documentation	250
6.72	KnitroSolver Class Reference	251
6.72.1	Detailed Description	253
6.72.2	Constructor & Destructor Documentation	253
6.72.3	Member Function Documentation	253
6.73	LindoSolver Class Reference	254
6.73.1	Detailed Description	256
6.73.2	Constructor & Destructor Documentation	257
6.73.3	Member Function Documentation	257
6.73.4	Member Data Documentation	259
6.74	LinearConstraintCoefficients Class Reference	259

6.74.1 Detailed Description	260
6.74.2 Constructor & Destructor Documentation	260
6.74.3 Member Function Documentation	260
6.74.4 Member Data Documentation	260
6.75 LinearMatrixElement Class Reference	261
6.75.1 Detailed Description	262
6.75.2 Constructor & Destructor Documentation	262
6.75.3 Member Function Documentation	262
6.75.4 Member Data Documentation	263
6.76 LinearMatrixElements Class Reference	263
6.76.1 Detailed Description	265
6.76.2 Constructor & Destructor Documentation	265
6.76.3 Member Function Documentation	265
6.76.4 Member Data Documentation	267
6.77 LinearMatrixElementTerm Class Reference	267
6.77.1 Detailed Description	268
6.77.2 Constructor & Destructor Documentation	268
6.77.3 Member Function Documentation	268
6.77.4 Member Data Documentation	268
6.78 LinearMatrixValues Class Reference	269
6.78.1 Detailed Description	270
6.78.2 Constructor & Destructor Documentation	270
6.78.3 Member Function Documentation	270
6.78.4 Member Data Documentation	270
6.79 MathUtil Class Reference	271
6.79.1 Detailed Description	271
6.79.2 Constructor & Destructor Documentation	272
6.79.3 Member Function Documentation	272
6.80 Matrices Class Reference	273
6.80.1 Detailed Description	274
6.80.2 Constructor & Destructor Documentation	274
6.80.3 Member Function Documentation	274
6.80.4 Member Data Documentation	275
6.81 MatrixBlock Class Reference	275
6.81.1 Detailed Description	276
6.81.2 Constructor & Destructor Documentation	277
6.81.3 Member Function Documentation	277

6.81.4 Member Data Documentation	278
6.82 MatrixBlocks Class Reference	279
6.82.1 Detailed Description	281
6.82.2 Constructor & Destructor Documentation	281
6.82.3 Member Function Documentation	281
6.82.4 Member Data Documentation	283
6.83 MatrixCon Class Reference	283
6.83.1 Detailed Description	285
6.83.2 Constructor & Destructor Documentation	285
6.83.3 Member Function Documentation	285
6.83.4 Member Data Documentation	285
6.84 MatrixConstraints Class Reference	286
6.84.1 Detailed Description	287
6.84.2 Constructor & Destructor Documentation	288
6.84.3 Member Function Documentation	288
6.84.4 Member Data Documentation	288
6.85 MatrixConstructor Class Reference	288
6.85.1 Detailed Description	289
6.85.2 Constructor & Destructor Documentation	290
6.86 MatrixElements Class Reference	290
6.86.1 Detailed Description	291
6.86.2 Constructor & Destructor Documentation	292
6.86.3 Member Function Documentation	292
6.86.4 Member Data Documentation	292
6.87 MatrixElementValues Class Reference	292
6.87.1 Detailed Description	293
6.87.2 Constructor & Destructor Documentation	293
6.87.3 Member Data Documentation	293
6.88 MatrixExpression Class Reference	294
6.88.1 Detailed Description	295
6.88.2 Constructor & Destructor Documentation	295
6.88.3 Member Function Documentation	295
6.88.4 Member Data Documentation	295
6.89 MatrixExpressions Class Reference	296
6.89.1 Detailed Description	298
6.89.2 Constructor & Destructor Documentation	298
6.89.3 Member Function Documentation	298

6.89.4 Member Data Documentation	298
6.90 MatrixExpressionTree Class Reference	298
6.90.1 Detailed Description	301
6.90.2 Constructor & Destructor Documentation	301
6.90.3 Member Function Documentation	301
6.90.4 Member Data Documentation	302
6.91 MatrixNode Class Reference	302
6.91.1 Detailed Description	303
6.91.2 Constructor & Destructor Documentation	303
6.91.3 Member Function Documentation	304
6.91.4 Member Data Documentation	306
6.92 MatrixObj Class Reference	307
6.92.1 Detailed Description	308
6.92.2 Constructor & Destructor Documentation	308
6.92.3 Member Function Documentation	308
6.92.4 Member Data Documentation	308
6.93 MatrixObjectives Class Reference	309
6.93.1 Detailed Description	310
6.93.2 Constructor & Destructor Documentation	311
6.93.3 Member Function Documentation	311
6.93.4 Member Data Documentation	311
6.94 MatrixProgramming Class Reference	311
6.94.1 Detailed Description	313
6.94.2 Constructor & Destructor Documentation	313
6.94.3 Member Function Documentation	313
6.94.4 Member Data Documentation	313
6.95 MatrixTransformation Class Reference	314
6.95.1 Detailed Description	316
6.95.2 Constructor & Destructor Documentation	316
6.95.3 Member Function Documentation	316
6.95.4 Member Data Documentation	318
6.96 MatrixType Class Reference	318
6.96.1 Detailed Description	320
6.96.2 Constructor & Destructor Documentation	320
6.96.3 Member Function Documentation	320
6.96.4 Member Data Documentation	322
6.97 MatrixVar Class Reference	323

6.97.1 Detailed Description	325
6.97.2 Constructor & Destructor Documentation	325
6.97.3 Member Function Documentation	325
6.97.4 Member Data Documentation	325
6.98 MatrixVariables Class Reference	326
6.98.1 Detailed Description	327
6.98.2 Constructor & Destructor Documentation	328
6.98.3 Member Function Documentation	328
6.98.4 Member Data Documentation	328
6.99 MaxTime Class Reference	328
6.99.1 Detailed Description	329
6.99.2 Constructor & Destructor Documentation	330
6.99.3 Member Function Documentation	330
6.99.4 Member Data Documentation	330
6.100MinCPUNumber Class Reference	330
6.100.1 Detailed Description	331
6.100.2 Constructor & Destructor Documentation	332
6.100.3 Member Function Documentation	332
6.100.4 Member Data Documentation	332
6.101MinCPUSpeed Class Reference	332
6.101.1 Detailed Description	333
6.101.2 Constructor & Destructor Documentation	334
6.101.3 Member Function Documentation	334
6.101.4 Member Data Documentation	334
6.102MinDiskSpace Class Reference	335
6.102.1 Detailed Description	336
6.102.2 Constructor & Destructor Documentation	336
6.102.3 Member Function Documentation	336
6.102.4 Member Data Documentation	336
6.103MinMemorySize Class Reference	337
6.103.1 Detailed Description	338
6.103.2 Constructor & Destructor Documentation	338
6.103.3 Member Function Documentation	338
6.103.4 Member Data Documentation	338
6.104NI Class Reference	339
6.104.1 Detailed Description	341
6.104.2 Constructor & Destructor Documentation	341

6.104.3 Member Function Documentation	341
6.104.4 Member Data Documentation	341
6.105NonlinearExpressions Class Reference	342
6.105.1 Detailed Description	344
6.105.2 Constructor & Destructor Documentation	344
6.105.3 Member Function Documentation	344
6.105.4 Member Data Documentation	344
6.106NonnegativeCone Class Reference	345
6.106.1 Detailed Description	346
6.106.2 Constructor & Destructor Documentation	346
6.106.3 Member Function Documentation	346
6.107NonpositiveCone Class Reference	347
6.107.1 Detailed Description	349
6.107.2 Constructor & Destructor Documentation	349
6.107.3 Member Function Documentation	349
6.108ObjCoef Class Reference	350
6.108.1 Detailed Description	351
6.108.2 Constructor & Destructor Documentation	351
6.108.3 Member Function Documentation	351
6.108.4 Member Data Documentation	351
6.109Objective Class Reference	351
6.109.1 Detailed Description	353
6.109.2 Constructor & Destructor Documentation	353
6.109.3 Member Function Documentation	353
6.109.4 Member Data Documentation	353
6.110ObjectiveOption Class Reference	354
6.110.1 Detailed Description	355
6.110.2 Constructor & Destructor Documentation	355
6.110.3 Member Function Documentation	356
6.110.4 Member Data Documentation	356
6.111Objectives Class Reference	357
6.111.1 Detailed Description	358
6.111.2 Constructor & Destructor Documentation	359
6.111.3 Member Function Documentation	359
6.111.4 Member Data Documentation	359
6.112ObjectiveSolution Class Reference	359
6.112.1 Detailed Description	361

6.112.2 Constructor & Destructor Documentation	361
6.112.3 Member Function Documentation	361
6.112.4 Member Data Documentation	362
6.113ObjectiveValues Class Reference	362
6.113.1 Detailed Description	364
6.113.2 Constructor & Destructor Documentation	364
6.113.3 Member Function Documentation	364
6.113.4 Member Data Documentation	364
6.114ObjReferenceMatrixElements Class Reference	365
6.114.1 Detailed Description	367
6.114.2 Constructor & Destructor Documentation	367
6.114.3 Member Function Documentation	367
6.114.4 Member Data Documentation	369
6.115ObjReferenceMatrixValues Class Reference	369
6.115.1 Detailed Description	370
6.115.2 Constructor & Destructor Documentation	370
6.115.3 Member Function Documentation	370
6.115.4 Member Data Documentation	371
6.116ObjValue Class Reference	371
6.116.1 Detailed Description	372
6.116.2 Constructor & Destructor Documentation	372
6.116.3 Member Function Documentation	372
6.116.4 Member Data Documentation	373
6.117OptimizationOption Class Reference	373
6.117.1 Detailed Description	375
6.117.2 Constructor & Destructor Documentation	375
6.117.3 Member Function Documentation	375
6.117.4 Member Data Documentation	376
6.118OptimizationResult Class Reference	377
6.118.1 Detailed Description	378
6.118.2 Constructor & Destructor Documentation	378
6.118.3 Member Function Documentation	378
6.118.4 Member Data Documentation	379
6.119OptimizationSolution Class Reference	379
6.119.1 Detailed Description	381
6.119.2 Constructor & Destructor Documentation	381
6.119.3 Member Function Documentation	381

6.119.4 Member Data Documentation	382
6.120 OptimizationSolutionStatus Class Reference	383
6.120.1 Detailed Description	384
6.120.2 Constructor & Destructor Documentation	384
6.120.3 Member Function Documentation	384
6.120.4 Member Data Documentation	385
6.121 OptimizationSolutionSubstatus Class Reference	385
6.121.1 Detailed Description	386
6.121.2 Constructor & Destructor Documentation	387
6.121.3 Member Function Documentation	387
6.121.4 Member Data Documentation	387
6.122 OrthantCone Class Reference	388
6.122.1 Detailed Description	389
6.122.2 Constructor & Destructor Documentation	389
6.122.3 Member Function Documentation	389
6.122.4 Member Data Documentation	390
6.123 OS_AMPL_SUFFIX Struct Reference	391
6.123.1 Detailed Description	391
6.123.2 Member Data Documentation	391
6.124 OSCommandLine Class Reference	392
6.124.1 Detailed Description	395
6.124.2 Constructor & Destructor Documentation	395
6.124.3 Member Function Documentation	395
6.124.4 Member Data Documentation	395
6.125 OSCommandLineReader Class Reference	399
6.125.1 Detailed Description	400
6.125.2 Constructor & Destructor Documentation	400
6.125.3 Member Function Documentation	400
6.126 OSExpressionTree Class Reference	401
6.126.1 Detailed Description	403
6.126.2 Constructor & Destructor Documentation	403
6.126.3 Member Function Documentation	403
6.126.4 Member Data Documentation	403
6.127 OSgams2osil Class Reference	404
6.127.1 Detailed Description	404
6.127.2 Constructor & Destructor Documentation	405
6.127.3 Member Function Documentation	405

6.127.4 Member Data Documentation	405
6.128OSGeneral Class Reference	405
6.128.1 Detailed Description	405
6.129OSgLParseData Class Reference	406
6.129.1 Detailed Description	408
6.129.2 Constructor & Destructor Documentation	409
6.129.3 Member Data Documentation	409
6.130OShL Class Reference	416
6.130.1 Detailed Description	416
6.130.2 Constructor & Destructor Documentation	417
6.130.3 Member Function Documentation	417
6.131OSiLParseData Class Reference	419
6.131.1 Detailed Description	422
6.131.2 Constructor & Destructor Documentation	423
6.131.3 Member Data Documentation	423
6.132OSiLReader Class Reference	431
6.132.1 Detailed Description	431
6.132.2 Constructor & Destructor Documentation	431
6.132.3 Member Function Documentation	432
6.133OSiLWriter Class Reference	432
6.133.1 Detailed Description	433
6.133.2 Constructor & Destructor Documentation	433
6.133.3 Member Function Documentation	433
6.133.4 Member Data Documentation	433
6.134OSInstance Class Reference	434
6.134.1 Detailed Description	442
6.134.2 Constructor & Destructor Documentation	443
6.134.3 Member Function Documentation	443
6.134.4 Member Data Documentation	482
6.135OSMatlab Class Reference	483
6.135.1 Detailed Description	486
6.135.2 Constructor & Destructor Documentation	486
6.135.3 Member Function Documentation	486
6.135.4 Member Data Documentation	486
6.136OSMatrix Class Reference	489
6.136.1 Detailed Description	490
6.136.2 Constructor & Destructor Documentation	491

6.136.3 Member Function Documentation	491
6.136.4 Member Data Documentation	494
6.137OSmps2OS Class Reference	494
6.137.1 Detailed Description	496
6.137.2 Constructor & Destructor Documentation	496
6.137.3 Member Function Documentation	496
6.137.4 Member Data Documentation	497
6.138OSmps2osil Class Reference	497
6.138.1 Detailed Description	498
6.138.2 Constructor & Destructor Documentation	499
6.138.3 Member Function Documentation	499
6.138.4 Member Data Documentation	499
6.139OSnI2OS Class Reference	499
6.139.1 Detailed Description	501
6.139.2 Constructor & Destructor Documentation	502
6.139.3 Member Function Documentation	502
6.139.4 Member Data Documentation	504
6.140OSnLMNode Class Reference	504
6.140.1 Detailed Description	507
6.140.2 Constructor & Destructor Documentation	507
6.140.3 Member Function Documentation	507
6.141OSnLMNodeDiagonalMatrixFromVector Class Reference	509
6.141.1 Detailed Description	510
6.141.2 Constructor & Destructor Documentation	510
6.141.3 Member Function Documentation	511
6.142OSnLMNodeIdentityMatrix Class Reference	511
6.142.1 Detailed Description	512
6.142.2 Constructor & Destructor Documentation	512
6.142.3 Member Function Documentation	513
6.143OSnLMNodeMatrixCon Class Reference	513
6.143.1 Detailed Description	514
6.143.2 Constructor & Destructor Documentation	515
6.143.3 Member Function Documentation	515
6.143.4 Member Data Documentation	515
6.144OSnLMNodeMatrixDiagonal Class Reference	516
6.144.1 Detailed Description	517
6.144.2 Constructor & Destructor Documentation	517

6.144.3 Member Function Documentation	517
6.145OSnLMNodeMatrixDotTimes Class Reference	517
6.145.1 Detailed Description	519
6.145.2 Constructor & Destructor Documentation	519
6.145.3 Member Function Documentation	519
6.146OSnLMNodeMatrixInverse Class Reference	519
6.146.1 Detailed Description	521
6.146.2 Constructor & Destructor Documentation	521
6.146.3 Member Function Documentation	521
6.147OSnLMNodeMatrixLowerTriangle Class Reference	521
6.147.1 Detailed Description	523
6.147.2 Constructor & Destructor Documentation	523
6.147.3 Member Function Documentation	523
6.147.4 Member Data Documentation	524
6.148OSnLMNodeMatrixMinus Class Reference	524
6.148.1 Detailed Description	525
6.148.2 Constructor & Destructor Documentation	525
6.148.3 Member Function Documentation	526
6.149OSnLMNodeMatrixNegate Class Reference	526
6.149.1 Detailed Description	527
6.149.2 Constructor & Destructor Documentation	527
6.149.3 Member Function Documentation	528
6.150OSnLMNodeMatrixObj Class Reference	528
6.150.1 Detailed Description	529
6.150.2 Constructor & Destructor Documentation	530
6.150.3 Member Function Documentation	530
6.150.4 Member Data Documentation	530
6.151OSnLMNodeMatrixPlus Class Reference	531
6.151.1 Detailed Description	532
6.151.2 Constructor & Destructor Documentation	532
6.151.3 Member Function Documentation	532
6.152OSnLMNodeMatrixProduct Class Reference	532
6.152.1 Detailed Description	534
6.152.2 Constructor & Destructor Documentation	534
6.152.3 Member Function Documentation	534
6.153OSnLMNodeMatrixReference Class Reference	535
6.153.1 Detailed Description	536

6.153.2 Constructor & Destructor Documentation	537
6.153.3 Member Function Documentation	537
6.153.4 Member Data Documentation	537
6.154OSnLMNodeMatrixScalarTimes Class Reference	538
6.154.1 Detailed Description	539
6.154.2 Constructor & Destructor Documentation	539
6.154.3 Member Function Documentation	539
6.155OSnLMNodeMatrixSubmatrixAt Class Reference	539
6.155.1 Detailed Description	541
6.155.2 Constructor & Destructor Documentation	541
6.155.3 Member Function Documentation	541
6.156OSnLMNodeMatrixSum Class Reference	541
6.156.1 Detailed Description	543
6.156.2 Constructor & Destructor Documentation	543
6.156.3 Member Function Documentation	543
6.157OSnLMNodeMatrixTimes Class Reference	543
6.157.1 Detailed Description	545
6.157.2 Constructor & Destructor Documentation	545
6.157.3 Member Function Documentation	545
6.158OSnLMNodeMatrixTranspose Class Reference	545
6.158.1 Detailed Description	547
6.158.2 Constructor & Destructor Documentation	547
6.158.3 Member Function Documentation	547
6.159OSnLMNodeMatrixUpperTriangle Class Reference	547
6.159.1 Detailed Description	549
6.159.2 Constructor & Destructor Documentation	549
6.159.3 Member Function Documentation	549
6.159.4 Member Data Documentation	550
6.160OSnLMNodeMatrixVar Class Reference	550
6.160.1 Detailed Description	551
6.160.2 Constructor & Destructor Documentation	552
6.160.3 Member Function Documentation	552
6.160.4 Member Data Documentation	552
6.161OSnLNode Class Reference	553
6.161.1 Detailed Description	556
6.161.2 Constructor & Destructor Documentation	556
6.161.3 Member Function Documentation	556

6.161.4 Member Data Documentation	559
6.162OSnNodeAbs Class Reference	559
6.162.1 Detailed Description	561
6.162.2 Constructor & Destructor Documentation	561
6.162.3 Member Function Documentation	561
6.163OSnNodeAllDiff Class Reference	562
6.163.1 Detailed Description	564
6.163.2 Constructor & Destructor Documentation	564
6.163.3 Member Function Documentation	564
6.164OSnNodeCos Class Reference	565
6.164.1 Detailed Description	567
6.164.2 Constructor & Destructor Documentation	567
6.164.3 Member Function Documentation	567
6.165OSnNodeDivide Class Reference	568
6.165.1 Detailed Description	570
6.165.2 Constructor & Destructor Documentation	570
6.165.3 Member Function Documentation	570
6.166OSnNodeE Class Reference	571
6.166.1 Detailed Description	573
6.166.2 Constructor & Destructor Documentation	573
6.166.3 Member Function Documentation	573
6.167OSnNodeErf Class Reference	575
6.167.1 Detailed Description	576
6.167.2 Constructor & Destructor Documentation	576
6.167.3 Member Function Documentation	577
6.168OSnNodeExp Class Reference	577
6.168.1 Detailed Description	579
6.168.2 Constructor & Destructor Documentation	579
6.168.3 Member Function Documentation	579
6.169OSnNodeIrf Class Reference	580
6.169.1 Detailed Description	582
6.169.2 Constructor & Destructor Documentation	582
6.169.3 Member Function Documentation	582
6.170OSnNodeLn Class Reference	583
6.170.1 Detailed Description	585
6.170.2 Constructor & Destructor Documentation	585
6.170.3 Member Function Documentation	585

6.171OSnLNodeMatrixDeterminant Class Reference	586
6.171.1 Detailed Description	588
6.171.2 Constructor & Destructor Documentation	588
6.171.3 Member Function Documentation	588
6.172OSnLNodeMatrixToScalar Class Reference	589
6.172.1 Detailed Description	591
6.172.2 Constructor & Destructor Documentation	591
6.172.3 Member Function Documentation	591
6.173OSnLNodeMatrixTrace Class Reference	592
6.173.1 Detailed Description	594
6.173.2 Constructor & Destructor Documentation	594
6.173.3 Member Function Documentation	594
6.174OSnLNodeMax Class Reference	595
6.174.1 Detailed Description	597
6.174.2 Constructor & Destructor Documentation	597
6.174.3 Member Function Documentation	597
6.175OSnLNodeMin Class Reference	598
6.175.1 Detailed Description	600
6.175.2 Constructor & Destructor Documentation	600
6.175.3 Member Function Documentation	600
6.176OSnLNodeMinus Class Reference	601
6.176.1 Detailed Description	603
6.176.2 Constructor & Destructor Documentation	603
6.176.3 Member Function Documentation	603
6.177OSnLNodeNegate Class Reference	604
6.177.1 Detailed Description	606
6.177.2 Constructor & Destructor Documentation	606
6.177.3 Member Function Documentation	606
6.178OSnLNodeNumber Class Reference	607
6.178.1 Detailed Description	609
6.178.2 Constructor & Destructor Documentation	610
6.178.3 Member Function Documentation	610
6.178.4 Member Data Documentation	611
6.179OSnLNodePI Class Reference	611
6.179.1 Detailed Description	613
6.179.2 Constructor & Destructor Documentation	613
6.179.3 Member Function Documentation	613

6.180OSnNodePlus Class Reference	615
6.180.1 Detailed Description	616
6.180.2 Constructor & Destructor Documentation	616
6.180.3 Member Function Documentation	617
6.181OSnNodePower Class Reference	617
6.181.1 Detailed Description	619
6.181.2 Constructor & Destructor Documentation	619
6.181.3 Member Function Documentation	619
6.182OSnNodeProduct Class Reference	620
6.182.1 Detailed Description	622
6.182.2 Constructor & Destructor Documentation	622
6.182.3 Member Function Documentation	622
6.183OSnNodeSin Class Reference	623
6.183.1 Detailed Description	625
6.183.2 Constructor & Destructor Documentation	625
6.183.3 Member Function Documentation	625
6.184OSnNodeSqrt Class Reference	626
6.184.1 Detailed Description	628
6.184.2 Constructor & Destructor Documentation	628
6.184.3 Member Function Documentation	628
6.185OSnNodeSquare Class Reference	629
6.185.1 Detailed Description	631
6.185.2 Constructor & Destructor Documentation	631
6.185.3 Member Function Documentation	631
6.186OSnNodeSum Class Reference	632
6.186.1 Detailed Description	634
6.186.2 Constructor & Destructor Documentation	634
6.186.3 Member Function Documentation	634
6.187OSnNodeTimes Class Reference	635
6.187.1 Detailed Description	637
6.187.2 Constructor & Destructor Documentation	637
6.187.3 Member Function Documentation	637
6.188OSnNodeVariable Class Reference	638
6.188.1 Detailed Description	640
6.188.2 Constructor & Destructor Documentation	640
6.188.3 Member Function Documentation	641
6.188.4 Member Data Documentation	642

6.189OSnLParserData Class Reference	642
6.189.1 Detailed Description	646
6.189.2 Constructor & Destructor Documentation	646
6.189.3 Member Data Documentation	646
6.190OSoLParserData Class Reference	652
6.190.1 Detailed Description	657
6.190.2 Constructor & Destructor Documentation	657
6.190.3 Member Data Documentation	658
6.191OSoLReader Class Reference	671
6.191.1 Detailed Description	672
6.191.2 Constructor & Destructor Documentation	672
6.191.3 Member Function Documentation	672
6.192OSoLWriter Class Reference	673
6.192.1 Detailed Description	674
6.192.2 Constructor & Destructor Documentation	674
6.192.3 Member Function Documentation	674
6.192.4 Member Data Documentation	674
6.193OSOption Class Reference	675
6.193.1 Detailed Description	685
6.193.2 Constructor & Destructor Documentation	686
6.193.3 Member Function Documentation	686
6.193.4 Member Data Documentation	713
6.194osOptionsStruc Struct Reference	714
6.194.1 Detailed Description	716
6.194.2 Constructor & Destructor Documentation	716
6.194.3 Member Function Documentation	716
6.194.4 Member Data Documentation	717
6.195OSosrI2ampl Class Reference	720
6.195.1 Detailed Description	720
6.195.2 Constructor & Destructor Documentation	721
6.195.3 Member Function Documentation	721
6.196OSOutput Class Reference	721
6.196.1 Detailed Description	723
6.196.2 Constructor & Destructor Documentation	723
6.196.3 Member Function Documentation	723
6.197OSOutputChannel Class Reference	725
6.197.1 Detailed Description	725

6.197.2 Constructor & Destructor Documentation	726
6.197.3 Member Function Documentation	726
6.198OSReferencedObject Class Reference	727
6.198.1 Detailed Description	728
6.198.2 Constructor & Destructor Documentation	729
6.198.3 Member Function Documentation	729
6.199OSReferencer Class Reference	730
6.199.1 Detailed Description	730
6.200OSResult Class Reference	730
6.200.1 Detailed Description	744
6.200.2 Constructor & Destructor Documentation	744
6.200.3 Member Function Documentation	745
6.200.4 Member Data Documentation	810
6.201OSrL2Gams Class Reference	811
6.201.1 Detailed Description	811
6.201.2 Constructor & Destructor Documentation	812
6.201.3 Member Function Documentation	812
6.202OSrLParserData Class Reference	812
6.202.1 Detailed Description	817
6.202.2 Constructor & Destructor Documentation	817
6.202.3 Member Data Documentation	817
6.203OSrLReader Class Reference	827
6.203.1 Detailed Description	827
6.203.2 Constructor & Destructor Documentation	828
6.203.3 Member Function Documentation	828
6.204OSrLWriter Class Reference	828
6.204.1 Detailed Description	829
6.204.2 Constructor & Destructor Documentation	830
6.204.3 Member Function Documentation	830
6.204.4 Member Data Documentation	830
6.205OSSmartPtr< T > Class Template Reference	830
6.205.1 Detailed Description	832
6.205.2 Constructor & Destructor Documentation	834
6.205.3 Member Function Documentation	835
6.205.4 Friends And Related Function Documentation	835
6.206OSSolverAgent Class Reference	836
6.206.1 Detailed Description	837

6.206.2 Constructor & Destructor Documentation	838
6.206.3 Member Function Documentation	838
6.207 OtherConOption Class Reference	840
6.207.1 Detailed Description	841
6.207.2 Constructor & Destructor Documentation	841
6.207.3 Member Function Documentation	842
6.207.4 Member Data Documentation	842
6.208 OtherConResult Class Reference	843
6.208.1 Detailed Description	844
6.208.2 Constructor & Destructor Documentation	844
6.208.3 Member Function Documentation	844
6.208.4 Member Data Documentation	845
6.209 OtherConstraintOption Class Reference	845
6.209.1 Detailed Description	847
6.209.2 Constructor & Destructor Documentation	848
6.209.3 Member Function Documentation	848
6.209.4 Member Data Documentation	849
6.210 OtherConstraintResult Class Reference	850
6.210.1 Detailed Description	852
6.210.2 Constructor & Destructor Documentation	852
6.210.3 Member Function Documentation	852
6.210.4 Member Data Documentation	853
6.211 OtherObjectiveOption Class Reference	854
6.211.1 Detailed Description	856
6.211.2 Constructor & Destructor Documentation	856
6.211.3 Member Function Documentation	856
6.211.4 Member Data Documentation	857
6.212 OtherObjectiveResult Class Reference	858
6.212.1 Detailed Description	860
6.212.2 Constructor & Destructor Documentation	860
6.212.3 Member Function Documentation	860
6.212.4 Member Data Documentation	861
6.213 OtherObjOption Class Reference	862
6.213.1 Detailed Description	863
6.213.2 Constructor & Destructor Documentation	863
6.213.3 Member Function Documentation	864
6.213.4 Member Data Documentation	864

6.214OtherObjResult Class Reference	865
6.214.1 Detailed Description	866
6.214.2 Constructor & Destructor Documentation	866
6.214.3 Member Function Documentation	866
6.214.4 Member Data Documentation	867
6.215OtherOption Class Reference	867
6.215.1 Detailed Description	868
6.215.2 Constructor & Destructor Documentation	868
6.215.3 Member Function Documentation	869
6.215.4 Member Data Documentation	869
6.216OtherOptionEnumeration Class Reference	870
6.216.1 Detailed Description	871
6.216.2 Constructor & Destructor Documentation	871
6.216.3 Member Function Documentation	871
6.216.4 Member Data Documentation	872
6.217OtherOptions Class Reference	873
6.217.1 Detailed Description	874
6.217.2 Constructor & Destructor Documentation	874
6.217.3 Member Function Documentation	874
6.217.4 Member Data Documentation	875
6.218OtherResult Class Reference	876
6.218.1 Detailed Description	877
6.218.2 Constructor & Destructor Documentation	877
6.218.3 Member Function Documentation	877
6.218.4 Member Data Documentation	878
6.219OtherResults Class Reference	878
6.219.1 Detailed Description	880
6.219.2 Constructor & Destructor Documentation	880
6.219.3 Member Function Documentation	880
6.219.4 Member Data Documentation	880
6.220OtherSolutionResult Class Reference	881
6.220.1 Detailed Description	882
6.220.2 Constructor & Destructor Documentation	882
6.220.3 Member Function Documentation	883
6.220.4 Member Data Documentation	883
6.221OtherSolutionResults Class Reference	884
6.221.1 Detailed Description	885

6.221.2 Constructor & Destructor Documentation	885
6.221.3 Member Function Documentation	885
6.221.4 Member Data Documentation	886
6.222OtherSolverOutput Class Reference	886
6.222.1 Detailed Description	887
6.222.2 Constructor & Destructor Documentation	887
6.222.3 Member Function Documentation	888
6.222.4 Member Data Documentation	888
6.223OtherVariableOption Class Reference	888
6.223.1 Detailed Description	890
6.223.2 Constructor & Destructor Documentation	890
6.223.3 Member Function Documentation	890
6.223.4 Member Data Documentation	891
6.224OtherVariableResult Class Reference	893
6.224.1 Detailed Description	894
6.224.2 Constructor & Destructor Documentation	894
6.224.3 Member Function Documentation	895
6.224.4 Member Data Documentation	895
6.225OtherVariableResultStruct Struct Reference	896
6.225.1 Detailed Description	897
6.225.2 Member Data Documentation	897
6.226OtherVarOption Class Reference	898
6.226.1 Detailed Description	899
6.226.2 Constructor & Destructor Documentation	899
6.226.3 Member Function Documentation	899
6.226.4 Member Data Documentation	900
6.227OtherVarResult Class Reference	901
6.227.1 Detailed Description	902
6.227.2 Constructor & Destructor Documentation	902
6.227.3 Member Function Documentation	902
6.227.4 Member Data Documentation	902
6.228PathPair Class Reference	903
6.228.1 Detailed Description	904
6.228.2 Constructor & Destructor Documentation	904
6.228.3 Member Function Documentation	905
6.228.4 Member Data Documentation	905
6.229PathPairs Class Reference	906

6.229.1 Detailed Description	907
6.229.2 Constructor & Destructor Documentation	907
6.229.3 Member Function Documentation	907
6.229.4 Member Data Documentation	909
6.230PolarCone Class Reference	909
6.230.1 Detailed Description	911
6.230.2 Constructor & Destructor Documentation	911
6.230.3 Member Function Documentation	911
6.230.4 Member Data Documentation	912
6.231PolyhedralCone Class Reference	913
6.231.1 Detailed Description	914
6.231.2 Constructor & Destructor Documentation	914
6.231.3 Member Function Documentation	915
6.231.4 Member Data Documentation	915
6.232Processes Class Reference	916
6.232.1 Detailed Description	918
6.232.2 Constructor & Destructor Documentation	918
6.232.3 Member Function Documentation	918
6.232.4 Member Data Documentation	919
6.233ProductCone Class Reference	919
6.233.1 Detailed Description	921
6.233.2 Constructor & Destructor Documentation	921
6.233.3 Member Function Documentation	921
6.233.4 Member Data Documentation	922
6.234QuadraticCoefficients Class Reference	923
6.234.1 Detailed Description	924
6.234.2 Constructor & Destructor Documentation	924
6.234.3 Member Function Documentation	924
6.234.4 Member Data Documentation	924
6.235QuadraticCone Class Reference	925
6.235.1 Detailed Description	926
6.235.2 Constructor & Destructor Documentation	926
6.235.3 Member Function Documentation	927
6.235.4 Member Data Documentation	928
6.236QuadraticTerm Class Reference	929
6.236.1 Detailed Description	929
6.236.2 Constructor & Destructor Documentation	929

6.236.3 Member Function Documentation	930
6.236.4 Member Data Documentation	930
6.237 QuadraticTerms Class Reference	930
6.237.1 Detailed Description	931
6.237.2 Constructor & Destructor Documentation	931
6.237.3 Member Data Documentation	931
6.238 RotatedQuadraticCone Class Reference	931
6.238.1 Detailed Description	933
6.238.2 Constructor & Destructor Documentation	933
6.238.3 Member Function Documentation	934
6.238.4 Member Data Documentation	934
6.239 RowReferenceMatrixElements Class Reference	936
6.239.1 Detailed Description	937
6.239.2 Constructor & Destructor Documentation	938
6.239.3 Member Function Documentation	938
6.239.4 Member Data Documentation	939
6.240 ScalarExpressionTree Class Reference	940
6.240.1 Detailed Description	942
6.240.2 Constructor & Destructor Documentation	942
6.240.3 Member Function Documentation	942
6.240.4 Member Data Documentation	943
6.241 SemidefiniteCone Class Reference	943
6.241.1 Detailed Description	945
6.241.2 Constructor & Destructor Documentation	945
6.241.3 Member Function Documentation	945
6.241.4 Member Data Documentation	946
6.242 ServiceOption Class Reference	947
6.242.1 Detailed Description	949
6.242.2 Constructor & Destructor Documentation	949
6.242.3 Member Function Documentation	949
6.242.4 Member Data Documentation	950
6.243 ServiceResult Class Reference	950
6.243.1 Detailed Description	952
6.243.2 Constructor & Destructor Documentation	952
6.243.3 Member Function Documentation	952
6.243.4 Member Data Documentation	953
6.244 SolverOption Class Reference	953

6.244.1 Detailed Description	955
6.244.2 Constructor & Destructor Documentation	955
6.244.3 Member Function Documentation	955
6.244.4 Member Data Documentation	956
6.245 SolverOptions Class Reference	957
6.245.1 Detailed Description	959
6.245.2 Constructor & Destructor Documentation	959
6.245.3 Member Function Documentation	959
6.245.4 Member Data Documentation	960
6.246 SolverOutput Class Reference	961
6.246.1 Detailed Description	962
6.246.2 Constructor & Destructor Documentation	962
6.246.3 Member Function Documentation	962
6.246.4 Member Data Documentation	963
6.247 SOSVariableBranchingWeights Class Reference	963
6.247.1 Detailed Description	965
6.247.2 Constructor & Destructor Documentation	965
6.247.3 Member Function Documentation	965
6.247.4 Member Data Documentation	966
6.248 SOSWeights Class Reference	966
6.248.1 Detailed Description	968
6.248.2 Constructor & Destructor Documentation	968
6.248.3 Member Function Documentation	968
6.248.4 Member Data Documentation	969
6.249 SparseHessianMatrix Class Reference	970
6.249.1 Detailed Description	970
6.249.2 Constructor & Destructor Documentation	971
6.249.3 Member Data Documentation	971
6.250 SparseIntVector Class Reference	972
6.250.1 Detailed Description	972
6.250.2 Constructor & Destructor Documentation	972
6.250.3 Member Data Documentation	973
6.251 SparseJacobianMatrix Class Reference	973
6.251.1 Detailed Description	974
6.251.2 Constructor & Destructor Documentation	974
6.251.3 Member Data Documentation	974
6.252 SparseMatrix Class Reference	975

6.252.1 Detailed Description	976
6.252.2 Constructor & Destructor Documentation	976
6.252.3 Member Function Documentation	976
6.252.4 Member Data Documentation	977
6.253SparseVector Class Reference	977
6.253.1 Detailed Description	978
6.253.2 Constructor & Destructor Documentation	978
6.253.3 Member Data Documentation	978
6.254StorageCapacity Class Reference	979
6.254.1 Detailed Description	980
6.254.2 Constructor & Destructor Documentation	980
6.254.3 Member Function Documentation	980
6.254.4 Member Data Documentation	981
6.255SystemOption Class Reference	981
6.255.1 Detailed Description	983
6.255.2 Constructor & Destructor Documentation	983
6.255.3 Member Function Documentation	983
6.255.4 Member Data Documentation	984
6.256SystemResult Class Reference	984
6.256.1 Detailed Description	986
6.256.2 Constructor & Destructor Documentation	986
6.256.3 Member Function Documentation	986
6.256.4 Member Data Documentation	986
6.257TimeDomain Class Reference	987
6.257.1 Detailed Description	988
6.257.2 Constructor & Destructor Documentation	988
6.257.3 Member Data Documentation	988
6.258TimeDomainInterval Class Reference	988
6.258.1 Detailed Description	989
6.258.2 Constructor & Destructor Documentation	989
6.258.3 Member Data Documentation	989
6.259TimeDomainStage Class Reference	989
6.259.1 Detailed Description	990
6.259.2 Constructor & Destructor Documentation	990
6.259.3 Member Data Documentation	991
6.260TimeDomainStageCon Class Reference	991
6.260.1 Detailed Description	991

6.260.2 Constructor & Destructor Documentation	992
6.260.3 Member Data Documentation	992
6.261 TimeDomainStageConstraints Class Reference	992
6.261.1 Detailed Description	993
6.261.2 Constructor & Destructor Documentation	993
6.261.3 Member Data Documentation	993
6.262 TimeDomainStageObj Class Reference	993
6.262.1 Detailed Description	994
6.262.2 Constructor & Destructor Documentation	994
6.262.3 Member Data Documentation	994
6.263 TimeDomainStageObjectives Class Reference	994
6.263.1 Detailed Description	995
6.263.2 Constructor & Destructor Documentation	995
6.263.3 Member Data Documentation	996
6.264 TimeDomainStages Class Reference	996
6.264.1 Detailed Description	997
6.264.2 Constructor & Destructor Documentation	997
6.264.3 Member Data Documentation	997
6.265 TimeDomainStageVar Class Reference	997
6.265.1 Detailed Description	998
6.265.2 Constructor & Destructor Documentation	998
6.265.3 Member Data Documentation	998
6.266 TimeDomainStageVariables Class Reference	998
6.266.1 Detailed Description	999
6.266.2 Constructor & Destructor Documentation	999
6.266.3 Member Data Documentation	999
6.267 TimeMeasurement Class Reference	1000
6.267.1 Detailed Description	1001
6.267.2 Constructor & Destructor Documentation	1001
6.267.3 Member Function Documentation	1001
6.267.4 Member Data Documentation	1002
6.268 TimeSpan Class Reference	1002
6.268.1 Detailed Description	1004
6.268.2 Constructor & Destructor Documentation	1004
6.268.3 Member Function Documentation	1004
6.268.4 Member Data Documentation	1005
6.269 TimingInformation Class Reference	1005

6.269.1 Detailed Description	1007
6.269.2 Constructor & Destructor Documentation	1007
6.269.3 Member Function Documentation	1007
6.269.4 Member Data Documentation	1008
6.270Variable Class Reference	1008
6.270.1 Detailed Description	1009
6.270.2 Constructor & Destructor Documentation	1009
6.270.3 Member Function Documentation	1009
6.270.4 Member Data Documentation	1009
6.271VariableOption Class Reference	1010
6.271.1 Detailed Description	1011
6.271.2 Constructor & Destructor Documentation	1011
6.271.3 Member Function Documentation	1012
6.271.4 Member Data Documentation	1012
6.272Variables Class Reference	1013
6.272.1 Detailed Description	1014
6.272.2 Constructor & Destructor Documentation	1015
6.272.3 Member Function Documentation	1015
6.272.4 Member Data Documentation	1015
6.273VariableSolution Class Reference	1015
6.273.1 Detailed Description	1017
6.273.2 Constructor & Destructor Documentation	1017
6.273.3 Member Function Documentation	1017
6.273.4 Member Data Documentation	1017
6.274VariableValues Class Reference	1018
6.274.1 Detailed Description	1020
6.274.2 Constructor & Destructor Documentation	1020
6.274.3 Member Function Documentation	1020
6.274.4 Member Data Documentation	1020
6.275VariableValuesString Class Reference	1021
6.275.1 Detailed Description	1022
6.275.2 Constructor & Destructor Documentation	1022
6.275.3 Member Function Documentation	1022
6.275.4 Member Data Documentation	1023
6.276VarReferenceMatrixElements Class Reference	1023
6.276.1 Detailed Description	1025
6.276.2 Constructor & Destructor Documentation	1025

6.276.3 Member Function Documentation	1025
6.276.4 Member Data Documentation	1027
6.277VarReferenceMatrixValues Class Reference	1027
6.277.1 Detailed Description	1029
6.277.2 Constructor & Destructor Documentation	1029
6.277.3 Member Function Documentation	1029
6.277.4 Member Data Documentation	1029
6.278VarValue Class Reference	1030
6.278.1 Detailed Description	1031
6.278.2 Constructor & Destructor Documentation	1031
6.278.3 Member Function Documentation	1031
6.278.4 Member Data Documentation	1031
6.279VarValueString Class Reference	1032
6.279.1 Detailed Description	1033
6.279.2 Constructor & Destructor Documentation	1033
6.279.3 Member Function Documentation	1033
6.279.4 Member Data Documentation	1034
6.280WSUtil Class Reference	1034
6.280.1 Detailed Description	1035
6.280.2 Constructor & Destructor Documentation	1035
6.280.3 Member Function Documentation	1035
6.281YYLTYPE Struct Reference	1037
6.281.1 Detailed Description	1038
6.281.2 Member Data Documentation	1038
6.282YYSTYPE Union Reference	1038
6.282.1 Detailed Description	1038
6.282.2 Member Data Documentation	1038
7 File Documentation	1039
7.1 /home/ted/COIN/trunk/OS/src/config_default.h File Reference	1039
7.1.1 Macro Definition Documentation	1039
7.2 /home/ted/COIN/trunk/OS/src/config_os_default.h File Reference	1040
7.2.1 Macro Definition Documentation	1041
7.3 /home/ted/COIN/trunk/OS/src/OSAgent/OShL.h File Reference	1042
7.3.1 Detailed Description	1043
7.4 /home/ted/COIN/trunk/OS/src/OSAgent/OSSolverAgent.h File Reference	1043
7.4.1 Detailed Description	1044

7.5	/home/ted/COIN/trunk/OS/src/OSAgent/OSWSUtil.h File Reference	1044
7.5.1	Detailed Description	1045
7.5.2	Macro Definition Documentation	1045
7.6	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLine.h File Reference	1046
7.6.1	Detailed Description	1047
7.7	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLineReader.h File Reference	1047
7.7.1	Detailed Description	1047
7.8	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSDefaultSolver.h File Reference	1048
7.9	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTree.h File Reference	1048
7.9.1	Detailed Description	1049
7.10	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h File Reference	1050
7.10.1	Detailed Description	1051
7.10.2	Function Documentation	1052
7.11	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGLWriter.h File Reference	1052
7.11.1	Detailed Description	1053
7.11.2	Function Documentation	1053
7.12	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSILReader.h File Reference	1055
7.12.1	Detailed Description	1055
7.13	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSILWriter.h File Reference	1056
7.13.1	Detailed Description	1057
7.14	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h File Reference	1057
7.14.1	Detailed Description	1060
7.15	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h File Reference	1060
7.15.1	Detailed Description	1063
7.16	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h File Reference	1063
7.16.1	Detailed Description	1066
7.16.2	Typedef Documentation	1066
7.17	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLReader.h File Reference	1066
7.17.1	Detailed Description	1067
7.18	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLWriter.h File Reference	1068
7.18.1	Detailed Description	1068
7.19	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h File Reference	1069
7.19.1	Detailed Description	1072
7.20	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h File Reference	1072
7.20.1	Detailed Description	1074
7.21	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLReader.h File Reference	1075
7.21.1	Detailed Description	1076

7.22	/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLWriter.h File Reference	1076
7.22.1	Detailed Description	1077
7.23	/home/ted/COIN/trunk/OS/src/OSConfig.h File Reference	1077
7.24	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSgams2osil.hpp File Reference	1078
7.25	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSmps2OS.h File Reference	1078
7.25.1	Detailed Description	1079
7.26	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSmps2osil.h File Reference	1079
7.26.1	Detailed Description	1080
7.27	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSnl2OS.h File Reference	1080
7.27.1	Enumeration Type Documentation	1081
7.28	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosl2ampl.h File Reference	1082
7.28.1	Detailed Description	1082
7.29	/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosl2gams.hpp File Reference	1083
7.30	/home/ted/COIN/trunk/OS/src/OSParsers/OSgLParserData.h File Reference	1083
7.30.1	Detailed Description	1084
7.30.2	Function Documentation	1085
7.31	/home/ted/COIN/trunk/OS/src/OSParsers/OSiLParserData.h File Reference	1085
7.31.1	Detailed Description	1086
7.32	/home/ted/COIN/trunk/OS/src/OSParsers/OSnLParserData.h File Reference	1086
7.32.1	Detailed Description	1087
7.32.2	Function Documentation	1088
7.33	/home/ted/COIN/trunk/OS/src/OSParsers/OSoLParserData.h File Reference	1088
7.33.1	Detailed Description	1089
7.34	/home/ted/COIN/trunk/OS/src/OSParsers/OSOptionsStruc.h File Reference	1089
7.34.1	Detailed Description	1090
7.35	/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosil.tab.hpp File Reference	1090
7.35.1	Macro Definition Documentation	1106
7.35.2	Typedef Documentation	1130
7.35.3	Enumeration Type Documentation	1130
7.36	/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosol.tab.hpp File Reference	1158
7.36.1	Macro Definition Documentation	1175
7.36.2	Typedef Documentation	1205
7.36.3	Enumeration Type Documentation	1205
7.37	/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosrl.tab.hpp File Reference	1233
7.37.1	Macro Definition Documentation	1249
7.37.2	Typedef Documentation	1278
7.37.3	Enumeration Type Documentation	1278

7.38	/home/ted/COIN/trunk/OS/src/OSParsers/OSrLParserData.h File Reference	1305
7.38.1	Detailed Description	1307
7.39	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSBonminSolver.h File Reference	1308
7.40	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCoinSolver.h File Reference	1309
7.41	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCouenneSolver.h File Reference	1310
7.41.1	Detailed Description	1311
7.42	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCsdpSolver.h File Reference	1311
7.42.1	Detailed Description	1312
7.43	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSIpoptSolver.h File Reference	1313
7.44	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSKnitroSolver.h File Reference	1314
7.45	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSLindoSolver.h File Reference	1315
7.46	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSMatlabSolver.h File Reference	1316
7.47	/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSRunSolver.h File Reference	1316
7.47.1	Detailed Description	1317
7.47.2	Function Documentation	1318
7.48	/home/ted/COIN/trunk/OS/src/OSUtils/OSBase64.h File Reference	1319
7.49	/home/ted/COIN/trunk/OS/src/OSUtils/OSDataStructures.h File Reference	1320
7.50	/home/ted/COIN/trunk/OS/src/OSUtils/OSdtoa.h File Reference	1321
7.50.1	Function Documentation	1321
7.51	/home/ted/COIN/trunk/OS/src/OSUtils/OSErrorClass.h File Reference	1322
7.52	/home/ted/COIN/trunk/OS/src/OSUtils/OSFileUtil.h File Reference	1322
7.53	/home/ted/COIN/trunk/OS/src/OSUtils/OSMathUtil.h File Reference	1323
7.53.1	Function Documentation	1325
7.54	/home/ted/COIN/trunk/OS/src/OSUtils/OSOutput.h File Reference	1326
7.54.1	Detailed Description	1327
7.54.2	Variable Documentation	1327
7.55	/home/ted/COIN/trunk/OS/src/OSUtils/OSParameters.h File Reference	1328
7.55.1	Detailed Description	1333
7.55.2	Macro Definition Documentation	1334
7.55.3	Enumeration Type Documentation	1337
7.55.4	Function Documentation	1345
7.55.5	Variable Documentation	1349
7.56	/home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.hpp File Reference	1349
7.57	/home/ted/COIN/trunk/OS/src/OSUtils/OSSmartPtr.hpp File Reference	1350
7.57.1	Function Documentation	1352
7.58	/home/ted/COIN/trunk/OS/src/OSUtils/OSStringUtil.h File Reference	1353
7.58.1	Detailed Description	1353

7.58.2 Function Documentation	1354
---	------

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Coin	30
Couenne	30

2 Class Index

2.1 Class Hierarchy

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

<code>_EKKfactinfo</code> [external]	
<code>doubleton_action::action</code> [external]	
<code>forcing_constraint_action::action</code> [external]	
<code>remove_fixed_action::action</code> [external]	
<code>tripleton_action::action</code> [external]	
Base64	30
<code>std::basic_fstream< char ></code>	
<code>std::basic_fstream< wchar_t ></code>	
<code>std::basic_ifstream< char ></code>	
<code>std::basic_ifstream< wchar_t ></code>	
<code>std::basic_ios< char ></code>	
<code>std::basic_ios< wchar_t ></code>	
<code>std::basic_iostream< char ></code>	
<code>std::basic_iostream< wchar_t ></code>	
<code>std::basic_istream< char ></code>	
<code>std::basic_istream< wchar_t ></code>	
<code>std::basic_istreamstream< char ></code>	
<code>std::basic_istreamstream< wchar_t ></code>	
<code>std::basic_ofstream< char ></code>	
<code>std::basic_ofstream< wchar_t ></code>	
<code>std::basic_ostream< char ></code>	
<code>std::basic_ostream< wchar_t ></code>	
<code>std::basic_ostreamstream< char ></code>	
<code>std::basic_ostreamstream< wchar_t ></code>	
<code>std::basic_string< char ></code>	
<code>std::basic_string< wchar_t ></code>	
<code>std::basic_stringstream< char ></code>	
<code>std::basic_stringstream< wchar_t ></code>	
BasisStatus	36
<code>BitVector128</code> [external]	
BonminProblem	41

BranchingWeight

50

```
CoinAbsFltEq [external]
CoinArrayWithLength [external]
  CoinArbitraryArrayWithLength [external]
  CoinBigIndexArrayWithLength [external]
  CoinDoubleArrayWithLength [external]
  CoinFactorizationDoubleArrayWithLength [external]
  CoinFactorizationLongDoubleArrayWithLength [external]
  CoinIntArrayWithLength [external]
  CoinUnsignedIntArrayWithLength [external]
  CoinVoidStarArrayWithLength [external]
CoinBaseModel [external]
  CoinModel [external]
  CoinStructuredModel [external]
CoinBuild [external]
CoinDenseVector< T > [external]
CoinError [external]
CoinExternalVectorFirstGreater_2< class, class, class > [external]
CoinExternalVectorFirstGreater_3< class, class, class, class > [external]
CoinExternalVectorFirstLess_2< class, class, class > [external]
CoinExternalVectorFirstLess_3< class, class, class, class > [external]
CoinFactorization [external]
CoinFileIOBase [external]
  CoinFileInput [external]
  CoinFileOutput [external]
CoinFirstAbsGreater_2< class, class > [external]
CoinFirstAbsGreater_3< class, class, class > [external]
CoinFirstAbsLess_2< class, class > [external]
CoinFirstAbsLess_3< class, class, class > [external]
CoinFirstGreater_2< class, class > [external]
CoinFirstGreater_3< class, class, class > [external]
CoinFirstLess_2< class, class > [external]
CoinFirstLess_3< class, class, class > [external]
CoinLpIO::CoinHashLink [external]
CoinMpsIO::CoinHashLink [external]
CoinIndexedVector [external]
  CoinPartitionedVector [external]
CoinLpIO [external]
CoinMessageHandler [external]
CoinMessages [external]
  CoinMessage [external]
CoinModelHash [external]
CoinModelHash2 [external]
CoinModelHashLink [external]
CoinModelInfo2 [external]
CoinModelLink [external]
CoinModelLinkedList [external]
CoinModelTriple [external]
CoinMpsCardReader [external]
CoinMpsIO [external]
CoinOneMessage [external]
CoinOtherFactorization [external]
  CoinDenseFactorization [external]
  CoinOslFactorization [external]
```

- CoinSimpFactorization [external]
- CoinPackedMatrix [external]
- CoinPackedVectorBase [external]
 - CoinPackedVector [external]
 - CoinShallowPackedVector [external]
- CoinPair< S, T > [external]
- CoinParam [external]
- CoinPrePostsolveMatrix [external]
 - CoinPostsolveMatrix [external]
 - CoinPresolveMatrix [external]
- CoinPresolveAction [external]
 - do_tighten_action [external]
 - doubleton_action [external]
 - drop_empty_cols_action [external]
 - drop_empty_rows_action [external]
 - drop_zero_coefficients_action [external]
 - dupcol_action [external]
 - duprow_action [external]
 - forcing_constraint_action [external]
 - gubrow_action [external]
 - implied_free_action [external]
 - isolated_constraint_action [external]
 - make_fixed_action [external]
 - remove_dual_action [external]
 - remove_fixed_action [external]
 - slack_doubleton_action [external]
 - slack_singleton_action [external]
 - subst_constraint_action [external]
 - tripleton_action [external]
 - twoxtwo_action [external]
 - useless_constraint_action [external]
- CoinPresolveMonitor [external]
- CoinRational [external]
- CoinRelFltEq [external]
- CoinSearchTreeBase [external]
 - CoinSearchTree< class > [external]
- CoinSearchTreeCompareBest [external]
- CoinSearchTreeCompareBreadth [external]
- CoinSearchTreeCompareDepth [external]
- CoinSearchTreeComparePreferred [external]
- CoinSearchTreeManager [external]
- CoinSet [external]
 - CoinSosSet [external]
- CoinSnapshot [external]
- CoinThreadRandom [external]
- CoinTimer [external]
- CoinTreeNode [external]
- CoinTreeSiblings [external]
- CoinTriple< S, T, U > [external]
- CoinWarmStart [external]
 - CoinWarmStartBasis [external]
 - CoinWarmStartDual [external]
 - CoinWarmStartPrimalDual [external]
 - CoinWarmStartVector< T > [external]

CoinWarmStartVectorPair< T, U >[external]	
CoinWarmStartDiff[external]	
CoinWarmStartBasisDiff[external]	
CoinWarmStartDualDiff[external]	
CoinWarmStartPrimalDualDiff[external]	
CoinWarmStartVectorDiff< T >[external]	
CoinWarmStartVectorPairDiff< T, U >[external]	
CoinYacc[external]	
Cone	60
CompletelyPositiveMatricesCone	57
CopositiveMatricesCone	93
DualCone	118
IntersectionCone	222
NonnegativeCone	345
NonpositiveCone	347
OrthantCone	388
PolarCone	909
PolyhedralCone	913
ProductCone	919
QuadraticCone	925
RotatedQuadraticCone	931
SemidefiniteCone	943
Cones	65
ConReferenceMatrixElement	67
Constraint	81
ConstraintOption	83
Constraints	86
ConstraintSolution	88
ContactOption	90
CPUNumber	100
CPU Speed	103
DefaultSolver	110
BonminSolver	46

CoinSolver	52
CouenneSolver	96
CsdpSolver	105
IpoptSolver	233
KnitroSolver	251
LindoSolver	254
DirectoriesAndFiles	114
DoubleVector	117
dropped_zero[external]	
DualVariableValues	122
DualVarValue	124
EKKHlink[external]	
ErrorClass	127
ExpandedMatrixBlocks	128
ExprNode	132
OSnLMNode	504
OSnLMNodeDiagonalMatrixFromVector	509
OSnLMNodeIdentityMatrix	511
OSnLMNodeMatrixCon	513
OSnLMNodeMatrixDiagonal	516
OSnLMNodeMatrixDotTimes	517
OSnLMNodeMatrixInverse	519
OSnLMNodeMatrixLowerTriangle	521
OSnLMNodeMatrixMinus	524
OSnLMNodeMatrixNegate	526
OSnLMNodeMatrixObj	528
OSnLMNodeMatrixPlus	531
OSnLMNodeMatrixProduct	532
OSnLMNodeMatrixReference	535
OSnLMNodeMatrixScalarTimes	538
OSnLMNodeMatrixSubmatrixAt	539

OSnLMNodeMatrixSum	541
OSnLMNodeMatrixTimes	543
OSnLMNodeMatrixTranspose	545
OSnLMNodeMatrixUpperTriangle	547
OSnLMNodeMatrixVar	550
OSnLNode	553
OSnLNodeAbs	559
OSnLNodeAllDiff	562
OSnLNodeCos	565
OSnLNodeDivide	568
OSnLNodeE	571
OSnLNodeErf	575
OSnLNodeExp	577
OSnLNodeIf	580
OSnLNodeLn	583
OSnLNodeMatrixDeterminant	586
OSnLNodeMatrixToScalar	589
OSnLNodeMatrixTrace	592
OSnLNodeMax	595
OSnLNodeMin	598
OSnLNodeMinus	601
OSnLNodeNegate	604
OSnLNodeNumber	607
OSnLNodePI	611
OSnLNodePlus	615
OSnLNodePower	617
OSnLNodeProduct	620
OSnLNodeSin	623
OSnLNodeSqrt	626
OSnLNodeSquare	629

OSnLNodeSum	632
OSnLNodeTimes	635
OSnLNodeVariable	638
FactorPointers[external]	
FileUtil	137
GeneralFileHeader	140
GeneralOption	149
GeneralResult	153
GeneralSparseMatrix	157
GeneralStatus	160
GeneralSubstatus	162
IndexStringPair	165
IndexValuePair	166
InitBasStatus	166
InitConstraintValues	169
InitConValue	173
InitDualVariableValues	176
InitDualVarValue	179
InitialBasisStatus	182
InitObjBound	185
InitObjectiveBounds	188
InitObjectiveValues	192
InitObjValue	196
InitVariableValues	199
InitVariableValuesString	203
InitVarValue	207
InitVarValueString	210
InstanceData	213
InstanceLocationOption	216
IntegerVariableBranchingWeights	218

Interval	226
IntVector	226
OtherOptionEnumeration	870
IpoptProblem	230
JobDependencies	238
JobOption	240
JobResult	244
KnitroProblem	248
LinearConstraintCoefficients	259
LinearMatrixElement	261
LinearMatrixElementTerm	267
MathUtil	271
Matrices	273
MatrixCon	283
MatrixConstraints	286
MatrixElementValues	292
ConReferenceMatrixValues	73
ConstantMatrixValues	79
GeneralMatrixValues	147
LinearMatrixValues	269
ObjReferenceMatrixValues	369
VarReferenceMatrixValues	1027
MatrixExpression	294
MatrixExpressions	296
MatrixNode	302
MatrixConstructor	288
BaseMatrix	32
MatrixBlocks	279
MatrixElements	290
ConReferenceMatrixElements	69

ConstantMatrixElements	75
GeneralMatrixElements	143
LinearMatrixElements	263
ObjReferenceMatrixElements	365
RowReferenceMatrixElements	936
VarReferenceMatrixElements	1023
MatrixTransformation	314
MatrixType	318
MatrixBlock	275
OSMatrix	489
MatrixObj	307
MatrixObjectives	309
MatrixProgramming	311
MatrixVar	323
MatrixVariables	326
MaxTime	328
MinCPUNumber	330
MinCPUSpeed	332
MinDiskSpace	335
MinMemorySize	337
NI	339
NonlinearExpressions	342
ObjCoef	350
Objective	351
ObjectiveOption	354
Objectives	357
ObjectiveSolution	359
ObjectiveValues	362
ObjValue	371
OptimizationOption	373

OptimizationResult	377
OptimizationSolution	379
OptimizationSolutionStatus	383
OptimizationSolutionSubstatus	385
OS_AMPL_SUFFIX	391
OSCommandLine	392
OSCommandLineReader	399
OSExpressionTree	401
MatrixExpressionTree	298
ScalarExpressionTree	940
OSgams2osil	404
OSGeneral	405
OSgLPParserData	406
OShL	416
OSSolverAgent	836
OSiLParserData	419
OSiLReader	431
OSiLWriter	432
OSInstance	434
OSMatlab	483
OSmps2OS	494
OSmps2osil	497
OSnl2OS	499
OSnLParserData	642
OSoLParserData	652
OSoLReader	671
OSoLWriter	673
OSOption	675
osOptionsStruc	714
OSosrl2AMPL	720

OSOutputChannel	725
OSReferencedObject	727
OSOutput	721
OSReferencer	730
OSSmartPtr< T >	830
OSResult	730
OSrL2Gams	811
OSrLParserData	812
OSrLReader	827
OSrLWriter	828
OtherConOption	840
OtherConResult	843
OtherConstraintOption	845
OtherConstraintResult	850
OtherObjectiveOption	854
OtherObjectiveResult	858
OtherObjOption	862
OtherObjResult	865
OtherOption	867
OtherOptions	873
OtherResult	876
OtherResults	878
OtherSolutionResult	881
OtherSolutionResults	884
OtherSolverOutput	886
OtherVariableOption	888
OtherVariableResult	893
OtherVariableResultStruct	896
OtherVarOption	898
OtherVarResult	901

PathPair	903
PathPairs	906
<code>presolvehlink[external]</code>	
Processes	916
QuadraticCoefficients	923
QuadraticTerm	929
QuadraticTerms	930
<code>ReferencedObject[external]</code>	
ServiceOption	947
ServiceResult	950
<code>SmartPtr< T >[external]</code>	
SolverOption	953
SolverOptions	957
SolverOutput	961
SOSVariableBranchingWeights	963
SOSWeights	966
SparseHessianMatrix	970
SparseIntVector	972
SparseJacobianMatrix	973
SparseMatrix	975
SparseVector	977
StorageCapacity	979
<code>symrec[external]</code>	
SystemOption	981
SystemResult	984
TimeDomain	987
TimeDomainInterval	988
TimeDomainStage	989
TimeDomainStageCon	991
TimeDomainStageConstraints	992
TimeDomainStageObj	993

TimeDomainStageObjectives	994
TimeDomainStages	996
TimeDomainStageVar	997
TimeDomainStageVariables	998
TimeSpan	1002
TimeMeasurement	1000
TimingInformation	1005
Variable	1008
VariableOption	1010
Variables	1013
VariableSolution	1015
VariableValues	1018
VariableValuesString	1021
VarValue	1030
VarValueString	1032
WSUtil	1034
YYLTYPE	1037
YYSTYPE	1038

3 Class Index

3.1 Class List

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

Base64	
Use this class to read and write data in base64	30
BaseMatrix	
Data structure to represent a point of departure for constructing a matrix by modifying parts of a previously defined matrix	32
BasisStatus	
Data structure to represent an LP basis on both input and output	36
BonminProblem	41
BonminSolver	
Solves problems using Ipopt	46

BranchingWeight BranchingWeight class	50
CoinSolver Implements a solve method for the Coin solvers	52
CompletelyPositiveMatricesCone The CompletelyPositiveMatricesCone Class	57
Cone The in-memory representation of a generic cone Specific cone types are derived from this generic class	60
Cones The in-memory representation of the <cones> element	65
ConReferenceMatrixElement Data structure to represent an entry in a conReferenceMatrix element, which consists of a constraint reference as well as a value type	67
ConReferenceMatrixElements Data structure to represent row reference elements in a MatrixType object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a constraint	69
ConReferenceMatrixValues Data structure to represent the nonzeros in a conReferenceMatrix element	73
ConstantMatrixElements Data structure to represent the constant elements in a MatrixType object	75
ConstantMatrixValues To represent the nonzeros in a constantMatrix element	79
Constraint The in-memory representation of the <con> element	81
ConstraintOption ConstraintOption class	83
Constraints The in-memory representation of the <constraints> element	86
ConstraintSolution The ConstraintSolution Class	88
ContactOption ContactOption class	90
CopositiveMatricesCone The CopositiveMatricesCone Class	93
CouenneSolver Solves problems using Ipopt	96
CPUNumber CPUNumber class	100

CPUSpeed	
CPUSpeed class	103
CsdpSolver	
Solves problems using Csdp	105
DefaultSolver	
The Default Solver Class	110
DirectoriesAndFiles	
DirectoriesAndFiles class	114
DoubleVector	
Double vector data structure	117
DualCone	
The in-memory representation of a dual cone	118
DualVariableValues	
The DualVariableValues Class	122
DualVarValue	
The DualVarValue Class	124
ErrorClass	
Used for throwing exceptions	127
ExpandedMatrixBlocks	
Sparse matrix data structure for matrices that can hold nonconstant values and have block structure In addition it is assumed that all nesting of blocks has been resolved	128
ExprNode	
A generic class from which we derive both OSnLNode and OSnLMNode	132
FileUtil	
Class used to make it easy to read and write files	137
GeneralFileHeader	
Data structure that holds general information about files that conform to one of the OSxL schemas	140
GeneralMatrixElements	
Data structure to represent the nonzero values in a generalMatrix element	143
GeneralMatrixValues	
Data structure to represent the nonzeros in a generalMatrix element	147
GeneralOption	
The GeneralOption Class	149
GeneralResult	
The GeneralResult Class	153
GeneralSparseMatrix	
Sparse matrix data structure for matrices that can hold nonconstant values	157
GeneralStatus	
The GeneralStatus Class	160

GeneralSubstatus	
The GeneralSubstatus Class	162
IndexStringPair	
A commonly used structure holding an index-string pair This definition is based on the definition of IndexValuePair in OSGeneral.h	165
IndexValuePair	
A commonly used structure holding an index-value pair	166
InitBasStatus	
InitBasStatus class	166
InitConstraintValues	
InitConstraintValues class	169
InitConValue	
InitConValue class	173
InitDualVariableValues	
InitDualVariableValues class	176
InitDualVarValue	
InitDualVarValue class	179
InitialBasisStatus	
InitialBasisStatus class	182
InitObjBound	
InitObjBound class	185
InitObjectiveBounds	
InitObjectiveBounds class	188
InitObjectiveValues	
InitObjectiveValues class	192
InitObjValue	
InitObjValue class	196
InitVariableValues	
InitVariableValues class	199
InitVariableValuesString	
InitVariableValuesString class	203
InitVarValue	
InitVarValue class	207
InitVarValueString	
InitVarValueString class	210
InstanceData	
The in-memory representation of the <instanceData> element	213
InstanceLocationOption	
InstanceLocationOption class	216

IntegerVariableBranchingWeights	
IntegerVariableBranchingWeights class	218
IntersectionCone	
The in-memory representation of an intersection cone	222
Interval	
The in-memory representation of the <interval> element	226
IntVector	
Integer Vector data structure	226
IpoptProblem	230
IpoptSolver	
Solves problems using Ipopt	233
JobDependencies	
JobDependencies class	238
JobOption	
JobOption class	240
JobResult	
The JobResult Class	244
KnitroProblem	248
KnitroSolver	
KnitroSolver class solves problems using Knitro	251
LindoSolver	
LindoSolver class solves problems using Lindo	254
LinearConstraintCoefficients	
The in-memory representation of the <linearConstraintCoefficients> element	259
LinearMatrixElement	
Data structure to represent an expression in a linearMatrix element A LinearMatrixElement is a (finite) sum of LinearMatrixElementTerms , with an optional additive constant	261
LinearMatrixElements	
Data structure to represent the nonzero values in a linearMatrix element	263
LinearMatrixElementTerm	
Data structure to represent a term in a linearMatrix element A term has the form $c \cdot x_{\{k\}}$, where c defaults to 1 and k is a valid index for a variable This is essentially an index-value pair, but with the presence of a default value	267
LinearMatrixValues	
Data structure to represent the linear expressions in a LinearMatrixElement object	269
MathUtil	
This class has routines for linear algebra	271
Matrices	
The in-memory representation of the <matrices> element	273

MatrixBlock	
Data structure to represent a MatrixBlock object (derived from MatrixType)	275
MatrixBlocks	
Data structure to represent the nonzeros of a matrix in a blockwise fashion	279
MatrixCon	
The in-memory representation of the <matrixCon> element	283
MatrixConstraints	
The in-memory representation of the <matrixConstraints> element	286
MatrixConstructor	
Data structure to describe one step in the construction of a matrix	288
MatrixElements	
Abstract class to help represent the elements in a MatrixType object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc	290
MatrixElementValues	
Abstract class to help represent the elements in a MatrixType object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc	292
MatrixExpression	
The in-memory representation of the <expr> element, which is like a nonlinear expression, but since it involves matrices, the expression could be linear, so a "shape" attribute is added to distinguish linear and nonlinear expressions	294
MatrixExpressions	
The in-memory representation of the <matrixExpressions> element	296
MatrixExpressionTree	
Used to hold the instance in memory	298
MatrixNode	
Generic class from which we derive matrix constructors (BaseMatrix , MatrixElements , MatrixTransformation and MatrixBlocks) as well as matrix types (OSMatrix and MatrixBlock)	302
MatrixObj	
The in-memory representation of the <matrixObj> element	307
MatrixObjectives	
The in-memory representation of the <matrixObjectives> element	309
MatrixProgramming	
The in-memory representation of the <matrixProgramming> element	311
MatrixTransformation	
Data structure to represent the nonzeros of a matrix by transformation from other (previously defined) matrices	314
MatrixType	
Data structure to represent a MatrixType object (from which we derive OSMatrix and MatrixBlock)	318

MatrixVar	
The in-memory representation of the <matrixVar> element	323
MatrixVariables	
The in-memory representation of the <matrixVariables> element	326
MaxTime	
MaxTime class	328
MinCPUNumber	
MinCPUNumber class	330
MinCPUSpeed	
MinCPUSpeed class	332
MinDiskSpace	
MinDiskSpace class	335
MinMemorySize	
MinMemorySize class	337
NI	
The in-memory representation of the <nl> element	339
NonlinearExpressions	
The in-memory representation of the <nonlinearExpressions> element	342
NonnegativeCone	
The NonnegativeCone Class	345
NonpositiveCone	
The NonpositiveCone Class	347
ObjCoef	
The in-memory representation of the objective function <coef> element	350
Objective	
The in-memory representation of the <obj> element	351
ObjectiveOption	
ObjectiveOption class	354
Objectives	
The in-memory representation of the <objectives> element	357
ObjectiveSolution	
The ObjectiveSolution Class	359
ObjectiveValues	
The ObjectiveValues Class	362
ObjReferenceMatrixElements	
Data structure to represent objective reference elements in a MatrixType object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of an objective (i.e., less than zero)	365
ObjReferenceMatrixValues	
To represent the nonzeros in an objReferenceMatrix element	369

ObjValue	
The ObjValue Class	371
OptimizationOption	
OptimizationOption class	373
OptimizationResult	
The OptimizationResult Class	377
OptimizationSolution	
The OptimizationSolution Class	379
OptimizationSolutionStatus	
The OptimizationSolutionStatus Class	383
OptimizationSolutionSubstatus	
The OptimizationSolutionSubstatus Class	385
OrthantCone	
The OrthantCone Class	388
OS_AMPL_SUFFIX	391
OSCommandLine	
This class is used to store command line options for the OSSolverService executable and to provide methods to manipulate them	392
OSCommandLineReader	
The OSCommandLineReader Class	399
OSExpressionTree	
Used to hold the instance in memory	401
OSgams2osil	
Creating a OSInstance from a GAMS model given as GAMS Modeling Object (GMO)	404
OSGeneral	405
OSgLParserData	
The OSgLParserData Class	406
OShL	
An interface that specified virtual methods to be implemented by agents	416
OSiLParserData	
The OSiLParserData Class, used to store parser data	419
OSiLReader	
Used to read an OSiL string	431
OSiLWriter	
Take an OSInstance object and write a string that validates against the OSiL schema	432
OSInstance	
The in-memory representation of an OSiL instance	434

OSMatlab	
The OSMatlab Class	483
OSMatrix	
Data structure to represent a matrix object (derived from MatrixType)	489
OSmps2OS	
The OSmps2OS Class	494
OSmps2osil	
The OSmps2osil Class	497
OSnl2OS	
The OSnl2OS Class	499
OSnLMNode	
The OSnLMNode Class for nonlinear expressions involving matrices	504
OSnLMNodeDiagonalMatrixFromVector	509
OSnLMNodeIdentityMatrix	511
OSnLMNodeMatrixCon	513
OSnLMNodeMatrixDiagonal	516
OSnLMNodeMatrixDotTimes	517
OSnLMNodeMatrixInverse	519
OSnLMNodeMatrixLowerTriangle	521
OSnLMNodeMatrixMinus	524
OSnLMNodeMatrixNegate	526
OSnLMNodeMatrixObj	528
OSnLMNodeMatrixPlus	531
OSnLMNodeMatrixProduct	
The OSnLMNodeMatrixProduct Class	532
OSnLMNodeMatrixReference	535
OSnLMNodeMatrixScalarTimes	538
OSnLMNodeMatrixSubmatrixAt	539
OSnLMNodeMatrixSum	541
OSnLMNodeMatrixTimes	543
OSnLMNodeMatrixTranspose	545
OSnLMNodeMatrixUpperTriangle	547
OSnLMNodeMatrixVar	550

OSnLNode	
The OSnLNode Class for nonlinear expressions	553
OSnLNodeAbs	
The OSnLNodeAbs Class	559
OSnLNodeAllDiff	
The OSnLNodeAllDiff Class	562
OSnLNodeCos	
The OSnLNodeCos Class	565
OSnLNodeDivide	
The OSnLNodeDivide Class	568
OSnLNodeE	
The OSnLNodeE Class	571
OSnLNodeErf	
The OSnLNodeErf Class	575
OSnLNodeExp	
The OSnLNodeExp Class	577
OSnLNodeIf	
The OSnLNodeIf Class	580
OSnLNodeLn	
The OSnLNodeLn Class	583
OSnLNodeMatrixDeterminant	
The next few nodes evaluate to a scalar even though one or more of its arguments are matrices	586
OSnLNodeMatrixToScalar	
The OSnLNodeMatrixTrace Class	589
OSnLNodeMatrixTrace	
The OSnLNodeMatrixTrace Class	592
OSnLNodeMax	
The OSnLNodeMax Class	595
OSnLNodeMin	
The OSnLNodeMin Class	598
OSnLNodeMinus	
The OSnLNodeMinus Class	601
OSnLNodeNegate	
The OSnLNodeNegate Class	604
OSnLNodeNumber	
The OSnLNodeNumber Class	607
OSnLNodePI	
The OSnLNodePI Class	611

OSnLNodePlus	
The OSnLNodePlus Class	615
OSnLNodePower	
The OSnLNodePower Class	617
OSnLNodeProduct	
The OSnLNodeProduct Class	620
OSnLNodeSin	
The OSnLNodeSin Class	623
OSnLNodeSqrt	
The OSnLNodeSqrt Class	626
OSnLNodeSquare	
The OSnLNodeSquare Class	629
OSnLNodeSum	
The OSnLNodeSum Class	632
OSnLNodeTimes	
The OSnLNodeTimes Class	635
OSnLNodeVariable	
The OSnLNodeVariable Class	638
OSnLParserData	
The OSnLParserData Class	642
OSoLParserData	
The OSoLParserData Class	652
OSoLReader	
Used to read an OSoL string	671
OSoLWriter	
Take an OSOption object and write a string that validates against the OSoL schema	673
OSOption	
The Option Class	675
osOptionsStruc	
This structure is used to store options for the OSSolverService executable	714
OSosrl2ampl	
The OSosrl2ampl Class	720
OSOutput	
This class handles all the output from OSSolverService, OSAmplClient and other executables derived from them	721
OSOutputChannel	
Class that holds information about one output channel (file, device, stream, peripheral, etc.)	725
OSReferencedObject	
ReferencedObject class	727

OSReferencer	
Pseudo-class, from which everything has to inherit that wants to use be registered as a Referencer for a ReferencedObject	730
OSResult	
The Result Class	730
OSrL2Gams	
Reads an optimization result and stores result and solution in a Gams Modeling Object	811
OSrLParserData	
The OSrLParserData Class	812
OSrLReader	
The OSrLReader Class	827
OSrLWriter	
Take an OSResult object and write a string that validates against OSrL	828
OSSmartPtr< T >	
Template class for Smart Pointers	830
OSSolverAgent	
Used by a client to invoke a remote solver	836
OtherConOption	
OtherConOption class	840
OtherConResult	
The OtherConResult Class	843
OtherConstraintOption	
OtherConstraintOption class	845
OtherConstraintResult	
The OtherConstraintResult Class	850
OtherObjectiveOption	
OtherObjectiveOption class	854
OtherObjectiveResult	
The OtherObjectiveResult Class	858
OtherObjOption	
OtherObjOption class	862
OtherObjResult	
The OtherObjResult Class	865
OtherOption	
OtherOption class	867
OtherOptionEnumeration	
Brief an integer vector data structure used in OSOption and OSResult	870
OtherOptions	
OtherOptions class	873

OtherResult	
The OtherResult Class	876
OtherResults	
The OtherResults Class	878
OtherSolutionResult	
The OtherSolutionResult Class	881
OtherSolutionResults	
The OtherSolutionResults Class	884
OtherSolverOutput	
The OtherSolverOutput Class	886
OtherVariableOption	
OtherVariableOption class	888
OtherVariableResult	
The OtherVariableResult Class	893
OtherVariableResultStruct	
A structure to information about an OtherVariableResult element	896
OtherVarOption	
OtherVarOption class	898
OtherVarResult	
OtherVarResult Class	901
PathPair	
PathPair class	903
PathPairs	
PathPairs class	906
PolarCone	
The in-memory representation of a polar cone	909
PolyhedralCone	
The in-memory representation of a polyhedral cone	913
Processes	
Processes class	916
ProductCone	
The in-memory representation of a product cone	919
QuadraticCoefficients	
The in-memory representation of the <quadraticCoefficients> element	923
QuadraticCone	
The in-memory representation of a quadratic cone	925
QuadraticTerm	
The in-memory representation of the <qTerm> element	929

QuadraticTerms	
Data structure for holding quadratic terms	930
RotatedQuadraticCone	
The in-memory representation of a rotated quadratic cone	931
RowReferenceMatrixElements	
Data structure to represent row reference elements in a MatrixType object Each nonzero element references a row (if index is negative) or constraint (otherwise) This class allows the combining of row and constraint references in a single matrix constructor	936
ScalarExpressionTree	
Used to hold part of the instance in memory	940
SemidefiniteCone	
The in-memory representation of a cone of semidefinite matrices	943
ServiceOption	
ServiceOption class	947
ServiceResult	
The ServiceResult Class	950
SolverOption	
SolverOption class	953
SolverOptions	
SolverOptions class	957
SolverOutput	
The SolverOutput Class	961
SOSVariableBranchingWeights	
SOSVariableBranchingWeights class	963
SOSWeights	
SOSWeights class	966
SparseHessianMatrix	
The in-memory representation of a SparseHessianMatrix	970
SparseIntVector	
Sparse vector data structure for integer vectors	972
SparseJacobianMatrix	
Sparse Jacobian matrix data structure	973
SparseMatrix	
Sparse matrix data structure	975
SparseVector	
Sparse vector data structure	977
StorageCapacity	
StorageCapacity class	979

SystemOption	
SystemOption class	981
SystemResult	
The SystemResult Class	984
TimeDomain	
The in-memory representation of the <timeDomain> element	987
TimeDomainInterval	988
TimeDomainStage	
The in-memory representation of the <stage> element	989
TimeDomainStageCon	
The in-memory representation of the <con> element	991
TimeDomainStageConstraints	
The in-memory representation of the <constraints> child of the <stage> element	992
TimeDomainStageObj	
The in-memory representation of the <obj> element	993
TimeDomainStageObjectives	
The in-memory representation of the <objectives> child of the <stage> element	994
TimeDomainStages	
The in-memory representation of the <stages> element	996
TimeDomainStageVar	
The in-memory representation of the <i>element</i>	997
TimeDomainStageVariables	
The in-memory representation of the <variables> child of the <stage> element	998
TimeMeasurement	
The TimeMeasurement Class	1000
TimeSpan	
TimeSpan class	1002
TimingInformation	
The TimingInformation Class	1005
Variable	
The in-memory representation of the variable element	1008
VariableOption	
VariableOption class	1010
Variables	
The in-memory representation of the variables element	1013
VariableSolution	
The VariableSolution Class	1015

VariableValues	
The VariableValues Class	1018
VariableValuesString	
The VariableValuesString Class	1021
VarReferenceMatrixElements	
Data structure to represent variable reference elements in a MatrixType object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a variable	1023
VarReferenceMatrixValues	
Abstract class to help represent the elements in a MatrixType object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc	1027
VarValue	
VarValue Class	1030
VarValueString	
VarValueString Class	1032
WSUtil	
Used by OSSolverAgent client for help in invoking a remote solver	1034
YYLTYPE	1037
YYSTYPE	1038

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/ted/COIN/trunk/OS/src/config_default.h	1039
/home/ted/COIN/trunk/OS/src/config_os_default.h	1040
/home/ted/COIN/trunk/OS/src/OSConfig.h	1077
/home/ted/COIN/trunk/OS/src/OSAgent/OShL.h	1042
/home/ted/COIN/trunk/OS/src/OSAgent/OSSolverAgent.h	1043
/home/ted/COIN/trunk/OS/src/OSAgent/OSWSUtil.h	1044
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLine.h	1046
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLineReader.h	1047
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSDefaultSolver.h	1048
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTree.h	1048
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h	1050

/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSgLWriter.h	1052
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSiLReader.h	1055
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSiLWriter.h	1056
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h This file defines the OSInstance class along with its supporting classes	1057
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h	1060
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h This file defines the OSnLNode class along with its derived classes	1063
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLReader.h	1066
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLWriter.h	1068
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOOption.h	1069
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h	1072
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLReader.h	1075
/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLWriter.h	1076
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSgams2osil.hpp	1078
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSmps2OS.h	1078
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSmps2osil.h	1079
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSnl2OS.h	1080
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosrl2AMPL.h	1082
/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosrl2gams.hpp	1083
/home/ted/COIN/trunk/OS/src/OSParsers/OSgLParseData.h	1083
/home/ted/COIN/trunk/OS/src/OSParsers/OSiLParseData.h	1085
/home/ted/COIN/trunk/OS/src/OSParsers/OSnLParseData.h	1086
/home/ted/COIN/trunk/OS/src/OSParsers/OSoLParseData.h	1088
/home/ted/COIN/trunk/OS/src/OSParsers/OSOptionsStruc.h	1089
/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosil.tab.hpp	1090
/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosol.tab.hpp	1158
/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosrl.tab.hpp	1233
/home/ted/COIN/trunk/OS/src/OSParsers/OSrLParseData.h	1305
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSBonminSolver.h	1308
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCoinSolver.h	1309

/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCouenneSolver.h	1310
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCsdpSolver.h	1311
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSIpoptSolver.h	1313
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSKnitroSolver.h	1314
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSLindoSolver.h	1315
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSMatlabSolver.h	1316
/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSRunSolver.h	1316
/home/ted/COIN/trunk/OS/src/OSUtils/OSBase64.h	1319
/home/ted/COIN/trunk/OS/src/OSUtils/OSDataStructures.h	1320
/home/ted/COIN/trunk/OS/src/OSUtils/OSdtoa.h	1321
/home/ted/COIN/trunk/OS/src/OSUtils/OSErrorClass.h	1322
/home/ted/COIN/trunk/OS/src/OSUtils/OSFileUtil.h	1322
/home/ted/COIN/trunk/OS/src/OSUtils/OSMathUtil.h	1323
/home/ted/COIN/trunk/OS/src/OSUtils/OSOutput.h	1326
/home/ted/COIN/trunk/OS/src/OSUtils/OSParameters.h	1328
/home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.hpp	1349
/home/ted/COIN/trunk/OS/src/OSUtils/OSSmartPtr.hpp	1350
/home/ted/COIN/trunk/OS/src/OSUtils/OSStringUtil.h	1353

5 Namespace Documentation

5.1 Coin Namespace Reference

5.2 Couenne Namespace Reference

6 Class Documentation

6.1 Base64 Class Reference

use this class to read and write data in base64.

```
#include <OSBase64.h>
```

Public Member Functions

- [Base64](#) ()

- [Base64](#) class constructor.
- [~Base64](#) ()
- [Base64](#) class destructor.

Static Public Member Functions

- static std::string [encodeb64](#) (char *bytes, int size)
encode the data in base 64
- static std::string [decodeb64](#) (char *b64bytes)
decode the data in base 64

6.1.1 Detailed Description

use this class to read and write data in base64.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

it is possible to save space by eliminating the need for all the <el> tabs by writing a long string of numbers in b64 format

Definition at line 33 of file OSBase64.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Base64::Base64 ()

[Base64](#) class constructor.

6.1.2.2 Base64::~~Base64 ()

[Base64](#) class destructor.

6.1.3 Member Function Documentation

6.1.3.1 static std::string Base64::encodeb64 (char * bytes, int size) [static]

encode the data in base 64

Parameters

<i>bytes</i>	is the input to be encoded.
<i>size</i>	is the size of the pointer in bytes

Returns

a string in base 64 format.

6.1.3.2 `static std::string Base64::decodeb64 (char * b64bytes) [static]`

decode the data in base 64

Parameters

<i>b64bytes</i>	is the input to be decoded
-----------------	----------------------------

Returns

a string that is decoded.

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

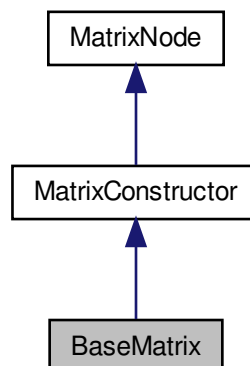
- [/home/ted/COIN/trunk/OS/src/OSUtils/OSBase64.h](#)

6.2 BaseMatrix Class Reference

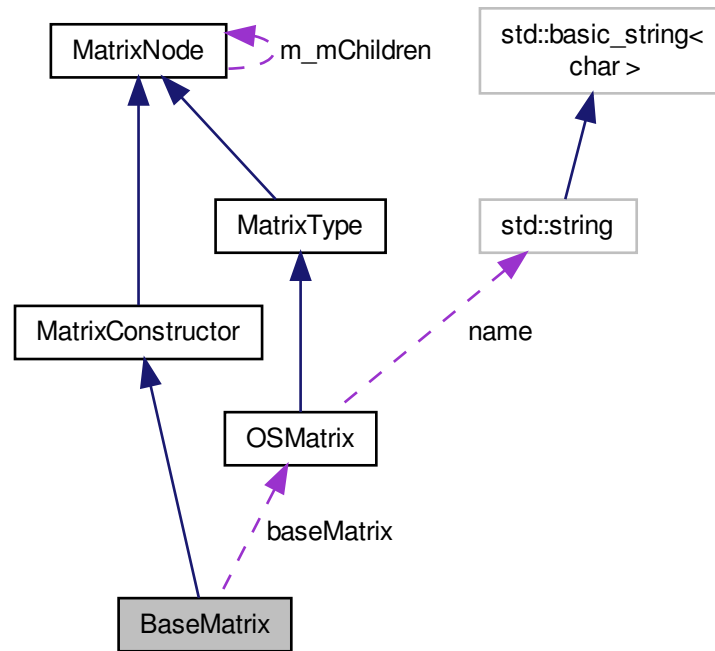
a data structure to represent a point of departure for constructing a matrix by modifying parts of a previously defined matrix

```
#include <OSMatrix.h>
```

Inheritance diagram for BaseMatrix:



Collaboration diagram for BaseMatrix:



Public Member Functions

- [BaseMatrix](#) ()
Standard constructor and destructor methods.
- [~BaseMatrix](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [std::string](#) [getNodeName](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual [std::string](#) [getMatrixNodeInXML](#) ()
- virtual [bool](#) [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [BaseMatrix](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- [bool](#) [isEqual](#) ([BaseMatrix](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [baseMatrixIdx](#)

- the index of the base matrix*
- [OSMatrix](#) * [baseMatrix](#)
 - a pointer to the base matrix*
- int [targetMatrixFirstRow](#)
 - to pinpoint the position of the upper left corner of the base matrix within the target matrix*
- int [targetMatrixFirstCol](#)
- int [baseMatrixStartRow](#)
 - to select the position of the upper left corner of the portion of the base matrix that is to be selected*
- int [baseMatrixStartCol](#)
- int [baseMatrixEndRow](#)
 - to select the position of the lower right corner of the portion of the base matrix that is to be selected*
- int [baseMatrixEndCol](#)
- bool [baseTranspose](#)
 - to allow the base matrix to be transposed before it is attached to the target matrix*
- double [scalarMultiplier](#)
 - to allow the base matrix to be scaled before it is attached to the target matrix*

6.2.1 Detailed Description

a data structure to represent a point of departure for constructing a matrix by modifying parts of a previously defined matrix

Definition at line 1546 of file OSMatrix.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BaseMatrix::BaseMatrix ()

Standard constructor and destructor methods.

6.2.2.2 BaseMatrix::~~BaseMatrix ()

6.2.3 Member Function Documentation

6.2.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE BaseMatrix::getNodeTypes () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.2.3.2 virtual std::string BaseMatrix::getNodeName () [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

6.2.3.3 `virtual ENUM_MATRIX_TYPE BaseMatrix::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.2.3.4 `virtual std::string BaseMatrix::getMatrixNodeInXML () [virtual]`

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.2.3.5 `virtual bool BaseMatrix::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]`

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Implements [MatrixNode](#).

6.2.3.6 `BaseMatrix * BaseMatrix::cloneMatrixNode () [virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [MatrixNode](#) of the proper type.

Implements [MatrixNode](#).

6.2.3.7 `bool BaseMatrix::isEqual (BaseMatrix * that)`

A function to check for the equality of two objects.

6.2.4 Member Data Documentation

6.2.4.1 `int BaseMatrix::baseMatrixIdx`

the index of the base matrix

Definition at line 1552 of file OSMatrix.h.

6.2.4.2 OSMatrix* BaseMatrix::baseMatrix

a pointer to the base matrix

Definition at line 1557 of file OSMatrix.h.

6.2.4.3 int BaseMatrix::targetMatrixFirstRow

to pinpoint the position of the upper left corner of the base matrix within the target matrix

Definition at line 1562 of file OSMatrix.h.

6.2.4.4 int BaseMatrix::targetMatrixFirstCol

Definition at line 1563 of file OSMatrix.h.

6.2.4.5 int BaseMatrix::baseMatrixStartRow

to select the position of the upper left corner of the portion of the base matrix that is to be selected

Definition at line 1569 of file OSMatrix.h.

6.2.4.6 int BaseMatrix::baseMatrixStartCol

Definition at line 1570 of file OSMatrix.h.

6.2.4.7 int BaseMatrix::baseMatrixEndRow

to select the position of the lower right corner of the portion of the base matrix that is to be selected

Definition at line 1576 of file OSMatrix.h.

6.2.4.8 int BaseMatrix::baseMatrixEndCol

Definition at line 1577 of file OSMatrix.h.

6.2.4.9 bool BaseMatrix::baseTranspose

to allow the base matrix to be transposed before it is attached to the target matrix

Definition at line 1582 of file OSMatrix.h.

6.2.4.10 double BaseMatrix::scalarMultiplier

to allow the base matrix to be scaled before it is attached to the target matrix

Definition at line 1587 of file OSMatrix.h.

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

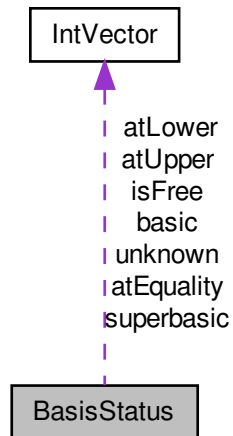
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.3 BasisStatus Class Reference

a data structure to represent an LP basis on both input and output

```
#include <OSGeneral.h>
```

Collaboration diagram for BasisStatus:



Public Member Functions

- `BasisStatus ()`
- `~BasisStatus ()`
- `bool isEqual (BasisStatus *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant, int iMin, int iMax)`
A function to make a random instance of this class.
- `bool deepCopyFrom (BasisStatus *that)`
A function to make a deep copy of an instance of this class.
- `bool setIntVector (int status, int *i, int ni)`
Set the indices for a particular status.
- `bool addIdx (int status, int idx)`
Add one index to a particular status.
- `int getNumberOfEI (int status)`
Get the number of indices for a particular status.
- `int getEI (int status, int j)`
Get one entry in the array of indices for a particular status.
- `bool getIntVector (int status, int *i)`
Get the entire array of indices for a particular status.
- `int getBasisDense (int *resultArray, int dim, bool flipIdx)`
Get the entire array of basis status in dense form.

Public Attributes

- [IntVector](#) * [basic](#)
- [IntVector](#) * [atLower](#)
- [IntVector](#) * [atUpper](#)
- [IntVector](#) * [atEquality](#)
- [IntVector](#) * [isFree](#)
- [IntVector](#) * [superbasic](#)
- [IntVector](#) * [unknown](#)

6.3.1 Detailed Description

a data structure to represent an LP basis on both input and output

Definition at line 645 of file OSGeneral.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BasisStatus::BasisStatus ()

6.3.2.2 BasisStatus::~~BasisStatus ()

6.3.3 Member Function Documentation

6.3.3.1 bool BasisStatus::isEqual (**BasisStatus** * *that*)

A function to check for the equality of two objects.

6.3.3.2 bool BasisStatus::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that an entry in this basis can take
<i>iMax</i> ,:	greatest index value (inclusive) that an entry in this basis can take

6.3.3.3 bool BasisStatus::deepCopyFrom (**BasisStatus** * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.3.3.4 bool BasisStatus::setIntVector (int *status*, int * *i*, int *ni*)

Set the indices for a particular status.

Parameters

<i>status</i>	is a string representing the allowed statuses (as defined in enumeration ENUM_BASIS_STATUS - see below)
<i>i</i>	contains the array of indices
<i>ni</i>	contains the number of elements in <i>i</i>

6.3.3.5 bool BasisStatus::addIdx (int *status*, int *idx*)

Add one index to a particular status.

Parameters

<i>status</i>	is a string representing the allowed statuses (as defined in enumeration ENUM_BASIS_STATUS - see below)
<i>idx</i>	contains the value of the index

6.3.3.6 int BasisStatus::getNumberOfEI (int *status*)

Get the number of indices for a particular status.

Parameters

<i>status</i>	is a string representing the allowed statuses (at present "basic", "atLower", "atUpper", "isFree", "superbasic", "unknown")
---------------	---

Returns

the number of indices or -1 if the object does not exist

6.3.3.7 int BasisStatus::getEI (int *status*, int *j*)

Get one entry in the array of indices for a particular status.

Parameters

<i>status</i>	is an integer representing the allowed statuses (as governed by enumeration ENUM_BASIS_STATUS — see below)
<i>j</i>	is the (zero-based) position of the entry within the array

Returns

the value

6.3.3.8 bool BasisStatus::getIntVector (int *status*, int * *i*)

Get the entire array of indices for a particular status.

Parameters

<i>status</i>	is a string representing the allowed statuses (as governed by enumeration ENUM_BASIS_STAT-US — see below)
<i>i</i>	is the location where the user wants to store the array

Returns

whether the operation was successful

Note

it is the user's responsibility to reserve sufficient memory to hold the vector being returned.

6.3.3.9 int BasisStatus::getBasisDense (int * *resultArray*, int *dim*, bool *flipIdx*)

Get the entire array of basis status in dense form.

Parameters

<i>resultArray</i>	is the location where the user wants to store the array
<i>dim</i>	is the size of the resultArray
<i>flipIdx</i>	indicates whether the index values need to be flipped (used for representations of objective rows)

Returns

status of the operation: < 0: error condition = 0: no new data found (i.e., basis information is empty)

0: number of elements found

Note

it is the user's responsibility to reserve sufficient memory to hold the vector being returned.

6.3.4 Member Data Documentation**6.3.4.1 IntVector* BasisStatus::basic**

Definition at line 648 of file OSGeneral.h.

6.3.4.2 IntVector* BasisStatus::atLower

Definition at line 649 of file OSGeneral.h.

6.3.4.3 IntVector* BasisStatus::atUpper

Definition at line 650 of file OSGeneral.h.

6.3.4.4 IntVector* BasisStatus::atEquality

Definition at line 651 of file OSGeneral.h.

6.3.4.5 IntVector* BasisStatus::isFree

Definition at line 652 of file OSGeneral.h.

6.3.4.6 IntVector* BasisStatus::superbasic

Definition at line 653 of file OSGeneral.h.

6.3.4.7 IntVector* BasisStatus::unknown

Definition at line 654 of file OSGeneral.h.

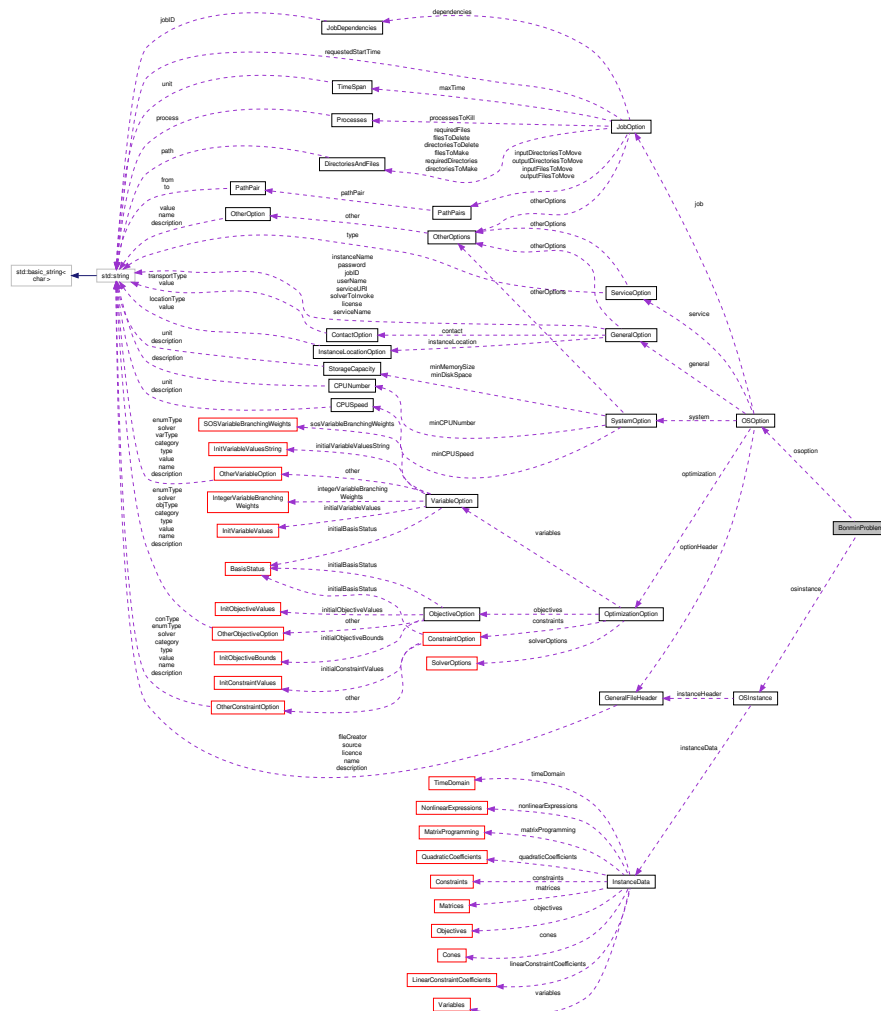
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSGeneral.h](#)

6.4 BonminProblem Class Reference

```
#include <OSBonminSolver.h>
```

Collaboration diagram for BonminProblem:



Public Member Functions

- [BonminProblem](#) ([OSInstance](#) *osinstance_, [OSOption](#) *osoption_)
the *BonminProblem* class constructor
- virtual [~BonminProblem](#) ()
the *BonminProblem* class destructor
- virtual bool [get_scaling_parameters](#) (lpopt::Number &obj_scaling, bool &use_x_scaling, lpopt::Index n, lpopt::Number *x_scaling, bool &use_g_scaling, lpopt::Index m, lpopt::Number *g_scaling)
- virtual const [SosInfo](#) * [sosConstraints](#) () const
- virtual const [BranchingInfo](#) * [branchingInfo](#) () const
- void [printSolutionAtEndOfAlgorithm](#) ()

Overloaded functions specific to a TMINLP.

now for some pure Bonmin methods

- virtual bool `get_variables_types` (lpopt::Index n, VariableType *var_types)

Pass the type of the variables (INTEGER, BINARY, CONTINUOUS) to the optimizer.

- virtual bool [get_variables_linearity](#) (Ipopt::Index n, Ipopt::TNLP::LinearityType *var_types)

Pass info about linear and nonlinear variables.

- virtual bool [get_constraints_linearity](#) (Ipopt::Index m, Ipopt::TNLP::LinearityType *const_types)

Pass the type of the constraints (LINEAR, NON_LINEAR) to the optimizer.

Overloaded functions defining a TNLP.

This group of function implement the various elements needed to define and solve a TNLP.

They are the same as those in a standard Ipopt NLP problem

- virtual bool [get_nlp_info](#) (Ipopt::Index &n, Ipopt::Index &m, Ipopt::Index &nnz_jac_g, Ipopt::Index &nnz_h_lag, Ipopt::TNLP::IndexStyleEnum &index_style)

Method to pass the main dimensions of the problem to Ipopt.

- virtual bool [get_bounds_info](#) (Ipopt::Index n, Ipopt::Number *x_l, Ipopt::Number *x_u, Ipopt::Index m, Ipopt::Number *g_l, Ipopt::Number *g_u)

Bonmin specific methods for defining the nlp problem.

- virtual bool [get_starting_point](#) (Ipopt::Index n, bool init_x, Ipopt::Number *x, bool init_z, Ipopt::Number *z_L, Ipopt::Number *z_U, Ipopt::Index m, bool init_lambda, Ipopt::Number *lambda)

Method to return the starting point for the algorithm.

- virtual bool [eval_f](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number &obj_value)

Method to return the objective value.

- virtual bool [eval_grad_f](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number *grad_f)

Method to return the gradient of the objective.

- virtual bool [eval_g](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Index m, Ipopt::Number *g)

Method to return the constraint residuals.

- virtual bool [eval_jac_g](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Index m, Ipopt::Index nele_jac, Ipopt::Index *iRow, Ipopt::Index *jCol, Ipopt::Number *values)

Method to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)

- virtual bool [eval_h](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number obj_factor, Ipopt::Index m, const Ipopt::Number *lambda, bool new_lambda, Ipopt::Index nele_hess, Ipopt::Index *iRow, Ipopt::Index *jCol, Ipopt::Number *values)

Method to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)

Solution Methods

- virtual void [finalize_solution](#) (Bonmin::TMINLP::SolverReturn status_, Ipopt::Index n, const Ipopt::Number *x, Ipopt::Number obj_value)

Method called by Ipopt at the end of optimization.

Public Attributes

- [OSInstance](#) * [osinstance](#)
- [OSOption](#) * [osoption](#)
- Bonmin::TMINLP::SolverReturn [status](#)

6.4.1 Detailed Description

Definition at line 53 of file OSBonminSolver.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 BonminProblem::BonminProblem (OSInstance * *osinstance_*, OSOption * *osoption_*)

the BonminProblemclass constructor

6.4.2.2 virtual BonminProblem::~~BonminProblem () [virtual]

the [BonminProblem](#) class destructor

6.4.3 Member Function Documentation

6.4.3.1 virtual bool BonminProblem::get_variables_types (Ipopt::Index *n*, VariableType * *var_types*) [virtual]

Pass the type of the variables (INTEGER, BINARY, CONTINUOUS) to the optimizer.

Parameters

<i>n</i>	size of <i>var_types</i> (has to be equal to the number of variables in the problem)
<i>var_types</i>	types of the variables (has to be filled by function).

6.4.3.2 virtual bool BonminProblem::get_variables_linearity (Ipopt::Index *n*, Ipopt::TNLP::LinearityType * *var_types*) [virtual]

Pass info about linear and nonlinear variables.

6.4.3.3 virtual bool BonminProblem::get_constraints_linearity (Ipopt::Index *m*, Ipopt::TNLP::LinearityType * *const_types*) [virtual]

Pass the type of the constraints (LINEAR, NON_LINEAR) to the optimizer.

Parameters

<i>m</i>	size of <i>const_types</i> (has to be equal to the number of constraints in the problem)
<i>const_types</i>	types of the constraints (has to be filled by function).

6.4.3.4 virtual bool BonminProblem::get_nlp_info (Ipopt::Index & *n*, Ipopt::Index & *m*, Ipopt::Index & *nnz_jac_g*, Ipopt::Index & *nnz_h_lag*, Ipopt::TNLP::IndexStyleEnum & *index_style*) [virtual]

Method to pass the main dimensions of the problem to Ipopt.

Parameters

<i>n</i>	number of variables in problem.
<i>m</i>	number of constraints.
<i>nnz_jac_g</i>	number of non zeroes in Jacobian of constraints system.
<i>nnz_h_lag</i>	number of non zeroes in Hessian of the Lagrangean.
<i>index_style</i>	indicate whether arrays are numbered from 0 (C-style) or from 1 (Fortran).

Returns

true in case of success.

6.4.3.5 `virtual bool BonminProblem::get_bounds_info (lpopt::Index n, lpopt::Number * x_l, lpopt::Number * x_u, lpopt::Index m, lpopt::Number * g_l, lpopt::Number * g_u)` [virtual]

Bonmin specific methods for defining the nlp problem.

Method to return the bounds for my problem

6.4.3.6 `virtual bool BonminProblem::get_starting_point (lpopt::Index n, bool init_x, lpopt::Number * x, bool init_z, lpopt::Number * z_L, lpopt::Number * z_U, lpopt::Index m, bool init_lambda, lpopt::Number * lambda)` [virtual]

Method to return the starting point for the algorithm.

6.4.3.7 `virtual bool BonminProblem::eval_f (lpopt::Index n, const lpopt::Number * x, bool new_x, lpopt::Number & obj_value)` [virtual]

Method to return the objective value.

6.4.3.8 `virtual bool BonminProblem::eval_grad_f (lpopt::Index n, const lpopt::Number * x, bool new_x, lpopt::Number * grad_f)` [virtual]

Method to return the gradient of the objective.

6.4.3.9 `virtual bool BonminProblem::eval_g (lpopt::Index n, const lpopt::Number * x, bool new_x, lpopt::Index m, lpopt::Number * g)` [virtual]

Method to return the constraint residuals.

6.4.3.10 `virtual bool BonminProblem::eval_jac_g (lpopt::Index n, const lpopt::Number * x, bool new_x, lpopt::Index m, lpopt::Index nele_jac, lpopt::Index * iRow, lpopt::Index * jCol, lpopt::Number * values)` [virtual]

Method to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)

6.4.3.11 `virtual bool BonminProblem::eval_h (lpopt::Index n, const lpopt::Number * x, bool new_x, lpopt::Number obj_factor, lpopt::Index m, const lpopt::Number * lambda, bool new_lambda, lpopt::Index nele_hess, lpopt::Index * iRow, lpopt::Index * jCol, lpopt::Number * values)` [virtual]

Method to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)

6.4.3.12 `virtual bool BonminProblem::get_scaling_parameters (lpopt::Number & obj_scaling, bool & use_x_scaling, lpopt::Index n, lpopt::Number * x_scaling, bool & use_g_scaling, lpopt::Index m, lpopt::Number * g_scaling)` [virtual]

6.4.3.13 `virtual void BonminProblem::finalize_solution (Bonmin::TMINLP::SolverReturn status, lpopt::Index n, const lpopt::Number * x, lpopt::Number obj_value)` [virtual]

Method called by lpopt at the end of optimization.

6.4.3.14 `virtual const SosInfo* BonminProblem::sosConstraints () const` [inline],[virtual]

Definition at line 166 of file OSBonminSolver.h.

6.4.3.15 `virtual const BranchingInfo* BonminProblem::branchingInfo () const` `[inline],[virtual]`

Definition at line 170 of file OSBonminSolver.h.

6.4.3.16 `void BonminProblem::printSolutionAtEndOfAlgorithm ()` `[inline]`

Definition at line 175 of file OSBonminSolver.h.

6.4.4 Member Data Documentation

6.4.4.1 `OSInstance* BonminProblem::osinstance`

Definition at line 66 of file OSBonminSolver.h.

6.4.4.2 `OSOption* BonminProblem::osoption`

Definition at line 68 of file OSBonminSolver.h.

6.4.4.3 `Bonmin::TMINLP::SolverReturn BonminProblem::status`

Definition at line 70 of file OSBonminSolver.h.

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

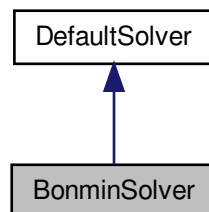
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSBonminSolver.h](#)

6.5 BonminSolver Class Reference

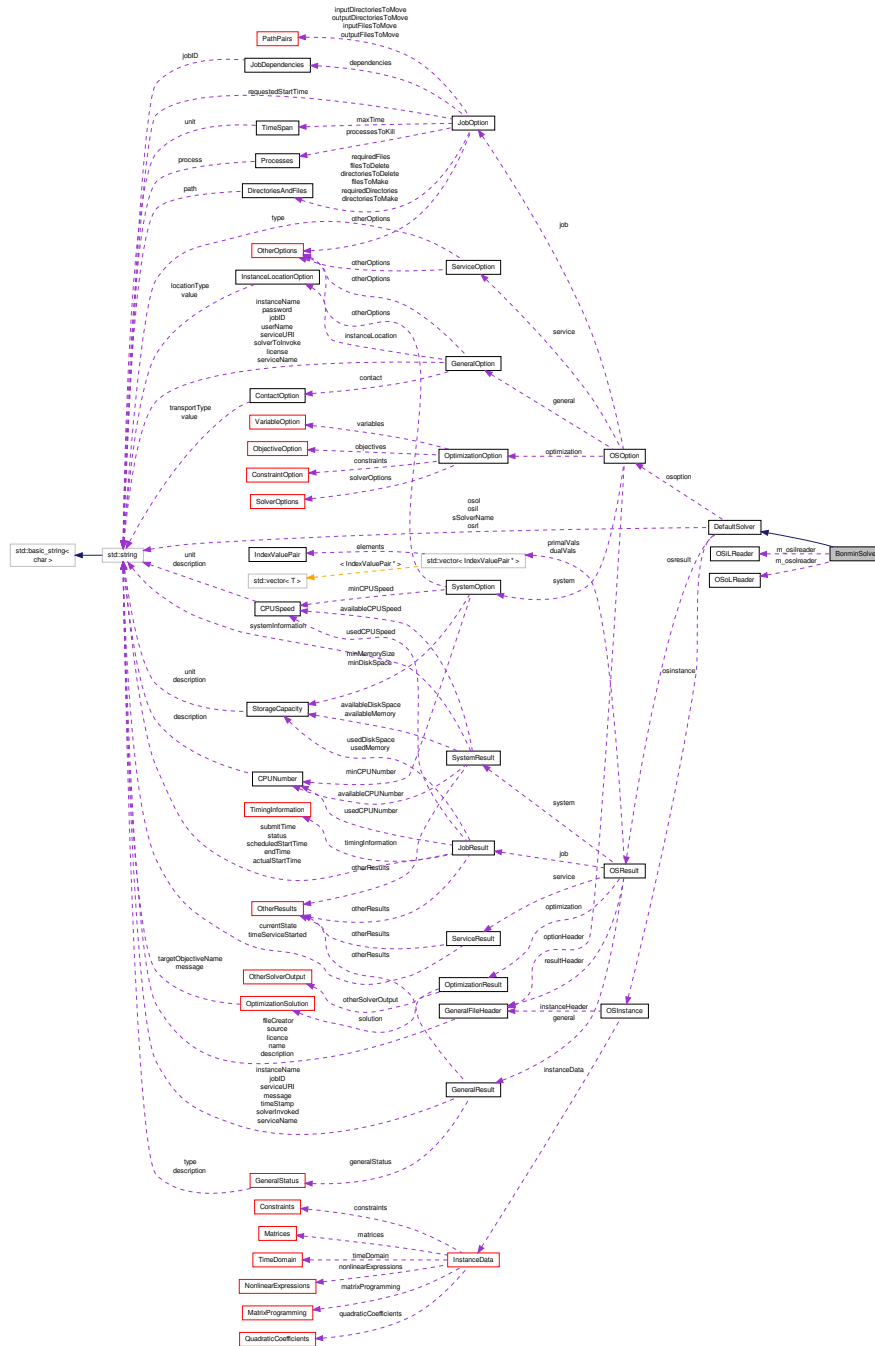
The [BonminSolver](#) class solves problems using Ipopt.

```
#include <OSBonminSolver.h>
```

Inheritance diagram for BonminSolver:



Collaboration diagram for BonminSolver:



Public Member Functions

- [BonminSolver](#) ()
the *BonminSolver* class constructor
- [~BonminSolver](#) ()

the [IpoptSolver](#) class destructor

- virtual void [solve](#) () throw (ErrorClass)
solve results in an instance being read into the Bonmin data structures and optimized
- virtual void [buildSolverInstance](#) () throw (ErrorClass)
buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API
- virtual void [setSolverOptions](#) () throw (ErrorClass)
The implementation of the virtual functions.
- void [dataEchoCheck](#) ()
use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file
- void [writeResult](#) ()
use this to write the solution information to an [OSResult](#) object

Public Attributes

- [Ipopt::SmartPtr< BonminProblem > tminlp](#)
- [Bonmin::Bab](#) [bb](#)
- [Bonmin::TMINLP::SolverReturn](#) [status](#)
- [OSiLReader](#) * [m_osilreader](#)
m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed
- [OSoLReader](#) * [m_osolreader](#)
m_osolreader is an [OSoLReader](#) object used to create an osoption from an osol string if needed

6.5.1 Detailed Description

The [BonminSolver](#) class solves problems using Ipopt.

Author

Jun Ma, Horand Gassmann, Kipp Martin

Version

1.0, 07/05/2008

Since

OS 1.0

Remarks

this class takes an OSiL instance and optimizes it using the COIN-OR Ipopt solver

Definition at line 225 of file OSBonminSolver.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [BonminSolver::BonminSolver](#) ()

the [BonminSolver](#) class constructor

6.5.2.2 BonminSolver::~~BonminSolver ()

the [IpoptSolver](#) class destructor

6.5.3 Member Function Documentation

6.5.3.1 virtual void BonminSolver::solve () throw (ErrorClass) [virtual]

solve results in an instance being read into the Bonmin data structures and optimized

Implements [DefaultSolver](#).

6.5.3.2 virtual void BonminSolver::buildSolverInstance () throw (ErrorClass) [virtual]

buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API

Implements [DefaultSolver](#).

6.5.3.3 void BonminSolver::setSolverOptions () throw (ErrorClass) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.5.3.4 void BonminSolver::dataEchoCheck ()

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

6.5.3.5 void BonminSolver::writeResult ()

use this to write the solution information to an [OSResult](#) object

6.5.4 Member Data Documentation

6.5.4.1 Ipopt::SmartPtr<BonminProblem> BonminSolver::tminlp

Definition at line 239 of file OSBonminSolver.h.

6.5.4.2 Bonmin::Bab BonminSolver::bb

Definition at line 242 of file OSBonminSolver.h.

6.5.4.3 Bonmin::TMINLP::SolverReturn BonminSolver::status

Definition at line 244 of file OSBonminSolver.h.

6.5.4.4 OSiLReader* BonminSolver::m_osilreader

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 276 of file OSBonminSolver.h.

6.5.4.5 OSoLReader* BonminSolver::m_osolreader

m_osolreader is an [OSoLReader](#) object used to create an ooption from an osol string if needed

Definition at line 282 of file OSBonminSolver.h.

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

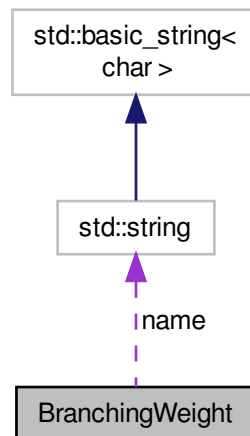
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/[OSBonminSolver.h](#)

6.6 BranchingWeight Class Reference

the [BranchingWeight](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for BranchingWeight:



Public Member Functions

- [BranchingWeight](#) ()
Default constructor.
- [~BranchingWeight](#) ()
Class destructor.
- bool [IsEqual](#) ([BranchingWeight](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([BranchingWeight](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int `idx`
index of the variable
- std::string `name`
optional variable name
- double `value`
branching weight

6.6.1 Detailed Description

the [BranchingWeight](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/11/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1611 of file OSOption.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 BranchingWeight::BranchingWeight ()

Default constructor.

6.6.2.2 BranchingWeight::~~BranchingWeight ()

Class destructor.

6.6.3 Member Function Documentation

6.6.3.1 bool BranchingWeight::IsEqual (BranchingWeight * *that*)

A function to check for the equality of two objects.

6.6.3.2 bool BranchingWeight::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.6.3.3 bool BranchingWeight::deepCopyFrom (BranchingWeight * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.6.4 Member Data Documentation

6.6.4.1 int BranchingWeight::idx

index of the variable

Definition at line 1616 of file OSOption.h.

6.6.4.2 std::string BranchingWeight::name

optional variable name

Definition at line 1619 of file OSOption.h.

6.6.4.3 double BranchingWeight::value

branching weight

Definition at line 1622 of file OSOption.h.

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

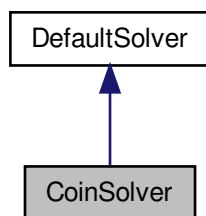
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.7 CoinSolver Class Reference

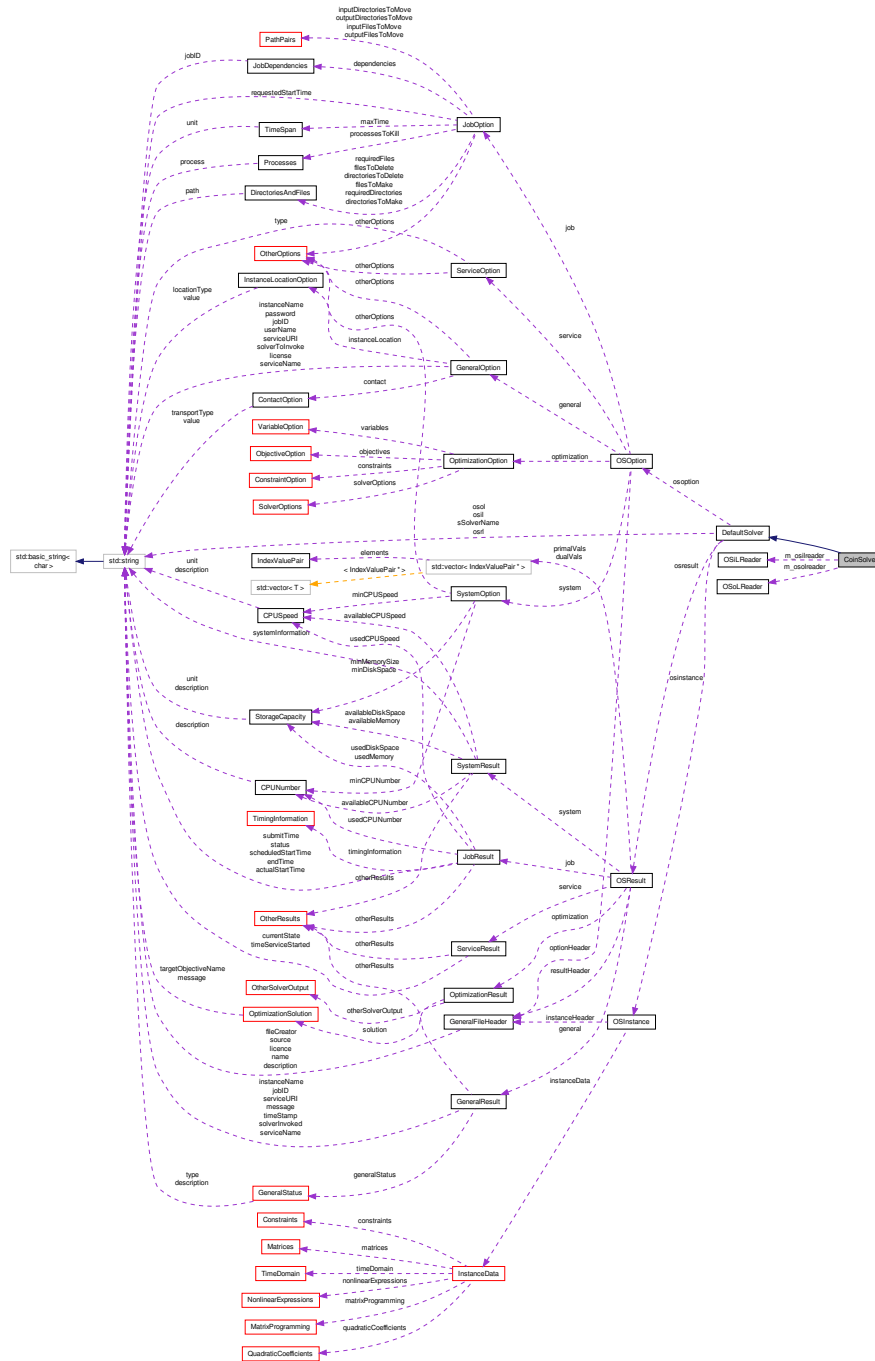
Implements a solve method for the [Coin](#) solvers.

```
#include <OSCoinSolver.h>
```

Inheritance diagram for CoinSolver:



Collaboration diagram for CoinSolver:



Public Member Functions

- [CoinSolver](#) ()
The class constructor.
- [~CoinSolver](#) ()

The class destructor.

- virtual void `solve` () throw (ErrorClass)

The implementation of the corresponding virtual function.

- virtual void `buildSolverInstance` () throw (ErrorClass)

The implementation of the corresponding virtual function.

- virtual void `setSolverOptions` () throw (ErrorClass)

The implementation of the corresponding virtual function.

- bool `setCoinPackedMatrix` ()

*Create a **CoinPackedMatrix**.*

- std::string `getCoinSolverType` (std::string osol_)

Get the solver type, e.g.

- void `dataEchoCheck` ()

Print out problem parameters.

- void `writeResult` (OsiSolverInterface *solver)
- void `writeResult` (CbcModel *model)

Public Attributes

- OsiSolverInterface * `osiSolver`

osiSolver is the osi solver object – in this case clp, glpk, cbc, cplex, symphony or dylp

- OSiLReader * `m_osilreader`

*m_osilreader is an **OSiLReader** object used to create an osinstance from an osil string if needed*

- OSoLReader * `m_osolreader`

*m_osolreader is an **OSoLReader** object used to create an ooption from an osol string if needed*

6.7.1 Detailed Description

Implements a solve method for the **Coin** solvers.

This class implements a solve method for the **Coin** solvers It reads an **OSInstance** object and puts into the **Coin** OSI format

Definition at line 37 of file OSCoinSolver.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 CoinSolver::CoinSolver ()

The class constructor.

6.7.2.2 CoinSolver::~~CoinSolver ()

The class destructor.

6.7.3 Member Function Documentation

6.7.3.1 void CoinSolver::solve () throw (ErrorClass) [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

6.7.3.2 void CoinSolver::buildSolverInstance () throw (ErrorClass) [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

6.7.3.3 void CoinSolver::setSolverOptions () throw (ErrorClass) [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

6.7.3.4 bool CoinSolver::setCoinPackedMatrix ()

Create a **CoinPackedMatrix**.

Returns

true if a **CoinPackedMatrix** successfully created.

6.7.3.5 string CoinSolver::getCoinSolverType (std::string osol_)

Get the solver type, e.g.

clp or glpk

Parameters

<i>a</i>	string that is an instance of OSoL
----------	------------------------------------

Returns

a string which contains the value of clp or glpk.

6.7.3.6 string CoinSolver::dataEchoCheck ()

Print out problem parameters.

Returns

void

6.7.3.7 void CoinSolver::writeResult (OsiSolverInterface * *solver*)

6.7.3.8 void CoinSolver::writeResult (CbcModel * *model*)

6.7.4 Member Data Documentation

6.7.4.1 OsiSolverInterface* CoinSolver::osiSolver

osiSolver is the osi solver object – in this case clp, glpk, cbc, cplex, symphony or dylp

Definition at line 93 of file OSCoinSolver.h.

6.7.4.2 OSiLReader* CoinSolver::m_osilreader

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 101 of file OSCoinSolver.h.

6.7.4.3 OSoLReader* CoinSolver::m_osolreader

m_osolreader is an [OSoLReader](#) object used to create an osoption from an osol string if needed

Definition at line 107 of file OSCoinSolver.h.

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

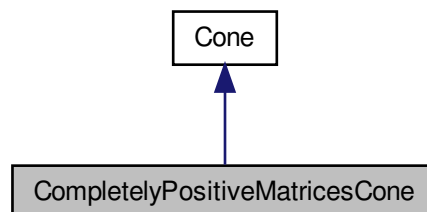
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCoinSolver.h](#)

6.8 CompletelyPositiveMatricesCone Class Reference

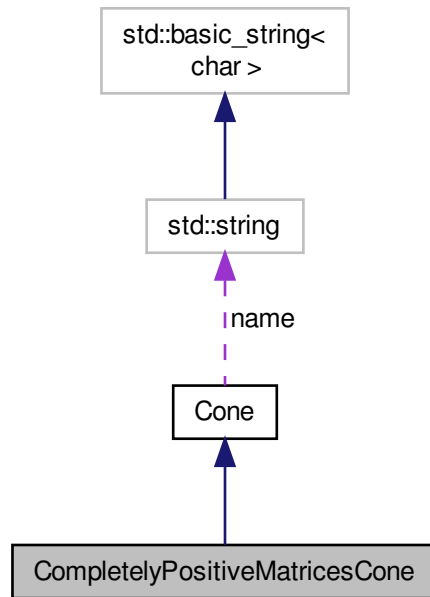
The [CompletelyPositiveMatricesCone](#) Class.

```
#include <OSInstance.h>
```

Inheritance diagram for CompletelyPositiveMatricesCone:



Collaboration diagram for CompletelyPositiveMatricesCone:



Public Member Functions

- [CompletelyPositiveMatricesCone](#) ()
default constructor.
- [~CompletelyPositiveMatricesCone](#) ()
default destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [CompletelyPositiveMatricesCone](#) object in XML format.
- bool [IsEqual](#) ([CompletelyPositiveMatricesCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([CompletelyPositiveMatricesCone](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

6.8.1 Detailed Description

The [CompletelyPositiveMatricesCone](#) Class.

Remarks

The in-memory representation of the OSiL element <completelyPositiveMatricesCone>

Definition at line 1187 of file OSInstance.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 CompletelyPositiveMatricesCone::CompletelyPositiveMatricesCone ()

default constructor.

6.8.2.2 CompletelyPositiveMatricesCone::~~CompletelyPositiveMatricesCone ()

default destructor.

6.8.3 Member Function Documentation

6.8.3.1 virtual std::string CompletelyPositiveMatricesCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.8.3.2 virtual std::string CompletelyPositiveMatricesCone::getConeInXML () [virtual]

Write a [CompletelyPositiveMatricesCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.8.3.3 bool CompletelyPositiveMatricesCone::isEqual (CompletelyPositiveMatricesCone * that)

A function to check for the equality of two objects.

6.8.3.4 bool CompletelyPositiveMatricesCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.8.3.5 `bool CompletelyPositiveMatricesCone::deepCopyFrom (CompletelyPositiveMatricesCone * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

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

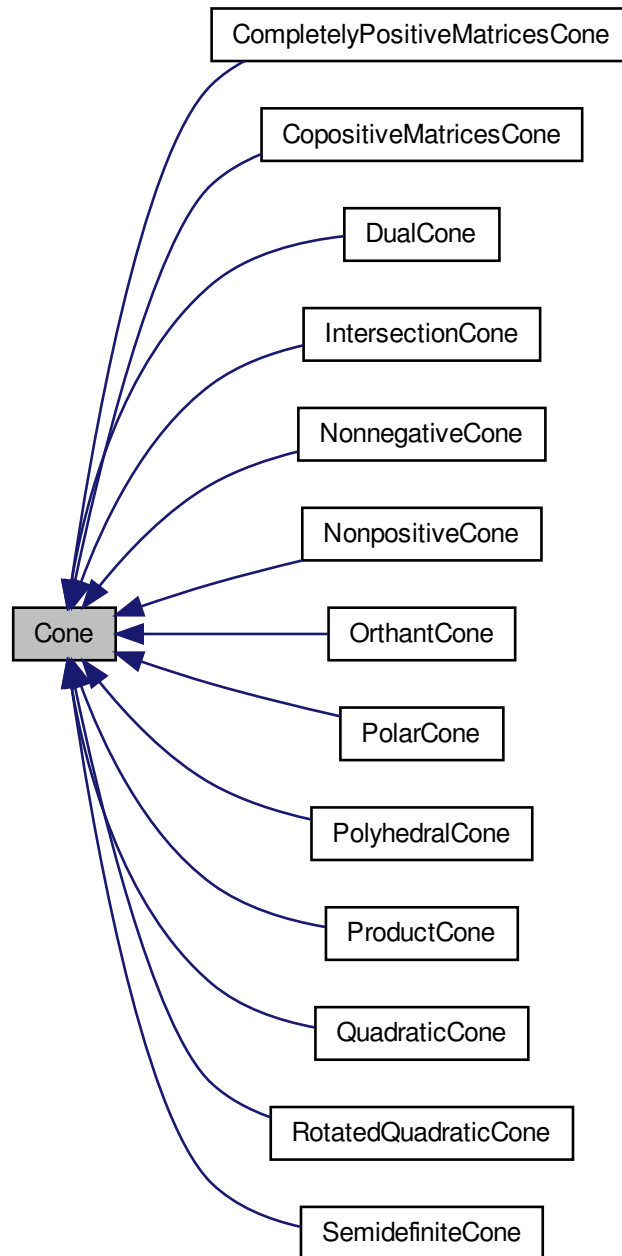
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.9 Cone Class Reference

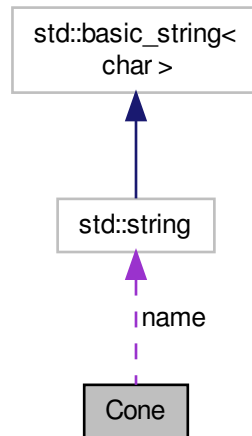
The in-memory representation of a generic cone Specific cone types are derived from this generic class.

```
#include <OSInstance.h>
```

Inheritance diagram for Cone:



Collaboration diagram for Cone:



Public Member Functions

- `Cone ()`
The `Cone` class constructor.
- `~Cone ()`
The `Cone` class destructor.
- virtual `std::string getConeName ()`
- virtual `std::string getConeInXML ()=0`
Write a `Cone` object in XML format.
- `bool IsEqual (Cone *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant, int iMin, int iMax)`
A function to make a random instance of this class.
- `bool deepCopyFrom (Cone *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int numberOfRows`
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- `int numberOfColumns`
- `int numberOfOtherIndexes`
`Cones` can also be formed by Multidimensional tensors.
- `int * otherIndexes`
- `ENUM_CONE_TYPE coneType`
The type of the cone.

- `std::string name`
The cone can have a name for easier identification.
- `int idx`
cones are referenced by an (automatically created) index

6.9.1 Detailed Description

The in-memory representation of a generic cone. Specific cone types are derived from this generic class.
Definition at line 530 of file `OSInstance.h`.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 `Cone::Cone ()`

The `Cone` class constructor.

6.9.2.2 `Cone::~~Cone ()`

The `Cone` class destructor.

6.9.3 Member Function Documentation

6.9.3.1 `virtual std::string Cone::getConeName () [virtual]`

Returns

the type of cone as a string

Reimplemented in `PolarCone`, `DualCone`, `IntersectionCone`, `ProductCone`, `CompletelyPositiveMatricesCone`, `CopositiveMatricesCone`, `SemidefiniteCone`, `RotatedQuadraticCone`, `QuadraticCone`, `PolyhedralCone`, `OrthantCone`, `NonpositiveCone`, and `NonnegativeCone`.

6.9.3.2 `virtual std::string Cone::getConeInXML () [pure virtual]`

Write a `Cone` object in XML format.

This is used by `OSILWriter` to write a `<cone>` element.

Returns

the cone and its children as an XML string.

Implemented in `IntersectionCone`, `ProductCone`, `CompletelyPositiveMatricesCone`, `CopositiveMatricesCone`, `SemidefiniteCone`, `RotatedQuadraticCone`, `QuadraticCone`, `PolyhedralCone`, `OrthantCone`, `NonpositiveCone`, and `NonnegativeCone`.

6.9.3.3 `bool Cone::IsEqual (Cone * that)`

A function to check for the equality of two objects.

6.9.3.4 `bool Cone::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented in [PolarCone](#), [DualCone](#), [IntersectionCone](#), [ProductCone](#), [CompletelyPositiveMatricesCone](#), [CopositiveMatricesCone](#), [SemidefiniteCone](#), [RotatedQuadraticCone](#), [QuadraticCone](#), [PolyhedralCone](#), [OrthantCone](#), [NonpositiveCone](#), and [NonnegativeCone](#).

6.9.3.5 bool Cone::deepCopyFrom (Cone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.9.4 Member Data Documentation

6.9.4.1 int Cone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 543 of file OSInstance.h.

6.9.4.2 int Cone::numberOfColumns

Definition at line 544 of file OSInstance.h.

6.9.4.3 int Cone::numberOfOtherIndexes

[Cones](#) can also be formed by Multidimensional tensors.

(the Kronecker product, for instance, can be thought of as a four-dimensional tensor). We therefore allow additional dimensions, although they have not yet been implemented.

Definition at line 552 of file OSInstance.h.

6.9.4.4 int* Cone::otherIndexes

Definition at line 553 of file OSInstance.h.

6.9.4.5 ENUM_CONE_TYPE Cone::coneType

The type of the cone.

Definition at line 556 of file OSInstance.h.

6.9.4.6 std::string Cone::name

The cone can have a name for easier identification.

Definition at line 559 of file OSInstance.h.

6.9.4.7 int Cone::idx

cones are referenced by an (automatically created) index

Definition at line 562 of file OSInstance.h.

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

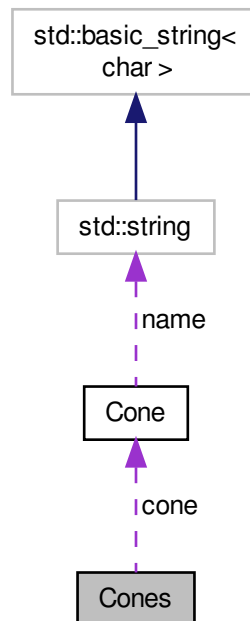
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.10 Cones Class Reference

The in-memory representation of the <**cones**> element.

```
#include <OSInstance.h>
```

Collaboration diagram for Cones:



Public Member Functions

- [Cones](#) ()
The `Cones` class constructor.
- [~Cones](#) ()
The `Cones` class destructor.

- bool `isEqual` (`Cones *that`)
A function to check for the equality of two objects.
- bool `setRandom` (double *density*, bool *conformant*, int *iMin*, int *iMax*)
A function to make a random instance of this class.
- bool `deepCopyFrom` (`Cones *that`)
A function to make a deep copy of an instance of this class.

Public Attributes

- int `numberOfCones`
numberOfCones is the number of <nl> elements in the <cones> element.
- `Cone ** cone`
cone is pointer to an array of `Cone` object pointers

6.10.1 Detailed Description

The in-memory representation of the <**cones**> element.

Definition at line 1533 of file OSInstance.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `Cones::Cones ()`

The `Cones` class constructor.

6.10.2.2 `Cones::~~Cones ()`

The `Cones` class destructor.

6.10.3 Member Function Documentation

6.10.3.1 `bool Cones::isEqual (Cones * that)`

A function to check for the equality of two objects.

6.10.3.2 `bool Cones::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.10.3.3 `bool Cones::deepCopyFrom (Cones * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.10.4 Member Data Documentation

6.10.4.1 int Cones::numberOfCones

numberOfCones is the number of <nl> elements in the <cones> element.

Definition at line 1547 of file OSInstance.h.

6.10.4.2 Cone** Cones::cone

cone is pointer to an array of [Cone](#) object pointers

Definition at line 1550 of file OSInstance.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.11 ConReferenceMatrixElement Class Reference

a data structure to represent an entry in a conReferenceMatrix element, which consists of a constraint reference as well as a value type.

```
#include <OSMatrix.h>
```

Public Member Functions

- [ConReferenceMatrixElement](#) ()
- [~ConReferenceMatrixElement](#) ()
- bool [isEqual](#) ([ConReferenceMatrixElement](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([ConReferenceMatrixElement](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [conReference](#)
contains a reference to a row of the problem (objective if negative, constraint otherwise)
- [ENUM_CONREFERENCE_VALUETYPE](#) valueType
Several different types of values can be derived from a problem constraint.
- double [value](#)
This element contains the value.

6.11.1 Detailed Description

a data structure to represent an entry in a conReferenceMatrix element, which consists of a constraint reference as well as a value type.

Remarks

We use the same class to describe rowReferenceMatrix elements. A rowReferenceMatrix is obtained by combining objReferenceMatrix elements and conReferenceMatrix elements into a single matrix constructor.

Definition at line 453 of file OSMatrix.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ConReferenceMatrixElement::ConReferenceMatrixElement ()

6.11.2.2 ConReferenceMatrixElement::~~ConReferenceMatrixElement ()

6.11.3 Member Function Documentation

6.11.3.1 bool ConReferenceMatrixElement::isEqual (ConReferenceMatrixElement * *that*)

A function to check for the equality of two objects.

6.11.3.2 bool ConReferenceMatrixElement::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.11.3.3 bool ConReferenceMatrixElement::deepCopyFrom (ConReferenceMatrixElement * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.11.4 Member Data Documentation

6.11.4.1 int ConReferenceMatrixElement::conReference

contains a reference to a row of the problem (objective if negative, constraint otherwise)

Remarks

If used in a ConReferenceMatrix, the nonnegativity required is verified and enforced

Definition at line 460 of file OSMatrix.h.

6.11.4.2 ENUM_CONREFERENCE_VALUETYPE ConReferenceMatrixElement::valueType

Several different types of values can be derived from a problem constraint.

(See [OSParameters.h](#) for an enumeration.)

Definition at line 466 of file OSMatrix.h.

6.11.4.3 double ConReferenceMatrixElement::value

This element contains the value.

Definition at line 469 of file OSMatrix.h.

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

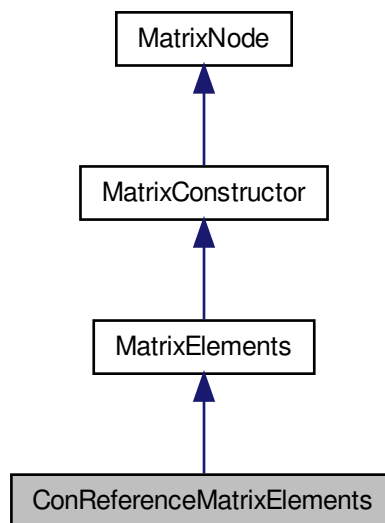
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.12 ConReferenceMatrixElements Class Reference

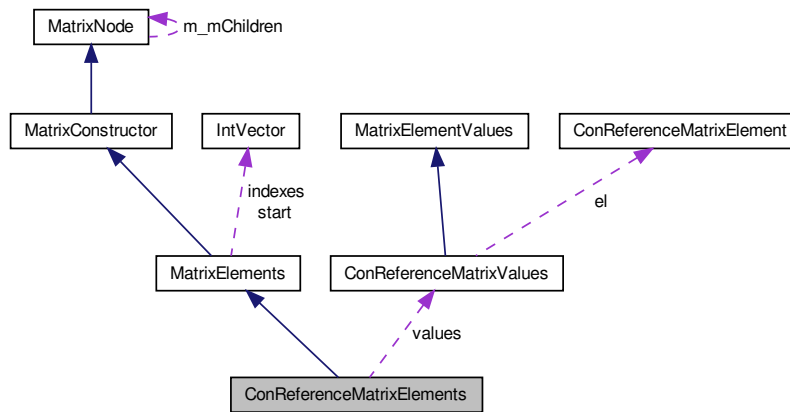
a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a constraint

```
#include <OSMatrix.h>
```

Inheritance diagram for ConReferenceMatrixElements:



Collaboration diagram for ConReferenceMatrixElements:



Public Member Functions

- [ConReferenceMatrixElements](#) ()
- [~ConReferenceMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [ConReferenceMatrixElements](#) * [cloneMatrixNode](#) ()
- bool [isEqual](#) ([ConReferenceMatrixElements](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([ConReferenceMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ConReferenceMatrixValues](#) * [values](#)
The constraint references (indexes of core constraints and value types) of the elements.

6.12.1 Detailed Description

a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a constraint

Definition at line 1178 of file OSMatrix.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 ConReferenceMatrixElements::ConReferenceMatrixElements ()

6.12.2.2 ConReferenceMatrixElements::~~ConReferenceMatrixElements ()

6.12.3 Member Function Documentation

6.12.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE ConReferenceMatrixElements::getNodeType () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.12.3.2 virtual ENUM_MATRIX_TYPE ConReferenceMatrixElements::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.12.3.3 virtual std::string ConReferenceMatrixElements::getNodeName () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.12.3.4 virtual std::string ConReferenceMatrixElements::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.Implements [MatrixNode](#).6.12.3.5 virtual bool ConReferenceMatrixElements::alignsOnBlockBoundary (int *firstRow*, int *firstColumn*, int *nRows*, int *nCols*) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.12.3.6 virtual ConReferenceMatrixElements* ConReferenceMatrixElements::cloneMatrixNode () [virtual]

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.12.3.7 bool ConReferenceMatrixElements::isEqual (ConReferenceMatrixElements * that)

A function to check for the equality of two objects.

6.12.3.8 bool ConReferenceMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.12.3.9 bool ConReferenceMatrixElements::deepCopyFrom (ConReferenceMatrixElements * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.12.4 Member Data Documentation

6.12.4.1 ConReferenceMatrixValues* ConReferenceMatrixElements::values

The constraint references (indexes of core constraints and value types) of the elements.

Definition at line 1182 of file OSMatrix.h.

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

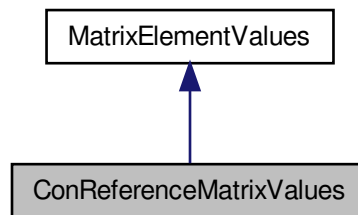
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.13 ConReferenceMatrixValues Class Reference

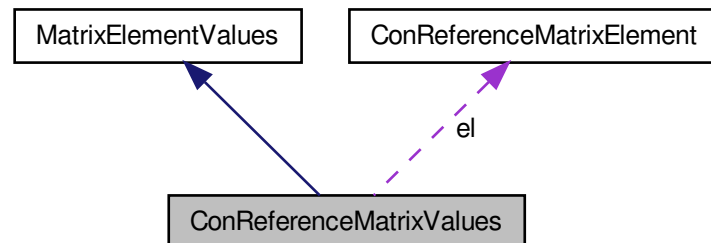
a data structure to represent the nonzeros in a conReferenceMatrix element

```
#include <OSMatrix.h>
```

Inheritance diagram for ConReferenceMatrixValues:



Collaboration diagram for ConReferenceMatrixValues:



Public Member Functions

- `ConReferenceMatrixValues` ()
- `~ConReferenceMatrixValues` ()
- `bool IsEqual (ConReferenceMatrixValues *that)`
A function to check for the equality of two objects.
- `bool setRandom` (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- `bool deepCopyFrom (ConReferenceMatrixValues *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- [ConReferenceMatrixElement](#) ** *el*

el contains the indices of the matrix constraints along with the *valueType*.

6.13.1 Detailed Description

a data structure to represent the nonzeros in a `conReferenceMatrix` element

Definition at line 712 of file `OSMatrix.h`.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `ConReferenceMatrixValues::ConReferenceMatrixValues ()`

6.13.2.2 `ConReferenceMatrixValues::~~ConReferenceMatrixValues ()`

6.13.3 Member Function Documentation

6.13.3.1 `bool ConReferenceMatrixValues::isEqual (ConReferenceMatrixValues * that)`

A function to check for the equality of two objects.

6.13.3.2 `bool ConReferenceMatrixValues::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.13.3.3 `bool ConReferenceMatrixValues::deepCopyFrom (ConReferenceMatrixValues * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.13.4 Member Data Documentation

6.13.4.1 `ConReferenceMatrixElement** ConReferenceMatrixValues::el`

el contains the indices of the matrix constraints along with the *valueType*.

Definition at line 718 of file `OSMatrix.h`.

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

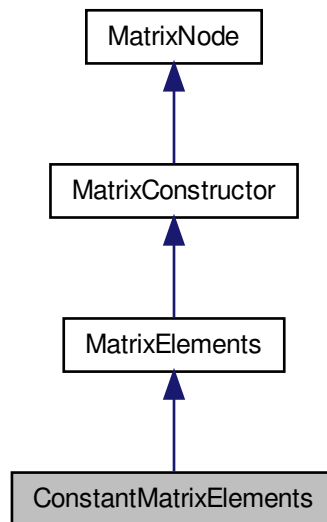
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.14 ConstantMatrixElements Class Reference

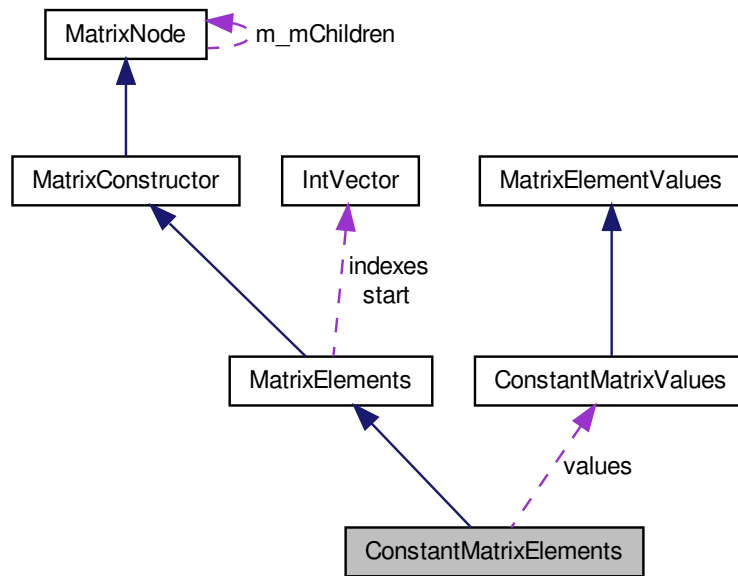
a data structure to represent the constant elements in a [MatrixType](#) object

```
#include <OSMatrix.h>
```

Inheritance diagram for ConstantMatrixElements:



Collaboration diagram for ConstantMatrixElements:



Public Member Functions

- [ConstantMatrixElements](#) ()
- [~ConstantMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) getNodeType ()
- virtual std::string getNodeName ()
- virtual [ENUM_MATRIX_TYPE](#) getMatrixType ()
- virtual std::string getMatrixNodeInXML ()
- virtual bool alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [ConstantMatrixElements](#) * cloneMatrixNode ()
- bool isEqual ([ConstantMatrixElements](#) *that)
A function to check for the equality of two objects.
- bool setRandom (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool deepCopyFrom ([ConstantMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ConstantMatrixValues](#) * values
The values of the (nonzero) constant elements.

6.14.1 Detailed Description

a data structure to represent the constant elements in a [MatrixType](#) object

Definition at line 750 of file OSMatrix.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ConstantMatrixElements::ConstantMatrixElements ()

6.14.2.2 ConstantMatrixElements::~~ConstantMatrixElements ()

6.14.3 Member Function Documentation

6.14.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE ConstantMatrixElements::getNodeType () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.14.3.2 virtual std::string ConstantMatrixElements::getNodeName () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.14.3.3 virtual ENUM_MATRIX_TYPE ConstantMatrixElements::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.14.3.4 virtual std::string ConstantMatrixElements::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.14.3.5 virtual bool ConstantMatrixElements::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.14.3.6 virtual ConstantMatrixElements* ConstantMatrixElements::cloneMatrixNode () [virtual]

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.14.3.7 bool ConstantMatrixElements::isEqual (ConstantMatrixElements * that)

A function to check for the equality of two objects.

6.14.3.8 bool ConstantMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.14.3.9 bool ConstantMatrixElements::deepCopyFrom (ConstantMatrixElements * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.14.4 Member Data Documentation

6.14.4.1 ConstantMatrixValues* ConstantMatrixElements::values

The values of the (nonzero) constant elements.

Definition at line 754 of file OSMatrix.h.

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

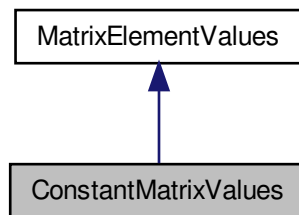
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.15 ConstantMatrixValues Class Reference

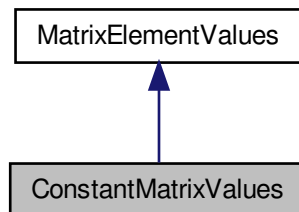
to represent the nonzeros in a constantMatrix element

```
#include <OSMatrix.h>
```

Inheritance diagram for ConstantMatrixValues:



Collaboration diagram for ConstantMatrixValues:



Public Member Functions

- [ConstantMatrixValues](#) ()
- [~ConstantMatrixValues](#) ()
- `bool` [IsEqual](#) ([ConstantMatrixValues](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([ConstantMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- double * [el](#)

6.15.1 Detailed Description

to represent the nonzeros in a constantMatrix element

Definition at line 501 of file OSMatrix.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ConstantMatrixValues::ConstantMatrixValues ()

6.15.2.2 ConstantMatrixValues::~~ConstantMatrixValues ()

6.15.3 Member Function Documentation

6.15.3.1 bool ConstantMatrixValues::isEqual (ConstantMatrixValues * *that*)

A function to check for the equality of two objects.

Returns

the value of nType
the type of the matrix elements
the name of the matrix constructor

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

6.15.3.2 bool ConstantMatrixValues::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.15.3.3 bool ConstantMatrixValues::deepCopyFrom (ConstantMatrixValues * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.15.4 Member Data Documentation**6.15.4.1 double* ConstantMatrixValues::el**

Definition at line 504 of file OSMatrix.h.

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

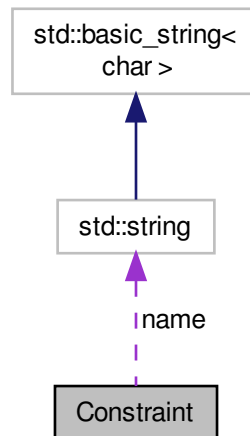
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.16 Constraint Class Reference

The in-memory representation of the <con> element.

```
#include <OSInstance.h>
```

Collaboration diagram for Constraint:

**Public Member Functions**

- `Constraint ()`
The *Constraint* class constructor.
- `~Constraint ()`
The *Constraint* class destructor.
- `bool IsEqual (Constraint *that)`
A function to check for the equality of two objects.

Public Attributes

- `std::string name`
name is the name of the constraint
- `double constant`
constant is a value that is added to the constraint
- `double lb`
lb is the lower bound on the constraint
- `double ub`
ub is the upper bound on the constraint

6.16.1 Detailed Description

The in-memory representation of the `<con>` element.
Definition at line 218 of file `OSInstance.h`.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 `Constraint::Constraint ()`

The `Constraint` class constructor.

6.16.2.2 `Constraint::~~Constraint ()`

The `Constraint` class destructor.

6.16.3 Member Function Documentation

6.16.3.1 `bool Constraint::IsEqual (Constraint * that)`

A function to check for the equality of two objects.

6.16.4 Member Data Documentation

6.16.4.1 `std::string Constraint::name`

`name` is the name of the constraint
Definition at line 229 of file `OSInstance.h`.

6.16.4.2 `double Constraint::constant`

`constant` is a value that is added to the constraint
Definition at line 232 of file `OSInstance.h`.

6.16.4.3 `double Constraint::lb`

`lb` is the lower bound on the constraint
Definition at line 235 of file `OSInstance.h`.

6.16.4.4 double Constraint::ub

ub is the upper bound on the constraint

Definition at line 238 of file OSInstance.h.

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

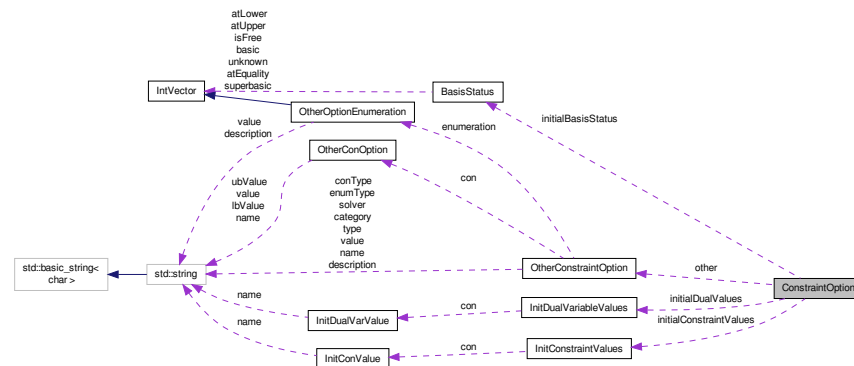
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h>

6.17 ConstraintOption Class Reference

the [ConstraintOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for ConstraintOption:



Public Member Functions

- [ConstraintOption](#) ()
Default constructor.
- [~ConstraintOption](#) ()
Class destructor.
- bool [isEqual](#) ([ConstraintOption](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([ConstraintOption](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setOther](#) (int numberOfOptions, [OtherConstraintOption](#) **other)
A function to set an array of <other> elements.
- bool [addOther](#) ([OtherConstraintOption](#) *other)
A function to add an <other> element.

Public Attributes

- int [numberOfOtherConstraintOptions](#)
number of <other> child elements
- [InitConstraintValues](#) * [initialConstraintValues](#)
initial values for the constraints
- [InitDualVariableValues](#) * [initialDualValues](#)
initial dual values for the constraints
- [BasisStatus](#) * [initialBasisStatus](#)
initial basis status for the slack variables
- [OtherConstraintOption](#) ** [other](#)
other information about the constraints

6.17.1 Detailed Description

the [ConstraintOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3263 of file OSOption.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 [ConstraintOption::ConstraintOption \(\)](#)

Default constructor.

6.17.2.2 [ConstraintOption::~~ConstraintOption \(\)](#)

Class destructor.

6.17.3 Member Function Documentation

6.17.3.1 [bool ConstraintOption::isEqual \(\[ConstraintOption\]\(#\) * *that* \)](#)

A function to check for the equality of two objects.

6.17.3.2 `bool ConstraintOption::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.17.3.3 `bool ConstraintOption::deepCopyFrom (ConstraintOption * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.17.3.4 `bool ConstraintOption::setOther (int numberOfOptions, OtherConstraintOption ** other)`

A function to set an array of <other> elements.

Parameters

<i>numberOfOptions</i> ,:	number of <other> elements to be set
<i>other</i> ,:	the array of <other> elements that are to be set

6.17.3.5 `bool ConstraintOption::addOther (OtherConstraintOption * other)`

A function to add an <other> element.

Parameters

<i>other</i> ,:	the content of the <other> element to be added
-----------------	--

6.17.4 Member Data Documentation

6.17.4.1 `int ConstraintOption::numberOfOtherConstraintOptions`

number of <other> child elements

Definition at line 3268 of file OSOption.h.

6.17.4.2 `InitConstraintValues* ConstraintOption::initialConstraintValues`

initial values for the constraints

Definition at line 3271 of file OSOption.h.

6.17.4.3 InitDualVariableValues* ConstraintOption::initialDualValues

initial dual values for the constraints

Definition at line 3274 of file OSOption.h.

6.17.4.4 BasisStatus* ConstraintOption::initialBasisStatus

initial basis status for the slack variables

Definition at line 3277 of file OSOption.h.

6.17.4.5 OtherConstraintOption** ConstraintOption::other

other information about the constraints

Definition at line 3280 of file OSOption.h.

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

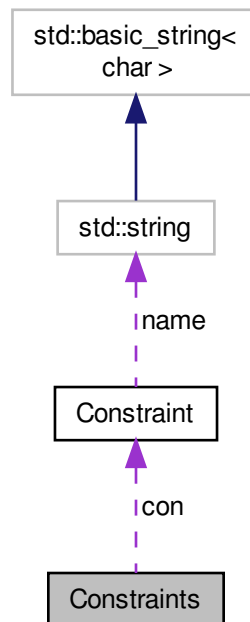
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.18 Constraints Class Reference

The in-memory representation of the <**constraints**> element.

```
#include <OSInstance.h>
```

Collaboration diagram for Constraints:



Public Member Functions

- [Constraints](#) ()
The [Constraints](#) class constructor.
- [~Constraints](#) ()
The [Constraints](#) class destructor.
- bool [IsEqual](#) ([Constraints](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfConstraints](#)
numberOfConstraints is the number of constraints in the instance
- [Constraint](#) ** [con](#)
con is pointer to an array of [Constraint](#) object pointers

6.18.1 Detailed Description

The in-memory representation of the <**constraints**> element.

Definition at line 251 of file OSInstance.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Constraints::Constraints ()

The [Constraints](#) class constructor.

6.18.2.2 Constraints::~~Constraints ()

The [Constraints](#) class destructor.

6.18.3 Member Function Documentation

6.18.3.1 bool Constraints::IsEqual (Constraints * that)

A function to check for the equality of two objects.

6.18.4 Member Data Documentation

6.18.4.1 int Constraints::numberOfConstraints

numberOfConstraints is the number of constraints in the instance

Definition at line 264 of file OSInstance.h.

6.18.4.2 Constraint** Constraints::con

con is pointer to an array of [Constraint](#) object pointers

Definition at line 268 of file OSInstance.h.

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

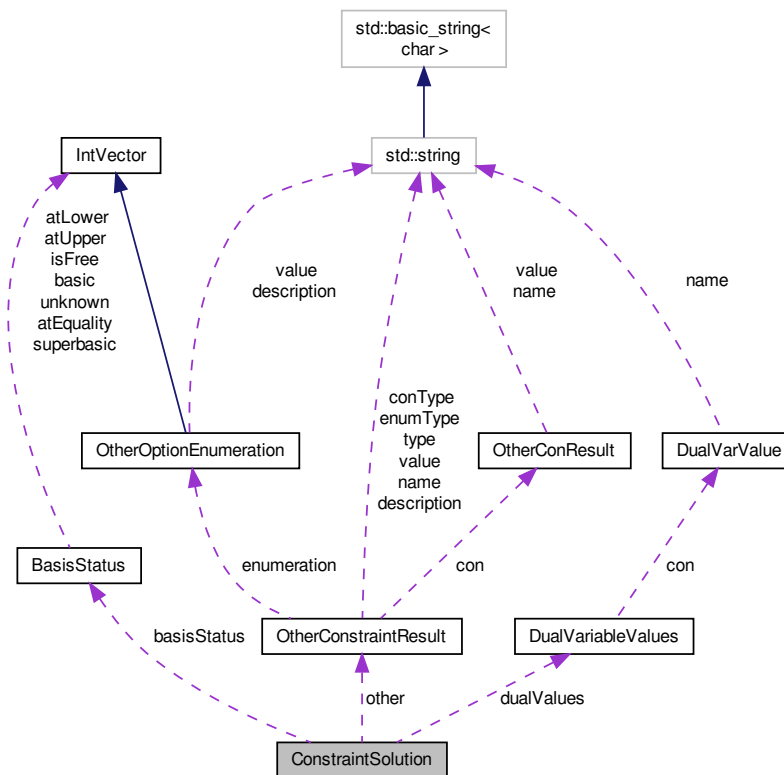
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.19 ConstraintSolution Class Reference

The [ConstraintSolution](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for ConstraintSolution:



Public Member Functions

- [ConstraintSolution](#) ()
Default constructor.
- [~ConstraintSolution](#) ()
Class destructor.
- [bool IsEqual](#) ([ConstraintSolution](#) *that)
A function to check for the equality of two objects.
- [bool setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `int numberOfOtherConstraintResults`
the number of types of constraint function results other than the basic constraint function values
- `DualVariableValues * dualValues`
a pointer to an array of [DualVariableValues](#) objects
- `BasisStatus * basisStatus`
a pointer to a [BasisStatus](#) object
- `OtherConstraintResult ** other`
a pointer to an array of other pointer objects for constraint functions

6.19.1 Detailed Description

The [ConstraintSolution](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting all of the types of solution values associated with objective functions.

Definition at line 1841 of file OSResult.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 `ConstraintSolution::ConstraintSolution ()`

Default constructor.

6.19.2.2 `ConstraintSolution::~~ConstraintSolution ()`

Class destructor.

6.19.3 Member Function Documentation

6.19.3.1 `bool ConstraintSolution::IsEqual (ConstraintSolution * that)`

A function to check for the equality of two objects.

6.19.3.2 bool ConstraintSolution::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.19.4 Member Data Documentation

6.19.4.1 int ConstraintSolution::numberOfOtherConstraintResults

the number of types of constraint function results other than the basic constraint function values

Definition at line 1849 of file OSResult.h.

6.19.4.2 DualVariableValues* ConstraintSolution::dualValues

a pointer to an array of [DualVariableValues](#) objects

Definition at line 1852 of file OSResult.h.

6.19.4.3 BasisStatus* ConstraintSolution::basisStatus

a pointer to a [BasisStatus](#) object

Definition at line 1855 of file OSResult.h.

6.19.4.4 OtherConstraintResult** ConstraintSolution::other

a pointer to an array of other pointer objects for constraint functions

Definition at line 1860 of file OSResult.h.

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

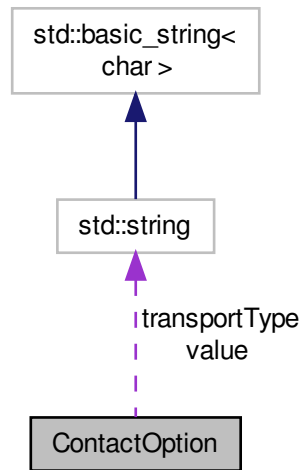
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.20 ContactOption Class Reference

the [ContactOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for ContactOption:



Public Member Functions

- [ContactOption](#) ()
Default constructor.
- [~ContactOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([ContactOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([ContactOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- `std::string` [transportType](#)
the contact mechanism
- `std::string` [value](#)
the value of the <contact> element

6.20.1 Detailed Description

the [ContactOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to the contact method in the OSoL schema.

Definition at line 96 of file OSOption.h.

6.20.2 Constructor & Destructor Documentation**6.20.2.1 ContactOption::ContactOption ()**

Default constructor.

6.20.2.2 ContactOption::~~ContactOption ()

Class destructor.

6.20.3 Member Function Documentation**6.20.3.1 bool ContactOption::isEqual (ContactOption * *that*)**

A function to check for the equality of two objects.

6.20.3.2 bool ContactOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.20.3.3 bool ContactOption::deepCopyFrom (ContactOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.20.4 Member Data Documentation**6.20.4.1 `std::string ContactOption::transportType`**

the contact mechanism

Definition at line 101 of file `OSOption.h`.

6.20.4.2 `std::string ContactOption::value`

the value of the `<contact>` element

Definition at line 104 of file `OSOption.h`.

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

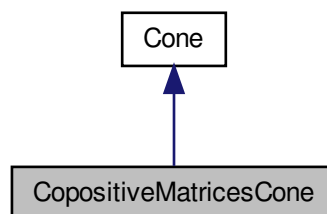
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h`

6.21 CopositiveMatricesCone Class Reference

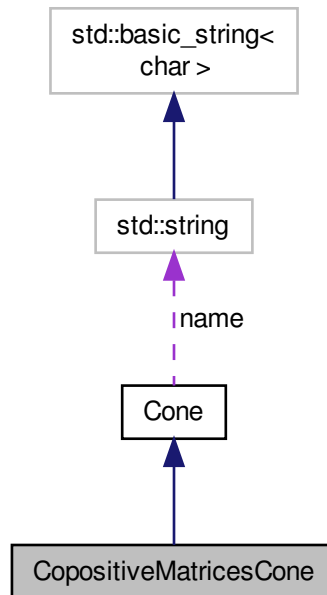
The [CopositiveMatricesCone](#) Class.

```
#include <OSInstance.h>
```

Inheritance diagram for `CopositiveMatricesCone`:



Collaboration diagram for CopositiveMatricesCone:



Public Member Functions

- [CopositiveMatricesCone](#) ()
default constructor.
- [~CopositiveMatricesCone](#) ()
default destructor.
- virtual `std::string` [getConeName](#) ()
- virtual `std::string` [getConeInXML](#) ()
Write a [CopositiveMatricesCone](#) object in XML format.
- `bool` [IsEqual](#) ([CopositiveMatricesCone](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([CopositiveMatricesCone](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

6.21.1 Detailed Description

The [CopositiveMatricesCone](#) Class.

Remarks

The in-memory representation of the OSiL element `<copositiveMatricesCone>`

Definition at line 1127 of file OSInstance.h.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 CopositiveMatricesCone::CopositiveMatricesCone ()

default constructor.

6.21.2.2 CopositiveMatricesCone::~~CopositiveMatricesCone ()

default destructor.

6.21.3 Member Function Documentation

6.21.3.1 virtual std::string CopositiveMatricesCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.21.3.2 virtual std::string CopositiveMatricesCone::getConeInXML () [virtual]

Write a [CopositiveMatricesCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a `<cone>` element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.21.3.3 bool CopositiveMatricesCone::IsEqual (CopositiveMatricesCone * that)

A function to check for the equality of two objects.

6.21.3.4 bool CopositiveMatricesCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <code><XXX></code> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.21.3.5 bool CpositiveMatricesCone::deepCopyFrom (CpositiveMatricesCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

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

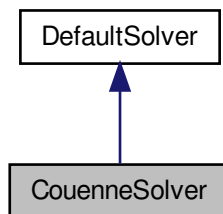
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.22 CouenneSolver Class Reference

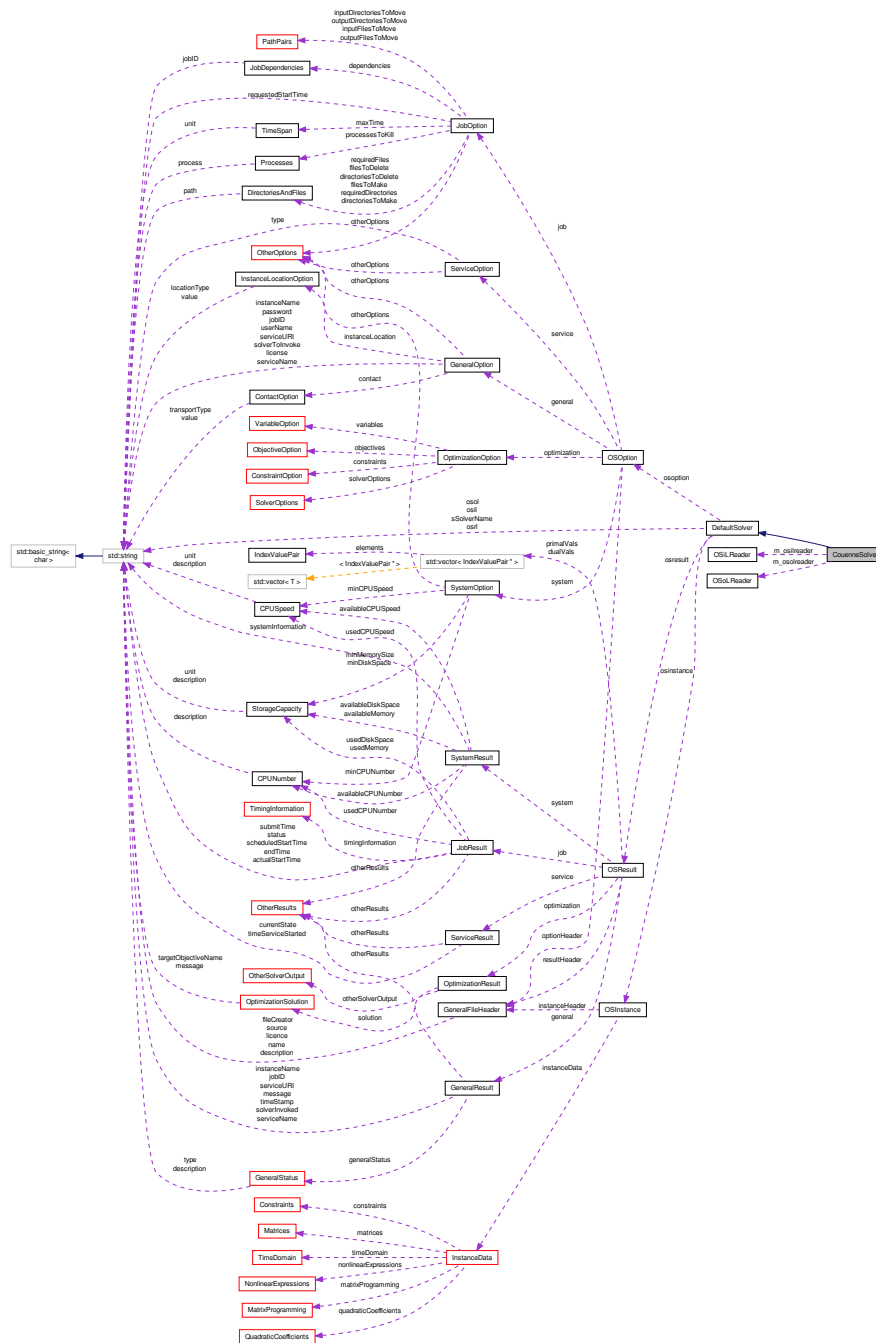
The [CouenneSolver](#) class solves problems using Ipopt.

```
#include <OSCouenneSolver.h>
```

Inheritance diagram for CouenneSolver:



Collaboration diagram for CouenneSolver:



Public Member Functions

- [CouenneSolver](#) ()
the *CouenneSolver* class constructor
- [~CouenneSolver](#) ()
the *IpoptSolver* class destructor

- virtual void [solve](#) () throw (ErrorClass)
solve results in an instance being read into the [Couenne](#) data structures and optimized
- virtual void [buildSolverInstance](#) () throw (ErrorClass)
buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API
- virtual void [setSolverOptions](#) () throw (ErrorClass)
The implementation of the virtual functions.
- void [dataEchoCheck](#) ()
use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file
- void [writeResult](#) ()
use this to write the solution information to an [OSResult](#) object

Public Attributes

- [OSiLReader](#) * [m_osilreader](#)
m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed
- [OSoLReader](#) * [m_osolreader](#)
m_osolreader is an [OSoLReader](#) object used to create an ooption from an osol string if needed
- Couenne::CouenneProblem * [couenne](#)
- Ipopt::SmartPtr< [BonminProblem](#) > [tminlp](#)
- Ipopt::SmartPtr
< Bonmin::TNLPSolver > [app_](#)
- Couenne::CouenneBab [bb](#)
- Bonmin::TMINLP::SolverReturn [status](#)
- Couenne::expression * [con_body](#)
- Couenne::expression * [obj_body](#)

6.22.1 Detailed Description

The [CouenneSolver](#) class solves problems using Ipopt.

Author

Jun Ma, Horand Gassmann, Kipp Martin

Version

1.0, 07/05/2008

Since

OS 1.0

Remarks

this class takes an OSiL instance and optimizes it using the COIN-OR Ipopt solver

Definition at line 67 of file OSCouenneSolver.h.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 CouenneSolver::CouenneSolver ()

the [CouenneSolver](#) class constructor

6.22.2.2 CouenneSolver::~~CouenneSolver ()

the [lpoptSolver](#) class destructor

6.22.3 Member Function Documentation

6.22.3.1 virtual void CouenneSolver::solve () throw (ErrorClass) [virtual]

solve results in an instance being read into the [Couenne](#) data structures and optimized

Implements [DefaultSolver](#).

6.22.3.2 virtual void CouenneSolver::buildSolverInstance () throw (ErrorClass) [virtual]

buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API

Implements [DefaultSolver](#).

6.22.3.3 void CouenneSolver::setSolverOptions () throw (ErrorClass) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.22.3.4 void CouenneSolver::dataEchoCheck ()

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

6.22.3.5 void CouenneSolver::writeResult ()

use this to write the solution information to an [OSResult](#) object

6.22.4 Member Data Documentation

6.22.4.1 OSiLReader* CouenneSolver::m_osilreader

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 110 of file OSCouenneSolver.h.

6.22.4.2 OSoLReader* CouenneSolver::m_osolreader

m_osolreader is an [OSoLReader](#) object used to create an osoption from an osol string if needed

Definition at line 116 of file OSCouenneSolver.h.

6.22.4.3 Couenne::CouenneProblem* CouenneSolver::couenne

Definition at line 118 of file OSCouenneSolver.h.

6.22.4.4 Ipopt::SmartPtr<BonminProblem> CouenneSolver::tminlp

Definition at line 120 of file OSCouenneSolver.h.

6.22.4.5 Ipopt::SmartPtr<Bonmin::TNLPSolver> CouenneSolver::app_

Definition at line 122 of file OSCouenneSolver.h.

6.22.4.6 Couenne::CouenneBab CouenneSolver::bb

Definition at line 127 of file OSCouenneSolver.h.

6.22.4.7 Bonmin::TMINLP::SolverReturn CouenneSolver::status

Definition at line 129 of file OSCouenneSolver.h.

6.22.4.8 Couenne::expression* CouenneSolver::con_body

Definition at line 131 of file OSCouenneSolver.h.

6.22.4.9 Couenne::expression* CouenneSolver::obj_body

Definition at line 132 of file OSCouenneSolver.h.

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

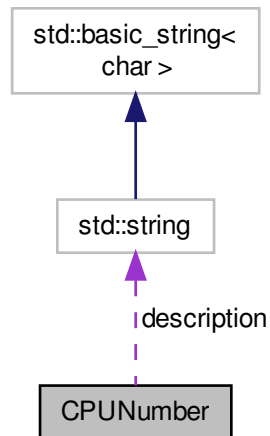
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCouenneSolver.h](#)

6.23 CPUNumber Class Reference

the [CPUNumber](#) class.

```
#include <OSGeneral.h>
```

Collaboration diagram for CPUNumber:



Public Member Functions

- `CPUNumber ()`
Default constructor.
- `~CPUNumber ()`
Class destructor.
- `bool isEqual (CPUNumber *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (CPUNumber *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `std::string description`
additional description about the CPU
- `int value`
the number of CPUs

6.23.1 Detailed Description

the `CPUNumber` class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSgL schema.

Definition at line 871 of file OSGeneral.h.

6.23.2 Constructor & Destructor Documentation**6.23.2.1 CPUNumber::CPUNumber ()**

Default constructor.

6.23.2.2 CPUNumber::~~CPUNumber ()

Class destructor.

6.23.3 Member Function Documentation**6.23.3.1 bool CPUNumber::isEqual (CPUNumber * *that*)**

A function to check for the equality of two objects.

6.23.3.2 bool CPUNumber::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.23.3.3 bool CPUNumber::deepCopyFrom (CPUNumber * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.23.4 Member Data Documentation**6.23.4.1 std::string CPUNumber::description**

additional description about the CPU

Definition at line 876 of file OSGeneral.h.

6.23.4.2 int CPUNumber::value

the number of CPUs

Definition at line 879 of file OSGeneral.h.

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

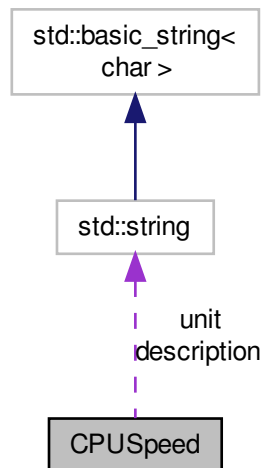
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.24 CPUSpeed Class Reference

the [CPUSpeed](#) class.

```
#include <OSGeneral.h>
```

Collaboration diagram for CPUSpeed:

**Public Member Functions**

- [CPUSpeed](#) ()

Default constructor.

- `~CPUSpeed ()`

Class destructor.

- `bool isEqual (CPUSpeed *that)`

A function to check for the equality of two objects.

- `bool setRandom (double density, bool conformant)`

A function to make a random instance of this class.

- `bool deepCopyFrom (CPUSpeed *that)`

A function to make a deep copy of an instance of this class.

Public Attributes

- `std::string unit`

the unit in which CPU speed is measured

- `std::string description`

additional description about the CPU speed

- `double value`

the CPU speed (expressed in multiples of unit)

6.24.1 Detailed Description

the `CPUSpeed` class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSgL schema.

Definition at line 812 of file OSGeneral.h.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 CPUSpeed::CPUSpeed ()

Default constructor.

6.24.2.2 CPUSpeed::~~CPUSpeed ()

Class destructor.

6.24.3 Member Function Documentation

6.24.3.1 `bool CPUSpeed::isEqual (CPUSpeed * that)`

A function to check for the equality of two objects.

6.24.3.2 `bool CPUSpeed::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.24.3.3 `bool CPUSpeed::deepCopyFrom (CPUSpeed * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.24.4 Member Data Documentation

6.24.4.1 `std::string CPUSpeed::unit`

the unit in which CPU speed is measured

Definition at line 817 of file OSGeneral.h.

6.24.4.2 `std::string CPUSpeed::description`

additional description about the CPU speed

Definition at line 820 of file OSGeneral.h.

6.24.4.3 `double CPUSpeed::value`

the CPU speed (expressed in multiples of unit)

Definition at line 823 of file OSGeneral.h.

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

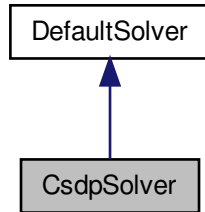
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.25 CsdpSolver Class Reference

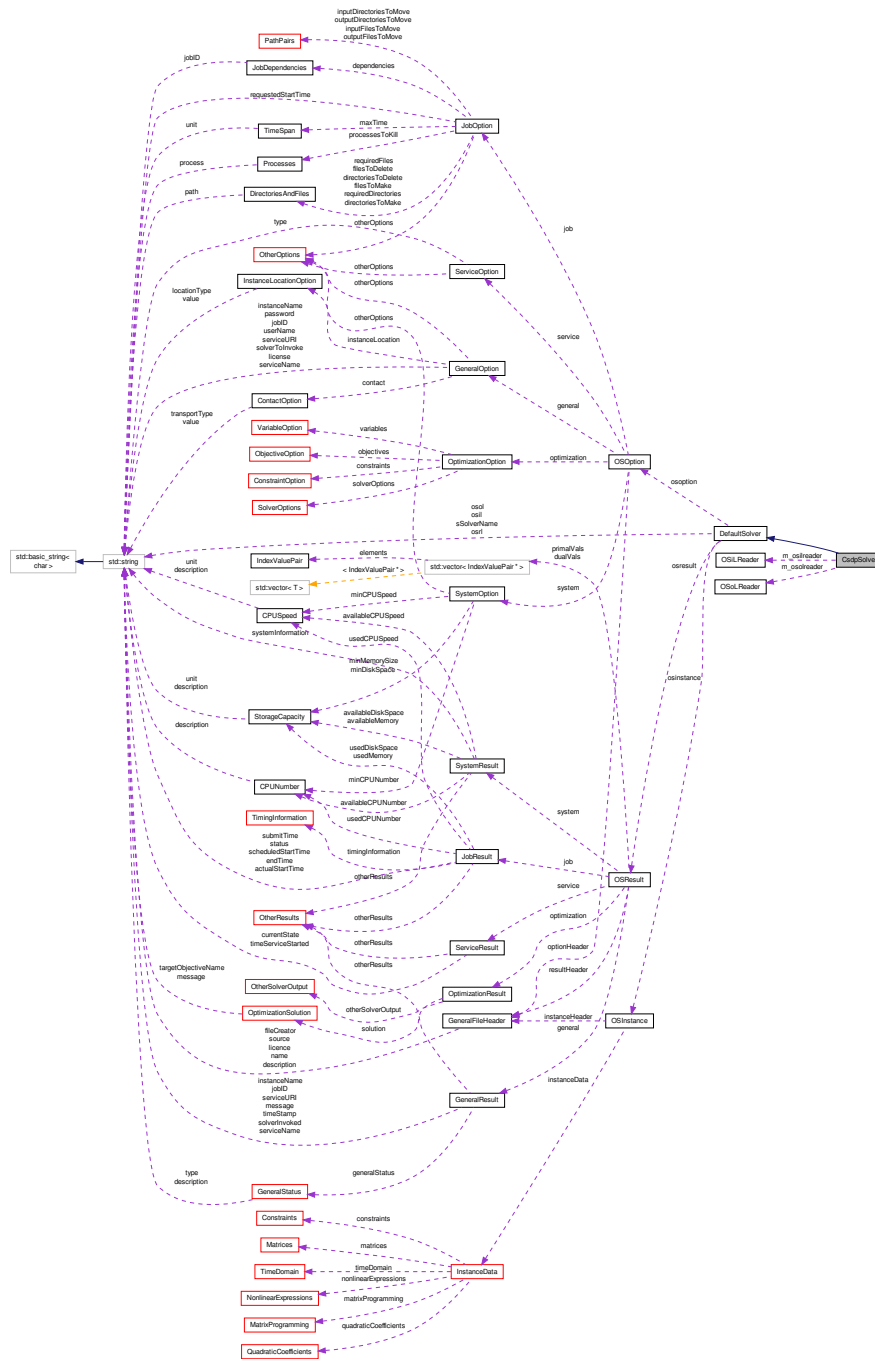
The [CsdpSolver](#) class solves problems using Csdp.

```
#include <OSCsdpSolver.h>
```

Inheritance diagram for CsdpSolver:



Collaboration diagram for CsdpSolver:



Public Member Functions

- [CsdpSolver](#) ()
the *CsdpSolver* class constructor
- [~CsdpSolver](#) ()

the [CsdpSolver](#) class destructor

- virtual void [solve](#) () throw (ErrorClass)

solve results in an instance being read into the Csdp data structures and optimize

- virtual void [buildSolverInstance](#) () throw (ErrorClass)

The implementation of the virtual functions.

- virtual void [setSolverOptions](#) () throw (ErrorClass)

The implementation of the virtual functions.

- void [verifyForm](#) () throw (ErrorClass)

Use to verify that the solver is appropriate CSDP requires a very special type of problem.

- void [dataEchoCheck](#) ()

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

Public Attributes

- [OSiLReader](#) * [m_osilreader](#)

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

- [OSoLReader](#) * [m_osolreader](#)

m_osolreader is an [OSoLReader](#) object used to create an ooption from an osol string if needed

6.25.1 Detailed Description

The [CsdpSolver](#) class solves problems using Csdp.

Author

Jun Ma, Kipp Martin, Horand Gassmann

Version

1.0, 05/26/2014

Since

OS 2.8

Remarks

this class takes an OSiL instance and optimizes it using the COIN-OR Csdp solver

Definition at line 173 of file OSCsdpSolver.h.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 [CsdpSolver::CsdpSolver](#) ()

the [CsdpSolver](#) class constructor

6.25.2.2 [CsdpSolver::~~CsdpSolver](#) ()

the [CsdpSolver](#) class destructor

6.25.3 Member Function Documentation

6.25.3.1 `virtual void CsdpSolver::solve () throw (ErrorClass) [virtual]`

solve results in an instance being read into the Csdp data structures and optimize

Implements [DefaultSolver](#).

6.25.3.2 `void CsdpSolver::buildSolverInstance () throw (ErrorClass) [virtual]`

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.25.3.3 `void CsdpSolver::setSolverOptions () throw (ErrorClass) [virtual]`

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.25.3.4 `CsdpSolver::verifyForm () throw (ErrorClass)`

Use to verify that the solver is appropriate CSDP requires a very special type of problem.

Returns

void.

6.25.3.5 `void CsdpSolver::dataEchoCheck ()`

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

6.25.4 Member Data Documentation

6.25.4.1 `OSiLReader* CsdpSolver::m_osilreader`

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 217 of file OSCsdpSolver.h.

6.25.4.2 `OSoLReader* CsdpSolver::m_osolreader`

m_osolreader is an [OSoLReader](#) object used to create an osoption from an osol string if needed

Definition at line 223 of file OSCsdpSolver.h.

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

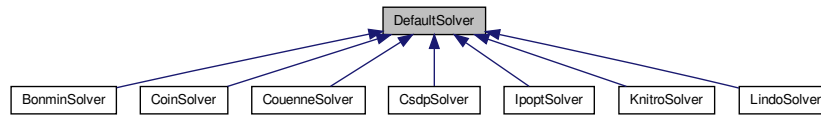
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCsdpSolver.h](#)

6.26 DefaultSolver Class Reference

The Default Solver Class.

```
#include <OSDefaultSolver.h>
```

Inheritance diagram for DefaultSolver:



The diagram illustrates the relationships between various options and results in the CPLEX API. It is structured as a dependency graph where nodes represent different types of objects, and dashed purple arrows indicate dependencies or relationships.

- Options (Left Column):** Includes `ContactOption`, `InstanceLocationOption`, `ConstraintOption`, `SolveOptions`, `VariableOption`, `ObjectiveOption`, `TimeSpan`, `Process`, `PathPairs`, `JobDependencies`, `OtherOptions`, `IndexValuePair`, `CPUSpeed`, `CPUNumber`, `StorageCapacity`, `TimingInformation`, `OtherResults`, `OptimizationSolution`, `OtherSolveOutput`, `GeneralStatus`, `Objectives`, `Cones`, `LinearConstraintCoefficients`, `TimeDomain`, `NonlinearExpressions`, `MainProgramming`, `QuadraticCoefficients`, `Constraints`, and `Matrices`.
- Options (Right Column):** Includes `GeneralOption`, `OptimizationOption`, `JobOption`, `ServiceOption`, `OSOption`, `SystemOption`, `GeneralFileHeader`, `SystemResult`, `JobResult`, `ServiceResult`, `OptimizationResult`, and `GeneralResult`.
- Results (Bottom Row):** Includes `InstanceData`, `LinearConstraintCoefficients`, `TimeDomain`, `NonlinearExpressions`, `MainProgramming`, `QuadraticCoefficients`, `Constraints`, and `Matrices`.
- Intermediate Results (Middle Columns):** Includes `InstanceLocation`, `Constraints`, `SolveOptions`, `Variables`, `Objectives`, `MaxTime`, `ProcessesToKill`, `OtherOptions`, `RequiredFlags`, `InputDirectoriesToMove`, `OutputDirectoriesToMove`, `InputFilesToMove`, `OutputFilesToMove`, `Dependencies`, `OtherOptions`, `ServiceOptions`, `OSOptions`, `SystemOptions`, `GeneralFileHeader`, `SystemResult`, `JobResult`, `ServiceResult`, `OptimizationResult`, and `GeneralResult`.
- Central Node:** A box labeled "set string" is connected to many other nodes via dashed purple arrows.
- Connections:** Dashed purple arrows show dependencies between nodes. For example, `InstanceLocationOption` depends on `InstanceLocation`. `GeneralOption` depends on `GeneralOption`. `OptimizationOption` depends on `OptimizationOption`. `JobOption` depends on `JobOption`. `ServiceOption` depends on `ServiceOption`. `OSOption` depends on `OSOption`. `SystemOption` depends on `SystemOption`. `GeneralFileHeader` depends on `GeneralFileHeader`. `SystemResult` depends on `SystemResult`. `JobResult` depends on `JobResult`. `ServiceResult` depends on `ServiceResult`. `OptimizationResult` depends on `OptimizationResult`. `GeneralResult` depends on `GeneralResult`.

- virtual void `solve` ()=0
solve is a virtual function – the actual solvers will implement their own solve method

- virtual void `buildSolverInstance` ()=0
buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API
- virtual void `setSolverOptions` ()=0
setSolverOptions is a virtual function – the actual solvers will implement their own setSolverOptions method – the solver options are the options the individual solver sees in its API
- `DefaultSolver` ()
default constructor.
- virtual `~DefaultSolver` ()=0
default destructor.

Public Attributes

- std::string `osil`
osil holds the problem instance as a std::string
- std::string `osol`
osol holds the options for the solver
- std::string `osrl`
osrl holds the solution or result of the model
- `OSInstance` * `osinstance`
osinstance holds the problem instance in-memory as an OSInstance object
- `OSOption` * `osoption`
osoption holds the solver options in-memory as an OSOption object
- `OSResult` * `osresult`
osresult holds the solution or result of the model in-memory as an OSResult object
- std::string `sSolverName`
sSolverName is the name of the Coin solver used, e.g.
- bool `bCallbuildSolverInstance`
bCallbuildSolverInstance is set to true if buildSolverService has been called
- bool `bSetSolverOptions`
bSetSolverOptions is set to true if setSolverOptions has been called, false otherwise

6.26.1 Detailed Description

The Default Solver Class.

Author

Robert Fourer, Jun Ma, Kipp Martin,

Version

1.0, 10/05/2005

Since

OS1.0

Definition at line 35 of file OSDefaultSolver.h.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 DefaultSolver::DefaultSolver ()

default constructor.

6.26.2.2 virtual DefaultSolver::~~DefaultSolver () [pure virtual]

default destructor.

6.26.3 Member Function Documentation

6.26.3.1 virtual void DefaultSolver::solve () [pure virtual]

solve is a virtual function – the actual solvers will implement their own solve method

Implemented in [BonminSolver](#), [CsdpSolver](#), [IpoptSolver](#), [KnitroSolver](#), [CouenneSolver](#), [LindoSolver](#), and [CoinSolver](#).

6.26.3.2 virtual void DefaultSolver::buildSolverInstance () [pure virtual]

buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API

Implemented in [BonminSolver](#), [CsdpSolver](#), [IpoptSolver](#), [KnitroSolver](#), [CouenneSolver](#), [LindoSolver](#), and [CoinSolver](#).

6.26.3.3 virtual void DefaultSolver::setSolverOptions () [pure virtual]

setSolverOptions is a virtual function – the actual solvers will implement their own setSolverOptions method – the solver options are the options the individual solver sees in its API

Implemented in [BonminSolver](#), [CsdpSolver](#), [IpoptSolver](#), [KnitroSolver](#), [CouenneSolver](#), [LindoSolver](#), and [CoinSolver](#).

6.26.4 Member Data Documentation

6.26.4.1 std::string DefaultSolver::osil

osil holds the problem instance as a std::string

Definition at line 43 of file OSDefaultSolver.h.

6.26.4.2 std::string DefaultSolver::osol

osol holds the options for the solver

Definition at line 46 of file OSDefaultSolver.h.

6.26.4.3 std::string DefaultSolver::osrl

osrl holds the solution or result of the model

Definition at line 50 of file OSDefaultSolver.h.

6.26.4.4 OSInstance* DefaultSolver::osinstance

osinstance holds the problem instance in-memory as an [OSInstance](#) object

Definition at line 54 of file OSDefaultSolver.h.

6.26.4.5 `OSOption*` `DefaultSolver::osoption`

`osoption` holds the solver options in-memory as an [OSOption](#) object

Definition at line 58 of file `OSDefaultSolver.h`.

6.26.4.6 `OSResult*` `DefaultSolver::osresult`

`osresult` holds the solution or result of the model in-memory as an [OSResult](#) object

Definition at line 61 of file `OSDefaultSolver.h`.

6.26.4.7 `std::string` `DefaultSolver::sSolverName`

`sSolverName` is the name of the [Coin](#) solver used, e.g.

`glpk`, or `clp`

Definition at line 68 of file `OSDefaultSolver.h`.

6.26.4.8 `bool` `DefaultSolver::bCallbuildSolverInstance`

`bCallbuildSolverInstance` is set to true if `buildSolverService` has been called

Definition at line 75 of file `OSDefaultSolver.h`.

6.26.4.9 `bool` `DefaultSolver::bSetSolverOptions`

`bSetSolverOptions` is set to true if `setSolverOptions` has been called, false otherwise

Definition at line 82 of file `OSDefaultSolver.h`.

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

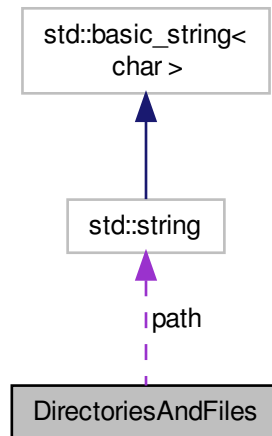
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSDefaultSolver.h`

6.27 DirectoriesAndFiles Class Reference

the [DirectoriesAndFiles](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for DirectoriesAndFiles:



Public Member Functions

- [DirectoriesAndFiles](#) ()
Default constructor.
- [~DirectoriesAndFiles](#) ()
Class destructor.
- bool [isEqual](#) ([DirectoriesAndFiles](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([DirectoriesAndFiles](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setPath](#) (int [numberOfPaths](#), std::string *path)
A function to set an array of <path> elements.
- bool [addPath](#) (std::string path)
A function to add a <path> element.

Public Attributes

- int [numberOfPaths](#)
the number of <path> children
- std::string * [path](#)
the list of directory and file paths

6.27.1 Detailed Description

the [DirectoriesAndFiles](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 780 of file OSOption.h.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 DirectoriesAndFiles::DirectoriesAndFiles ()

Default constructor.

6.27.2.2 DirectoriesAndFiles::~~DirectoriesAndFiles ()

Class destructor.

6.27.3 Member Function Documentation

6.27.3.1 bool DirectoriesAndFiles::isEqual (DirectoriesAndFiles * *that*)

A function to check for the equality of two objects.

6.27.3.2 bool DirectoriesAndFiles::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.27.3.3 bool DirectoriesAndFiles::deepCopyFrom (DirectoriesAndFiles * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.27.3.4 bool DirectoriesAndFiles::setPath (int *numberOfPaths*, std::string * *path*)

A function to set an array of <path> elements.

Parameters

<i>numberOfPaths,:</i>	number of <path> elements to be set
<i>path,:</i>	the array of <path> elements that are to be set

6.27.3.5 bool DirectoriesAndFiles::addPath (std::string *path*)

A function to add a <path> element.

Parameters

<i>path,:</i>	the path to be added
---------------	----------------------

6.27.4 Member Data Documentation

6.27.4.1 int DirectoriesAndFiles::numberOfPaths

the number of <path> children

Definition at line 785 of file OSOption.h.

6.27.4.2 std::string* DirectoriesAndFiles::path

the list of directory and file paths

Definition at line 788 of file OSOption.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.28 DoubleVector Class Reference

a double vector data structure

```
#include <OSGeneral.h>
```

Public Member Functions

- [DoubleVector](#) ()
- [~DoubleVector](#) ()
- bool [isEqual](#) ([DoubleVector](#) *that)

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- int [numberOfEl](#)
- double * [el](#)

6.28.1 Detailed Description

a double vector data structure

Definition at line 609 of file OSGeneral.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 DoubleVector::DoubleVector ()

6.28.2.2 DoubleVector::~~DoubleVector ()

6.28.3 Member Function Documentation

6.28.3.1 bool DoubleVector::isEqual (DoubleVector * *that*)

6.28.4 Member Data Documentation

6.28.4.1 bool DoubleVector::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 619 of file OSGeneral.h.

6.28.4.2 int DoubleVector::numberOfEl

Definition at line 620 of file OSGeneral.h.

6.28.4.3 double* DoubleVector::el

Definition at line 621 of file OSGeneral.h.

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

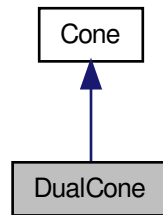
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.29 DualCone Class Reference

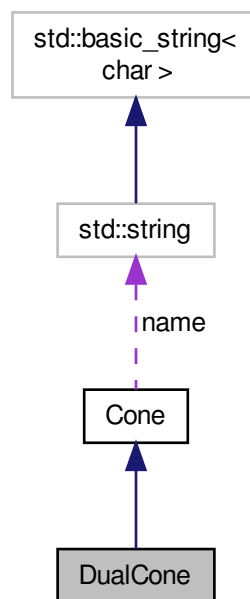
The in-memory representation of a dual cone.

```
#include <OSInstance.h>
```

Inheritance diagram for DualCone:



Collaboration diagram for DualCone:



Public Member Functions

- [DualCone](#) ()
The [DualCone](#) class constructor.
- [~DualCone](#) ()
The [DualCone](#) class destructor.

- virtual std::string [getConeName](#) ()
- bool [isEqual](#) ([DualCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([DualCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in ENUM_CONE_TYPE)
- int [idx](#)
cones are referenced by an (automatically created) index
- int [referenceConeIdx](#)
Dual cones use a reference to another, previously defined cone.

6.29.1 Detailed Description

The in-memory representation of a dual cone.

Definition at line 1401 of file OSInstance.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 [DualCone::DualCone](#) ()

The [DualCone](#) class constructor.

6.29.2.2 [DualCone::~~DualCone](#) ()

The [DualCone](#) class destructor.

6.29.3 Member Function Documentation

6.29.3.1 virtual std::string [DualCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.29.3.2 `bool DualCone::isEqual (DualCone * that)`

A function to check for the equality of two objects.

6.29.3.3 `bool DualCone::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.29.3.4 `bool DualCone::deepCopyFrom (DualCone * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.29.4 **Member Data Documentation****6.29.4.1** `int DualCone::numberOfRows`

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 1414 of file OSInstance.h.

6.29.4.2 `int DualCone::numberOfColumns`

Definition at line 1415 of file OSInstance.h.

6.29.4.3 `int DualCone::numberOfOtherIndexes`

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 1422 of file OSInstance.h.

6.29.4.4 `int* DualCone::otherIndexes`

Definition at line 1423 of file OSInstance.h.

6.29.4.5 `int DualCone::coneType`

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 1426 of file OSInstance.h.

6.29.4.6 int DualCone::idx

cones are referenced by an (automatically created) index

Definition at line 1429 of file OSInstance.h.

6.29.4.7 int DualCone::referenceConeldx

Dual cones use a reference to another, previously defined cone.

Definition at line 1432 of file OSInstance.h.

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

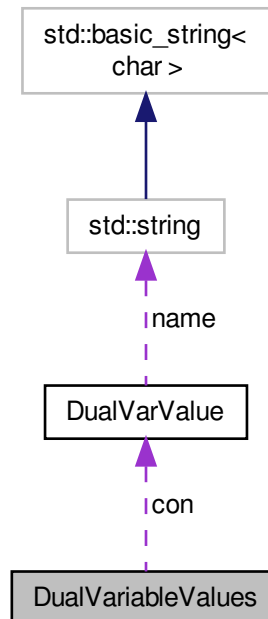
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.30 DualVariableValues Class Reference

The [DualVariableValues](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for DualVariableValues:



Public Member Functions

- [DualVariableValues](#) ()
Default constructor.
- [~DualVariableValues](#) ()
Class destructor.
- bool [isEqual](#) ([DualVariableValues](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [numberOfCon](#)
record the number of constraints for which values are given
- [DualVarValue](#) ** [con](#)
con is a vector of [DualVarValue](#) objects that give an index and dual variable value for each constraint function.

6.30.1 Detailed Description

The [DualVariableValues](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting dual variable values

Definition at line 1640 of file OSResult.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 [DualVariableValues::DualVariableValues](#) ()

Default constructor.

6.30.2.2 [DualVariableValues::~~DualVariableValues](#) ()

Class destructor.

6.30.3 Member Function Documentation

6.30.3.1 `bool DualVariableValues::isEqual (DualVariableValues * that)`

A function to check for the equality of two objects.

6.30.3.2 `bool DualVariableValues::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.30.4 Member Data Documentation

6.30.4.1 `int DualVariableValues::numberOfCon`

record the number of constraints for which values are given

Definition at line 1646 of file OSResult.h.

6.30.4.2 `DualVarValue** DualVariableValues::con`

con is a vector of [DualVarValue](#) objects that give an index and dual variable value for each constraint function.

Definition at line 1652 of file OSResult.h.

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

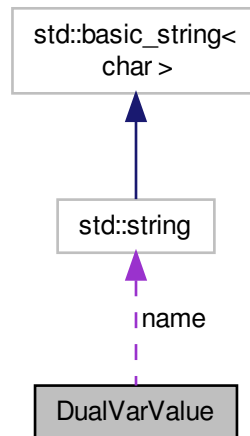
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.31 DualVarValue Class Reference

The [DualVarValue](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for DualVarValue:



Public Member Functions

- `DualVarValue ()`
Default constructor.
- `~DualVarValue ()`
Class destructor.
- `bool isEqual (DualVarValue *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int idx`
idx is the index on a constraint
- `std::string name`
optional name
- `double value`
value of dual variable on the constraint indexed by idx

6.31.1 Detailed Description

The `DualVarValue` Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the dual value of a constraint

Definition at line 1584 of file OSResult.h.

6.31.2 Constructor & Destructor Documentation**6.31.2.1 DualVarValue::DualVarValue ()**

Default constructor.

6.31.2.2 DualVarValue::~~DualVarValue ()

Class destructor.

6.31.3 Member Function Documentation**6.31.3.1 bool DualVarValue::isEqual (DualVarValue * *that*)**

A function to check for the equality of two objects.

6.31.3.2 bool DualVarValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.31.4 Member Data Documentation**6.31.4.1 int DualVarValue::idx**

idx is the index on a constraint

Definition at line 1589 of file OSResult.h.

6.31.4.2 `std::string DualVarValue::name`

optional name

Definition at line 1592 of file OSResult.h.

6.31.4.3 `double DualVarValue::value`

value of dual variable on the constraint indexed by idx

Definition at line 1597 of file OSResult.h.

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

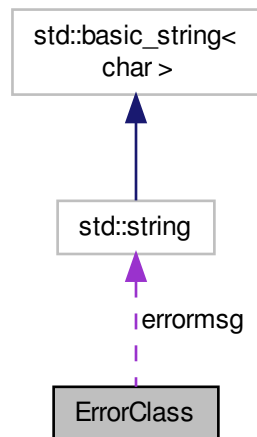
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.32 ErrorClass Class Reference

used for throwing exceptions.

```
#include <OSErrorClass.h>
```

Collaboration diagram for ErrorClass:



Public Member Functions

- [ErrorClass](#) (`std::string errmsg_`)
the class constructor

Public Attributes

- `std::string` [errmsg](#)
errmsg is the error that is causing the exception to be thrown

6.32.1 Detailed Description

used for throwing exceptions.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

when an error is encountered in a try-catch block we throw an [ErrorClass](#) with the errmsg_

Definition at line 31 of file OSErrClass.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 ErrorClass::ErrorClass (std::string errmsg_)

the class constructor

Parameters

<i>errmsg_</i>	holds the error message as a string.
----------------	--------------------------------------

6.32.3 Member Data Documentation

6.32.3.1 std::string ErrorClass::errmsg

errmsg is the error that is causing the exception to be thrown

Definition at line 42 of file OSErrClass.h.

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

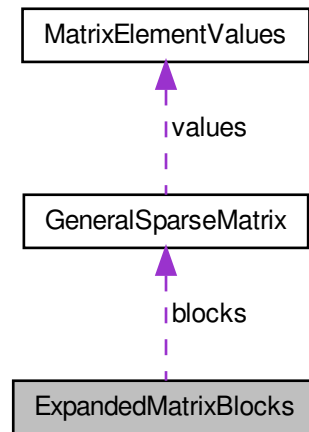
- /home/ted/COIN/trunk/OS/src/OSUtils/[OSErrClass.h](#)

6.33 ExpandedMatrixBlocks Class Reference

a sparse matrix data structure for matrices that can hold nonconstant values and have block structure In addition it is assumed that all nesting of blocks has been resolved.

```
#include <OSMatrix.h>
```


Collaboration diagram for ExpandedMatrixBlocks:



Public Member Functions

- [ExpandedMatrixBlocks](#) ()
Default constructor.
- [ExpandedMatrixBlocks](#) (bool isColumnMajor_, int startSize, int valueSize)
Alternate constructor.
- [~ExpandedMatrixBlocks](#) ()
Default destructor.
- bool [display](#) (int secondaryDim)
This method displays data structure in the matrix format.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- bool [isColumnMajor](#)
isColumnMajor holds whether the (nonzero) values holding the data are stored by column.
- int [blockNumber](#)
blockNumber gives the number of blocks (which is the size of the blockRows and blockColumns arrays).
- int * [rowOffsets](#)
rowOffsets gives the row offsets of the block decomposition It does not have to correspond to the row offsets in the matrix's <blocks> element (indeed there does not have to be such an element at all, or there may be several, possibly incompatible).
- int * [colOffsets](#)
colOffsets gives the column offsets of the block decomposition It does not have to correspond to the column offsets in the matrix's <blocks> element (indeed there does not have to be such an element at all, or there may be several, possibly incompatible).

- int * [blockRows](#)
blockRows holds an integer array of the row to which a block belongs.
- int * [blockColumns](#)
blockColumns holds an integer array of the column to which a block belongs.
- [GeneralSparseMatrix](#) ** [blocks](#)
blocks holds the blocks that make up the matrix.

6.33.1 Detailed Description

a sparse matrix data structure for matrices that can hold nonconstant values and have block structure In addition it is assumed that all nesting of blocks has been resolved.

Definition at line 1755 of file OSMatrix.h.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 ExpandedMatrixBlocks::ExpandedMatrixBlocks ()

Default constructor.

6.33.2.2 ExpandedMatrixBlocks::ExpandedMatrixBlocks (bool *isColumnMajor*_, int *startSize*, int *valueSize*)

Alternate constructor.

Parameters

<i>isColumnMajor</i>	holds whether the coefMatrix (AMatrix) holding linear program data is stored by column. If false, the matrix is stored by row.
<i>startSize</i>	holds the size of the start array.
<i>valueSize</i>	holds the size of the value and index arrays.

6.33.2.3 ExpandedMatrixBlocks::~~ExpandedMatrixBlocks ()

Default destructor.

6.33.3 Member Function Documentation

6.33.3.1 bool ExpandedMatrixBlocks::display (int *secondaryDim*)

This method displays data structure in the matrix format.

Returns

6.33.4 Member Data Documentation

6.33.4.1 bool ExpandedMatrixBlocks::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 1762 of file OSMatrix.h.

6.33.4.2 bool ExpandedMatrixBlocks::isColumnMajor

isColumnMajor holds whether the (nonzero) values holding the data are stored by column.

If false, the matrix is stored by row.

Definition at line 1768 of file OSMatrix.h.

6.33.4.3 int ExpandedMatrixBlocks::blockNumber

blockNumber gives the number of blocks (which is the size of the blockRows and blockColumns arrays).

Definition at line 1774 of file OSMatrix.h.

6.33.4.4 int* ExpandedMatrixBlocks::rowOffsets

rowOffsets gives the row offsets of the block decomposition. It does not have to correspond to the row offsets in the matrix's <blocks> element (indeed there does not have to be such an element at all, or there may be several, possibly incompatible).

Definition at line 1782 of file OSMatrix.h.

6.33.4.5 int* ExpandedMatrixBlocks::colOffsets

colOffsets gives the column offsets of the block decomposition. It does not have to correspond to the column offsets in the matrix's <blocks> element (indeed there does not have to be such an element at all, or there may be several, possibly incompatible).

Definition at line 1790 of file OSMatrix.h.

6.33.4.6 int* ExpandedMatrixBlocks::blockRows

blockRows holds an integer array of the row to which a block belongs.

It must be of dimension blockNumber. It is assumed that all blocks in a row have the same number of rows (while the number of columns is allowed to vary).

Definition at line 1798 of file OSMatrix.h.

6.33.4.7 int* ExpandedMatrixBlocks::blockColumns

blockColumns holds an integer array of the column to which a block belongs.

It must be of dimension blockNumber. It is assumed that all blocks in a column have the same number of columns (while the number of rows is allowed to vary).

Definition at line 1806 of file OSMatrix.h.

6.33.4.8 GeneralSparseMatrix ExpandedMatrixBlocks::blocks**

blocks holds the blocks that make up the matrix.

All blocks have the same type of values, which corresponds to the most general form found, and the same row/column major form.

Definition at line 1813 of file OSMatrix.h.

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

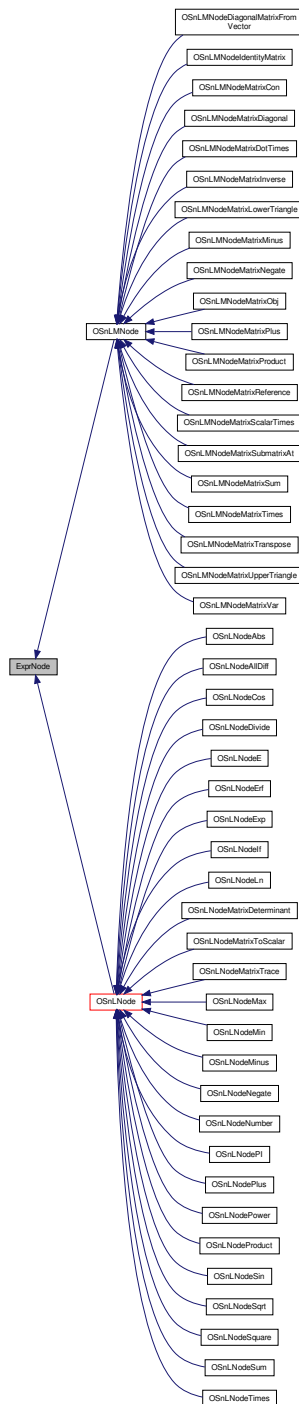
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h>

6.34 ExprNode Class Reference

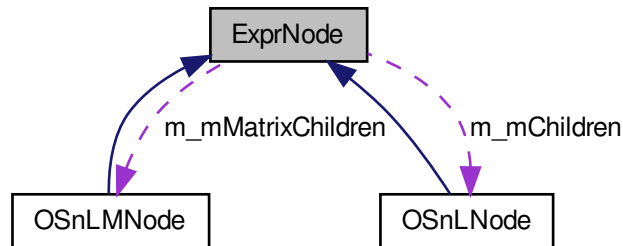
A generic class from which we derive both [OSnLNode](#) and [OSnLMNode](#).

```
#include <OSnLNode.h>
```

Inheritance diagram for ExprNode:



Collaboration diagram for ExprNode:



Public Member Functions

- [ExprNode](#) ()
default constructor.
- virtual [~ExprNode](#) ()
default destructor.
- virtual std::string [getTokenNumber](#) ()
- virtual std::string [getTokenName](#) ()=0
- virtual std::string [getNonlinearExpressionInXML](#) ()
The following method writes an [OSnLNode](#) or [OSnLMNode](#) in OSiL format.
- virtual std::vector< [ExprNode](#) * > [getPrefixFromExpressionTree](#) ()
Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the (scalar-valued or matrix-valued) expression tree in prefix format.
- virtual std::vector< [ExprNode](#) * > [preOrderOSnLNodeTraversal](#) (std::vector< [ExprNode](#) * > *prefixVector)
Called by [getPrefixFromExpressionTree](#)().
- virtual std::vector< [ExprNode](#) * > [getPostfixFromExpressionTree](#) ()
Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.
- virtual std::vector< [ExprNode](#) * > [postOrderOSnLNodeTraversal](#) (std::vector< [ExprNode](#) * > *postfixVector)
Called by [getPostfixFromExpressionTree](#)().
- virtual [ExprNode](#) * [cloneExprNode](#) ()=0
Create or clone a node of this type.
- virtual bool [IsEqual](#) ([ExprNode](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [inodeInt](#)
inodeInt is the unique integer assigned to the [OSnLNode](#) or [OSnLMNode](#) in [OSParameters.h](#).
- int [inodeType](#)
inodeType essentially tracks whether the number of children are known or not.
- unsigned int [numberOfChildren](#)

numberOfChildren is the number of [OSnLNode](#) child elements. If this number is not fixed, e.g., for a sum node, it is temporarily set to 0

- unsigned int [numberOfMatrixChildren](#)

numberOfMatrixChildren is the number of [OSnLMNode](#) child elements. If this number is not fixed, e.g., for a matrixProduct node, it is temporarily set to 0

- [OSnLNode](#) ** [m_mChildren](#)

m_mChildren holds all the operands, that is, nodes that the current node operates on.

- [OSnLMNode](#) ** [m_mMatrixChildren](#)

m_mMatrixChildren holds all the matrix-valued operands, if any.

6.34.1 Detailed Description

A generic class from which we derive both [OSnLNode](#) and [OSnLMNode](#).

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

2.9, 10/Sep/2014

Definition at line 56 of file [OSnLNode.h](#).

6.34.2 Constructor & Destructor Documentation

6.34.2.1 ExprNode::ExprNode ()

default constructor.

6.34.2.2 virtual ExprNode::~~ExprNode () [virtual]

default destructor.

6.34.3 Member Function Documentation

6.34.3.1 virtual std::string ExprNode::getTokenNumber () [virtual]

Returns

the value of `inodelnt`

Reimplemented in [OSnLMNodeMatrixCon](#), [OSnLMNodeMatrixObj](#), [OSnLMNodeMatrixVar](#), [OSnLMNodeMatrixReference](#), [OSnLNodeVariable](#), [OSnLNodePI](#), [OSnLNodeE](#), and [OSnLNodeNumber](#).

6.34.3.2 virtual std::string ExprNode::getTokenName () [pure virtual]

Returns

the value of the operator name

Implemented in [OSnLMNodeMatrixProduct](#), [OSnLMNodeMatrixCon](#), [OSnLMNodeMatrixObj](#), [OSnLMNodeMatrixVar](#), [OSnLMNodeMatrixReference](#), [OSnLMNodeMatrixSubmatrixAt](#), [OSnLMNodeDiagonalMatrixFromVector](#), [OSnLMNodeMatrixDiagonal](#), [OSnLMNodeMatrixUpperTriangle](#), [OSnLMNodeMatrixLowerTriangle](#), [OSnLMNodeIdentityMatrix](#), [OSnLMNodeMatrixDotTimes](#), [OSnLMNodeMatrixScalarTimes](#), [OSnLMNodeMatrixTranspose](#), [OSnLMNodeMatrixInverse](#), [OSnLMNodeMatrixTimes](#), [OSnLMNodeMatrixNegate](#), [OSnLMNodeMatrixMinus](#), [OSnLMNodeMatrixSum](#), [OSnLMNodeMatrixPlus](#), [OSnLMNodeMatrixToScalar](#), [OSnLMNodeMatrixTrace](#), [OSnLMNodeMatrixDeterminant](#), [OSnLMNodeAllDiff](#), [OSnLMNodeVariable](#), [OSnLMNodePI](#), [OSnLMNodeE](#), [OSnLMNodeNumber](#), [OSnLMNodeIf](#), [OSnLMNodeErf](#), [OSnLMNodeAbs](#), [OSnLMNodeExp](#), [OSnLMNodeSin](#), [OSnLMNodeCos](#), [OSnLMNodeSquare](#), [OSnLMNodeSqrt](#), [OSnLMNodeLn](#), [OSnLMNodeProduct](#), [OSnLMNodePower](#), [OSnLMNodeDivide](#), [OSnLMNodeTimes](#), [OSnLMNodeNegate](#), [OSnLMNodeMinus](#), [OSnLMNodeMin](#), [OSnLMNodeMax](#), [OSnLMNodeSum](#), and [OSnLMNodePlus](#).

6.34.3.3 `virtual std::string ExprNode::getNonlinearExpressionInXML () [virtual]`

The following method writes an [OSnLMNode](#) or [OSnLMNode](#) in OSiL format.

It is used by [OSiLWriter](#) to assist in writing an OSiL file from a corresponding [OSInstance](#).

Returns

the [ExprNode](#) and its children as an OSiL string.

Reimplemented in [OSnLMNodeMatrixCon](#), [OSnLMNodeMatrixObj](#), [OSnLMNodeMatrixVar](#), [OSnLMNodeMatrixReference](#), [OSnLMNodeMatrixUpperTriangle](#), [OSnLMNodeMatrixLowerTriangle](#), [OSnLMNodeVariable](#), [OSnLMNodePI](#), [OSnLMNodeE](#), and [OSnLMNodeNumber](#).

6.34.3.4 `virtual std::vector<ExprNode*> ExprNode::getPrefixFromExpressionTree () [virtual]`

Get a vector of pointers to [OSnLMNodes](#) and [OSnLMNodes](#) that correspond to the (scalar-valued or matrix-valued) expression tree in prefix format.

Returns

the expression tree as a vector of [ExprNodes](#) in prefix.

Reimplemented in [OSnLMNode](#), and [OSnLMNode](#).

6.34.3.5 `virtual std::vector<ExprNode*> ExprNode::preOrderOSnLMNodeTraversal (std::vector< ExprNode * > * prefixVector) [virtual]`

Called by [getPrefixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to [ExprNode](#) in prefix

Parameters

<i>a</i>	pointer <i>prefixVector</i> to a vector of pointers of ExprNodes
----------	--

Returns

a vector of pointers to [ExprNode](#) in prefix.

Reimplemented in [OSnLMNode](#), and [OSnLMNode](#).

6.34.3.6 `virtual std::vector<ExprNode*> ExprNode::getPostfixFromExpressionTree () [virtual]`

Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.

Returns

the expression tree as a vector of ExprNodes in postfix.

Reimplemented in [OSnLMNode](#), and [OSnLNode](#).

6.34.3.7 `virtual std::vector<ExprNode*> ExprNode::postOrderOSnLNodeTraversal (std::vector< ExprNode * > * postfixVector) [virtual]`

Called by [getPostfixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to ExprNodes in postfix.

Parameters

a	pointer postfixVector to a vector of pointers of ExprNodes
---	--

Returns

a vector of pointers to ExprNodes in postfix.

Reimplemented in [OSnLMNode](#), and [OSnLNode](#).

6.34.3.8 `virtual ExprNode* ExprNode::cloneExprNode () [pure virtual]`

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implemented in [OSnLMNodeMatrixProduct](#), [OSnLMNodeMatrixCon](#), [OSnLMNodeMatrixObj](#), [OSnLMNodeMatrixVar](#), [OSnLMNodeMatrixReference](#), [OSnLMNodeMatrixSubmatrixAt](#), [OSnLMNodeDiagonalMatrixFromVector](#), [OSnLMNodeMatrixDiagonal](#), [OSnLMNodeMatrixUpperTriangle](#), [OSnLMNodeMatrixLowerTriangle](#), [OSnLMNodeIdentityMatrix](#), [OSnLMNodeMatrixDotTimes](#), [OSnLMNodeMatrixScalarTimes](#), [OSnLMNodeMatrixTranspose](#), [OSnLMNodeMatrixInverse](#), [OSnLMNodeMatrixTimes](#), [OSnLMNodeMatrixNegate](#), [OSnLMNodeMatrixMinus](#), [OSnLMNodeMatrixSum](#), [OSnLMNodeMatrixPlus](#), [OSnLNodeMatrixToScalar](#), [OSnLNodeMatrixTrace](#), [OSnLNodeMatrixDeterminant](#), [OSnLNodeAllDiff](#), [OSnLNodeVariable](#), [OSnLNodePI](#), [OSnLNodeE](#), [OSnLNodeNumber](#), [OSnLNodeIf](#), [OSnLNodeErf](#), [OSnLNodeAbs](#), [OSnLNodeExp](#), [OSnLNodeSin](#), [OSnLNodeCos](#), [OSnLNodeSquare](#), [OSnLNodeSqrt](#), [OSnLNodeLn](#), [OSnLNodeProduct](#), [OSnLNodePower](#), [OSnLNodeDivide](#), [OSnLNodeTimes](#), [OSnLNodeNegate](#), [OSnLNodeMinus](#), [OSnLNodeMin](#), [OSnLNodeMax](#), [OSnLNodeSum](#), and [OSnLNodePlus](#).

6.34.3.9 `virtual bool ExprNode::isEqual (ExprNode * that) [virtual]`

A function to check for the equality of two objects.

6.34.4 Member Data Documentation

6.34.4.1 `int ExprNode::inodeInt`

inodeInt is the unique integer assigned to the [OSnLNode](#) or [OSnLMNode](#) in [OSParameters.h](#).

Definition at line 62 of file [OSnLNode.h](#).

6.34.4.2 int ExprNode::inodeType

inodeType essentially tracks whether the number of children are known or not.

For most nodes this is known, in which case inodeType is set to inumberOfChildren. For some nodes the number of children is not known a priori, e.g., a sum node, then inodeType is set to -1.

Definition at line 69 of file OSnLNode.h.

6.34.4.3 unsigned int ExprNode::inumberOfChildren

inumberOfChildren is the number of [OSnLNode](#) child elements If this number is not fixed, e.g., for a sum node, it is temporarily set to 0

Definition at line 74 of file OSnLNode.h.

6.34.4.4 unsigned int ExprNode::inumberOfMatrixChildren

inumberOfMatrixChildren is the number of [OSnLMNode](#) child elements If this number is not fixed, e.g., for a matrix-Product node, it is temporarily set to 0

Definition at line 79 of file OSnLNode.h.

6.34.4.5 OSnLNode** ExprNode::m_mChildren

m_mChildren holds all the operands, that is, nodes that the current node operates on.

Definition at line 84 of file OSnLNode.h.

6.34.4.6 OSnLMNode** ExprNode::m_mMatrixChildren

m_mMatrixChildren holds all the matrix-valued operands, if any.

Definition at line 89 of file OSnLNode.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.35 FileUtil Class Reference

class used to make it easy to read and write files.

```
#include <OSFileUtil.h>
```

Public Member Functions

- [FileUtil](#) ()
the class constructor
- [~FileUtil](#) ()
the class destructor
- std::string [getFileAsString](#) (const char *fname)
read a file and return contents as a string.
- char * [getFileAsChar](#) (const char *fname)
read a file and return contents as a char pointer.
- bool [writeFileFromString](#) (char *fname, std::string thestring)
write a file from an input string.

- bool [writeFileFromString](#) (std::string fname, std::string thestring)
write a file from an input string.
- bool [writeFileFromChar](#) (char *fname, char *ch)
write a file from an input char pointer.

6.35.1 Detailed Description

class used to make it easy to read and write files.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

The [FileUtil](#) class contains methods for reading and writing files from strings used by many classes in the Optimization Services (OS) framework.

Definition at line 37 of file OSFileUtil.h.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 FileUtil::FileUtil ()

the class constructor

6.35.2.2 FileUtil::~~FileUtil ()

the class destructor

6.35.3 Member Function Documentation

6.35.3.1 std::string FileUtil::getFileAsString (const char * fname)

read a file and return contents as a string.

Parameters

<i>fname</i>	holds the name of the file.
--------------	-----------------------------

Returns

the file contents as a string.

6.35.3.2 char* FileUtil::getFileAsChar (const char * *fname*)

read a file and return contents as a char pointer.

Parameters

<i>fname</i>	holds the name of the file.
--------------	-----------------------------

Returns

the file contents as a char pointer.

6.35.3.3 bool FileUtil::writeFileFromString (char * *fname*, std::string *thestring*)

write a file from an input string.

Parameters

<i>fname</i>	holds the name of the file to be written.
<i>thestring</i>	holds the string to be written to the file.

Returns

true if file successfully written.

6.35.3.4 bool FileUtil::writeFileFromString (std::string *fname*, std::string *thestring*)

write a file from an input string.

Parameters

<i>fname</i>	holds the name of the file to be written.
<i>thestring</i>	holds the string to be written to the file.

Returns

true if file successfully written.

6.35.3.5 bool FileUtil::writeFileFromChar (char * *fname*, char * *ch*)

write a file from an input char pointer.

Parameters

<i>fname</i>	holds the name of the file to be written.
<i>ch</i>	holds a pointer to a char array to be written to the file.

Returns

true if file successfully written.

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

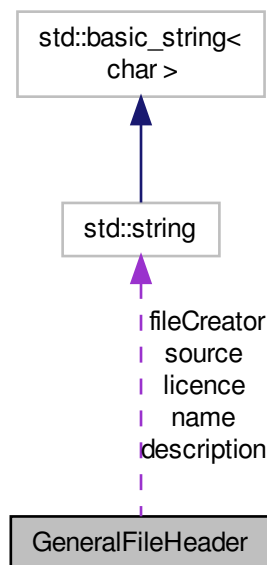
- /home/ted/COIN/trunk/OS/src/OSUtils/[OSFileUtil.h](#)

6.36 GeneralFileHeader Class Reference

a data structure that holds general information about files that conform to one of the OSxL schemas

```
#include <OSGeneral.h>
```

Collaboration diagram for GeneralFileHeader:

**Public Member Functions**

- [GeneralFileHeader](#) ()
Constructor.
- [~GeneralFileHeader](#) ()
Default destructor.
- bool [isEqual](#) ([GeneralFileHeader](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([GeneralFileHeader](#) *that)

A function to make a deep copy of an instance of this class.

- `std::string` [getHeaderItem](#) (`std::string item`)

A function to retrieve a data item contained in this class.

- `bool` [setHeader](#) (`std::string name`, `std::string source`, `std::string description`, `std::string fileCreator`, `std::string licence`)

A function to populate an instance of this class.

Public Attributes

- `std::string` [name](#)
used to give a name to the file or the problem contained within it
- `std::string` [source](#)
used when the file or problem appeared in the literature (could be in BiBTeX format or similar)
- `std::string` [description](#)
further information about the file or the problem contained within it
- `std::string` [fileCreator](#)
name(s) of author(s) who created this file
- `std::string` [licence](#)
licensing information if applicable

6.36.1 Detailed Description

a data structure that holds general information about files that conform to one of the OSxL schemas

Definition at line 32 of file OSGeneral.h.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 GeneralFileHeader::GeneralFileHeader ()

Constructor.

6.36.2.2 GeneralFileHeader::~~GeneralFileHeader ()

Default destructor.

6.36.3 Member Function Documentation

6.36.3.1 bool GeneralFileHeader::isEqual (GeneralFileHeader * that)

A function to check for the equality of two objects.

6.36.3.2 bool GeneralFileHeader::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.36.3.3 `bool GeneralFileHeader::deepCopyFrom (GeneralFileHeader * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.36.3.4 `std::string GeneralFileHeader::getHeaderItem (std::string item)`

A function to retrieve a data item contained in this class.

Parameters

<i>item</i> ,:	the type of information sought (name, source, description, fileCreator, licence)
----------------	--

6.36.3.5 `bool GeneralFileHeader::setHeader (std::string name, std::string source, std::string description, std::string fileCreator, std::string licence)`

A function to populate an instance of this class.

Parameters

<i>name</i> ,:	the name of this file or instance
<i>source</i> ,:	the source (e.g., in BiBTeX format)
<i>description</i> ,:	further description about this file and/or its contents
<i>fileCreator</i> ,:	the creator of this file
<i>licence</i> ,:	licence information if applicable

6.36.4 Member Data Documentation

6.36.4.1 `std::string GeneralFileHeader::name`

used to give a name to the file or the problem contained within it

Definition at line 39 of file OSGeneral.h.

6.36.4.2 `std::string GeneralFileHeader::source`

used when the file or problem appeared in the literature (could be in BiBTeX format or similar)

Definition at line 45 of file OSGeneral.h.

6.36.4.3 `std::string GeneralFileHeader::description`

further information about the file or the problem contained within it

Definition at line 50 of file OSGeneral.h.

6.36.4.4 `std::string GeneralFileHeader::fileCreator`

name(s) of author(s) who created this file

Definition at line 55 of file OSGeneral.h.

6.36.4.5 std::string GeneralFileHeader::licence

licensing information if applicable

Definition at line 60 of file OSGeneral.h.

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

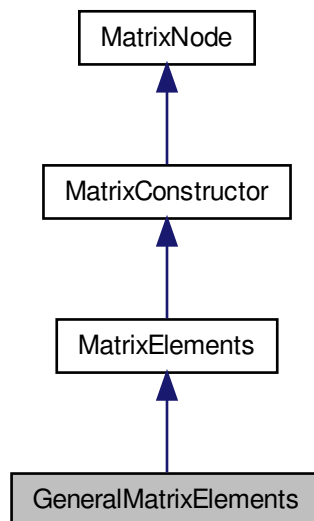
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSGeneral.h](#)

6.37 GeneralMatrixElements Class Reference

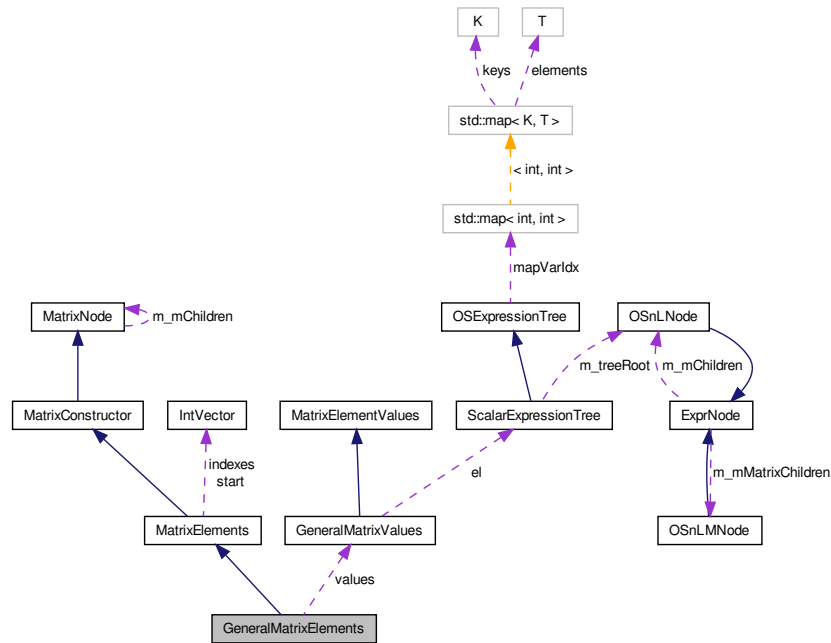
a data structure to represent the nonzero values in a generalMatrix element

```
#include <OSMatrix.h>
```

Inheritance diagram for GeneralMatrixElements:



Collaboration diagram for GeneralMatrixElements:



Public Member Functions

- [GeneralMatrixElements](#) ()
- [~GeneralMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual [std::string](#) [getNodeName](#) ()
- virtual [std::string](#) [getMatrixNodeInXML](#) ()
- virtual [bool](#) [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [GeneralMatrixElements](#) * [cloneMatrixNode](#) ()
- [bool](#) [isEqual](#) ([GeneralMatrixElements](#) *that)
A function to check for the equality of two objects.
- [bool](#) [setRandom](#) (double density, [bool](#) conformant, int iMin, int iMax)
A function to make a random instance of this class.
- [bool](#) [deepCopyFrom](#) ([GeneralMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [GeneralMatrixValues](#) * [values](#)
The values are general nonlinear expressions.

6.37.1 Detailed Description

a data structure to represent the nonzero values in a generalMatrix element

Definition at line 1006 of file OSMatrix.h.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 GeneralMatrixElements::GeneralMatrixElements ()

6.37.2.2 GeneralMatrixElements::~~GeneralMatrixElements ()

6.37.3 Member Function Documentation

6.37.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE GeneralMatrixElements::getNodeType () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.37.3.2 virtual ENUM_MATRIX_TYPE GeneralMatrixElements::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.37.3.3 virtual std::string GeneralMatrixElements::getNodeName () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.37.3.4 virtual std::string GeneralMatrixElements::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.37.3.5 virtual bool GeneralMatrixElements::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.37.3.6 virtual GeneralMatrixElements* GeneralMatrixElements::cloneMatrixNode () [virtual]

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.37.3.7 bool GeneralMatrixElements::isEqual (GeneralMatrixElements * that)

A function to check for the equality of two objects.

6.37.3.8 bool GeneralMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.37.3.9 bool GeneralMatrixElements::deepCopyFrom (GeneralMatrixElements * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.37.4 Member Data Documentation

6.37.4.1 GeneralMatrixValues* GeneralMatrixElements::values

The values are general nonlinear expressions.

Definition at line 1012 of file OSMatrix.h.

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

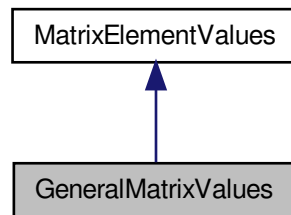
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.38 GeneralMatrixValues Class Reference

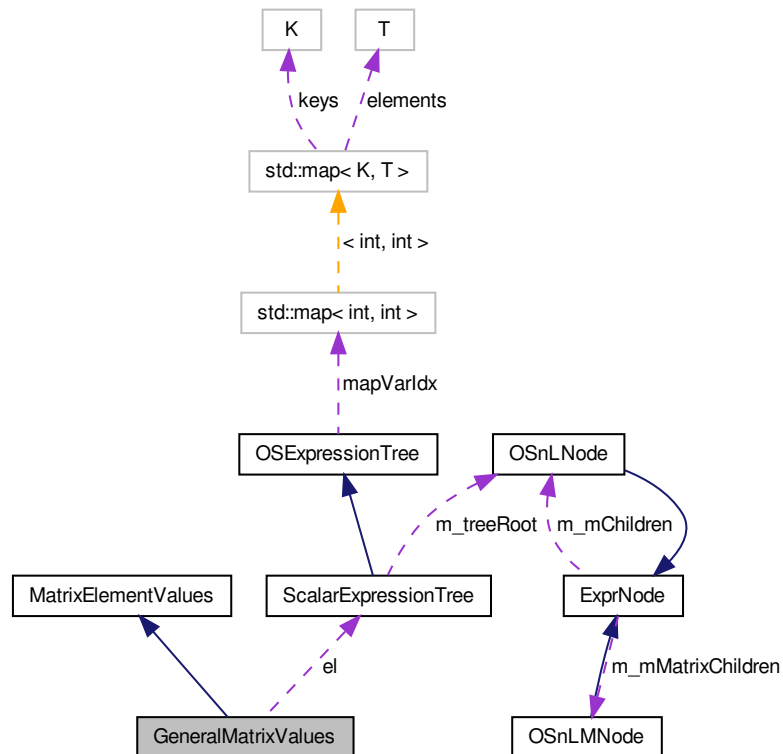
a data structure to represent the nonzeros in a generalMatrix element

```
#include <OSMatrix.h>
```

Inheritance diagram for GeneralMatrixValues:



Collaboration diagram for GeneralMatrixValues:



Public Member Functions

- [GeneralMatrixValues](#) ()
- [~GeneralMatrixValues](#) ()
- bool [isEqual](#) ([GeneralMatrixValues](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([GeneralMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ScalarExpressionTree](#) ** el

6.38.1 Detailed Description

a data structure to represent the nonzeros in a generalMatrix element
Definition at line 640 of file OSMatrix.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 [GeneralMatrixValues::GeneralMatrixValues](#) ()

6.38.2.2 [GeneralMatrixValues::~~GeneralMatrixValues](#) ()

6.38.3 Member Function Documentation

6.38.3.1 bool [GeneralMatrixValues::isEqual](#) ([GeneralMatrixValues](#) * that)

A function to check for the equality of two objects.

6.38.3.2 bool [GeneralMatrixValues::setRandom](#) (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.38.3.3 bool [GeneralMatrixValues::deepCopyFrom](#) ([GeneralMatrixValues](#) * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.38.4 Member Data Documentation**6.38.4.1 ScalarExpressionTree** GeneralMatrixValues::el**

Definition at line 643 of file OSMatrix.h.

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

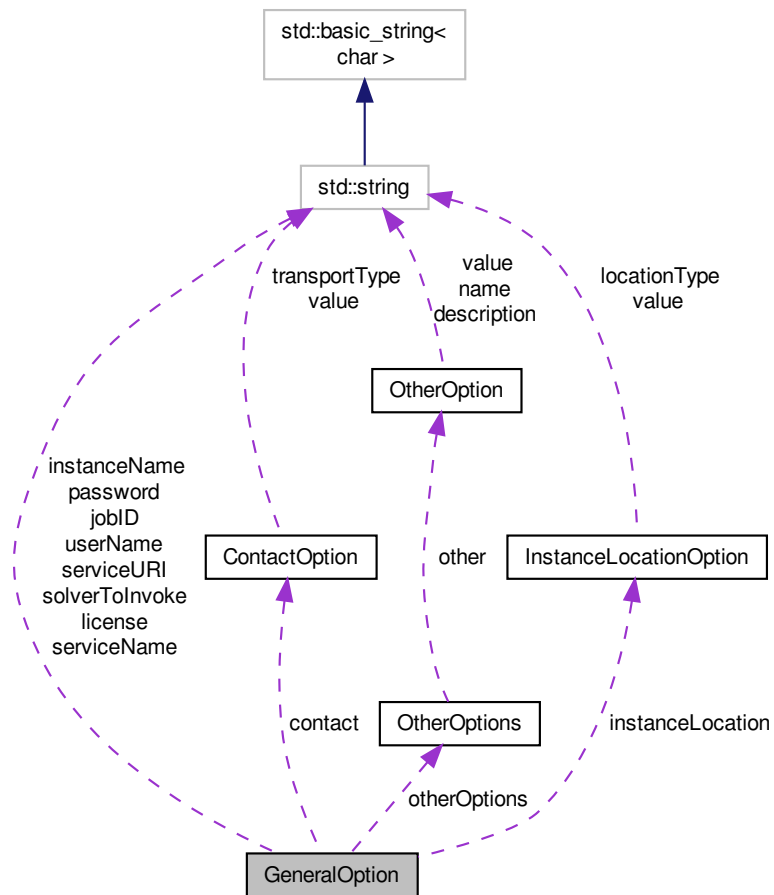
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.39 GeneralOption Class Reference

The [GeneralOption](#) Class.

```
#include <OSOption.h>
```

Collaboration diagram for GeneralOption:



Public Member Functions

- [GeneralOption](#) ()
Default constructor.
- [~GeneralOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([GeneralOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([GeneralOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string [serviceURI](#)
the service URI
- std::string [serviceName](#)
the name of the service
- std::string [instanceName](#)
the name of the instance
- [InstanceLocationOption](#) * [instanceLocation](#)
the location of the instance
- std::string [jobID](#)
the job ID
- std::string [solverToInvoke](#)
the solver to invoke
- std::string [license](#)
the license information
- std::string [userName](#)
the username
- std::string [password](#)
the password
- [ContactOption](#) * [contact](#)
the contact method
- [OtherOptions](#) * [otherOptions](#)
the list of other general options

6.39.1 Detailed Description

The [GeneralOption](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 284 of file OSOption.h.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 GeneralOption::GeneralOption ()

Default constructor.

6.39.2.2 GeneralOption::~~GeneralOption ()

Class destructor.

6.39.3 Member Function Documentation

6.39.3.1 bool GeneralOption::isEqual (GeneralOption * *that*)

A function to check for the equality of two objects.

6.39.3.2 bool GeneralOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.39.3.3 bool GeneralOption::deepCopyFrom (GeneralOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.39.4 Member Data Documentation

6.39.4.1 std::string GeneralOption::serviceURI

the service URI

Definition at line 290 of file OSOption.h.

6.39.4.2 std::string GeneralOption::serviceName

the name of the service

Definition at line 293 of file OSOption.h.

6.39.4.3 std::string GeneralOption::instanceName

the name of the instance

Definition at line 296 of file OSOption.h.

6.39.4.4 InstanceLocationOption* GeneralOption::instanceLocation

the location of the instance

Definition at line 299 of file OSOption.h.

6.39.4.5 std::string GeneralOption::jobID

the job ID

Definition at line 302 of file OSOption.h.

6.39.4.6 std::string GeneralOption::solverToInvoke

the solver to invoke

Definition at line 305 of file OSOption.h.

6.39.4.7 std::string GeneralOption::license

the license information

Definition at line 308 of file OSOption.h.

6.39.4.8 std::string GeneralOption::userName

the username

Definition at line 311 of file OSOption.h.

6.39.4.9 std::string GeneralOption::password

the password

Definition at line 314 of file OSOption.h.

6.39.4.10 ContactOption* GeneralOption::contact

the contact method

Definition at line 317 of file OSOption.h.

6.39.4.11 OtherOptions* GeneralOption::otherOptions

the list of other general options

Definition at line 320 of file OSOption.h.

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

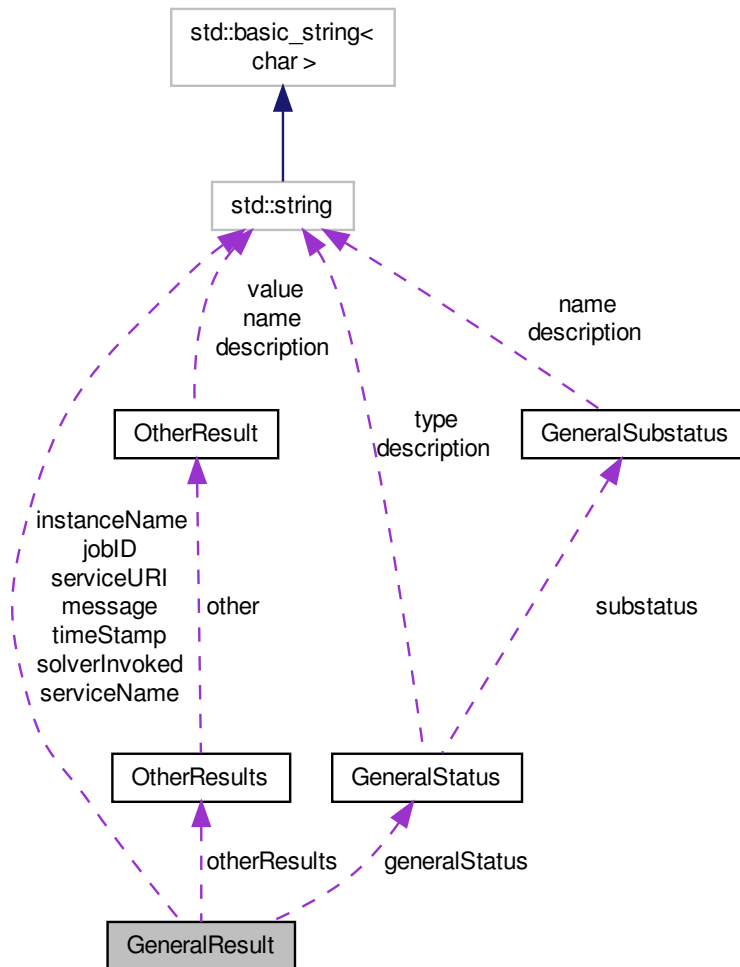
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.40 GeneralResult Class Reference

The [GeneralResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for GeneralResult:



Public Member Functions

- [GeneralResult](#) ()
Default constructor.
- [~GeneralResult](#) ()
Class destructor.
- `bool` [isEqual](#) ([GeneralResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- [GeneralStatus](#) * [generalStatus](#)
a pointer to the [GeneralStatus](#) class
- std::string [message](#)
any general message associated with the optimization
- std::string [serviceURI](#)
the serviceURI is the URI of the solver service that did the optimization
- std::string [serviceName](#)
the serviceName is the name of the solver service that did the optimization
- std::string [instanceName](#)
the name of the instance that was solved
- std::string [jobID](#)
the jobID is the ID associated with the solution of this instance
- std::string [solverInvoked](#)
the name of the solver used
- std::string [timeStamp](#)
a time stamp associated with the process
- [OtherResults](#) * [otherResults](#)
a pointer to the [OtherResults](#) class

6.40.1 Detailed Description

The [GeneralResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the general information that is defined in the OSrL schema.

Definition at line 265 of file OSResult.h.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 GeneralResult::GeneralResult ()

Default constructor.

6.40.2.2 GeneralResult::~GeneralResult ()

Class destructor.

6.40.3 Member Function Documentation

6.40.3.1 bool GeneralResult::isEqual (GeneralResult * *that*)

A function to check for the equality of two objects.

6.40.3.2 bool GeneralResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.40.4 Member Data Documentation

6.40.4.1 GeneralStatus* GeneralResult::generalStatus

a pointer to the [GeneralStatus](#) class

Definition at line 271 of file OSResult.h.

6.40.4.2 std::string GeneralResult::message

any general message associated with the optimization

Definition at line 275 of file OSResult.h.

6.40.4.3 std::string GeneralResult::serviceURI

the serviceURI is the URI of the solver service that did the optimization

Definition at line 280 of file OSResult.h.

6.40.4.4 std::string GeneralResult::serviceName

the serviceName is the name of the solver service that did the optimization

Definition at line 285 of file OSResult.h.

6.40.4.5 std::string GeneralResult::instanceName

the name of the instance that was solved

Definition at line 289 of file OSResult.h.

6.40.4.6 std::string GeneralResult::jobID

the jobID is the ID associated with the solution of this instance

Definition at line 294 of file OSResult.h.

6.40.4.7 `std::string GeneralResult::solverInvoked`

the name of the solver used

Definition at line 298 of file OSResult.h.

6.40.4.8 `std::string GeneralResult::timeStamp`

a time stamp associated with the process

Definition at line 302 of file OSResult.h.

6.40.4.9 `OtherResults*` `GeneralResult::otherResults`

a pointer to the [OtherResults](#) class

Definition at line 306 of file OSResult.h.

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

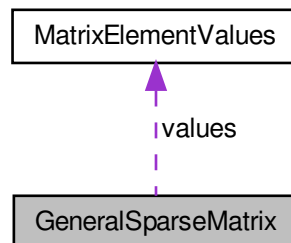
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h`

6.41 GeneralSparseMatrix Class Reference

a sparse matrix data structure for matrices that can hold nonconstant values

```
#include <OSMatrix.h>
```

Collaboration diagram for GeneralSparseMatrix:



Public Member Functions

- [GeneralSparseMatrix](#) ()
Default constructor.
- [GeneralSparseMatrix](#) (bool `isColumnMajor`, int `startSize`, int `valueSize`, [ENUM_MATRIX_TYPE](#) type)
Alternate constructor.
- [~GeneralSparseMatrix](#) ()
Default destructor.
- bool [display](#) (int `secondaryDim`)
This method displays the data contained in the matrix.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- bool [isColumnMajor](#)
isColumnMajor holds whether the matrix is stored by column.
- int [startSize](#)
startSize is the dimension of the starts array
- int [valueSize](#)
valueSize is the dimension of the indexes and values arrays
- int * [starts](#)
starts holds an integer array of start elements in the matrix, which points to the start of a column (row) of nonzero elements.
- int * [indexes](#)
indexes holds an integer array of rowldx (or colldx) elements in coefMatrix (AMatrix).
- [ENUM_MATRIX_TYPE](#) [vType](#)
vType holds the type of values found in the values array.
- [MatrixElementValues](#) ** [values](#)
values holds a general array of value elements in the matrix, which could be constants, linear expressions, general nonlinear expressions, variable, constraint or objective references, etc.

6.41.1 Detailed Description

a sparse matrix data structure for matrices that can hold nonconstant values

Definition at line 1664 of file OSMatrix.h.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 GeneralSparseMatrix::GeneralSparseMatrix ()

Default constructor.

6.41.2.2 GeneralSparseMatrix::GeneralSparseMatrix (bool *isColumnMajor*, int *startSize*, int *valueSize*, [ENUM_MATRIX_TYPE](#) *type*)

Alternate constructor.

Parameters

<i>isColumnMajor</i>	holds whether the matrix is stored by column. If false, the matrix is stored by row.
<i>startSize</i>	holds the size of the start array.
<i>valueSize</i>	holds the size of the value and index arrays.
<i>type</i>	describes the type of values held in the matrix (see OSParameters.h).

6.41.2.3 GeneralSparseMatrix::~GeneralSparseMatrix ()

Default destructor.

6.41.3 Member Function Documentation

6.41.3.1 bool GeneralSparseMatrix::display (int *secondaryDim*)

This method displays the data contained in the matrix.

Returns

6.41.4 Member Data Documentation

6.41.4.1 bool GeneralSparseMatrix::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 1672 of file OSMatrix.h.

6.41.4.2 bool GeneralSparseMatrix::isColumnMajor

isColumnMajor holds whether the matrix is stored by column.

If false, the matrix is stored by row.

Definition at line 1678 of file OSMatrix.h.

6.41.4.3 int GeneralSparseMatrix::startSize

startSize is the dimension of the starts array

Definition at line 1683 of file OSMatrix.h.

6.41.4.4 int GeneralSparseMatrix::valueSize

valueSize is the dimension of the indexes and values arrays

Definition at line 1688 of file OSMatrix.h.

6.41.4.5 int* GeneralSparseMatrix::starts

starts holds an integer array of start elements in the matrix, which points to the start of a column (row) of nonzero elements.

Definition at line 1694 of file OSMatrix.h.

6.41.4.6 int* GeneralSparseMatrix::indexes

indexes holds an integer array of rowIdx (or colIdx) elements in coefMatrix (AMatrix).

If the matrix is stored by column (row), rowIdx (colIdx) is the array of row (column) indices.

Definition at line 1700 of file OSMatrix.h.

6.41.4.7 ENUM_MATRIX_TYPE GeneralSparseMatrix::vType

vType holds the type of values found in the values array.

Remarks

See [OSParameters.h](#) for a list of possible types

Definition at line 1706 of file OSMatrix.h.

6.41.4.8 MatrixElementValues** GeneralSparseMatrix::values

values holds a general array of value elements in the matrix, which could be constants, linear expressions, general nonlinear expressions, variable, constraint or objective references, etc.

If mixed types are encountered (e.g., constant and nonlinear expression), they are converted to the most general form found.

Definition at line 1715 of file OSMatrix.h.

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

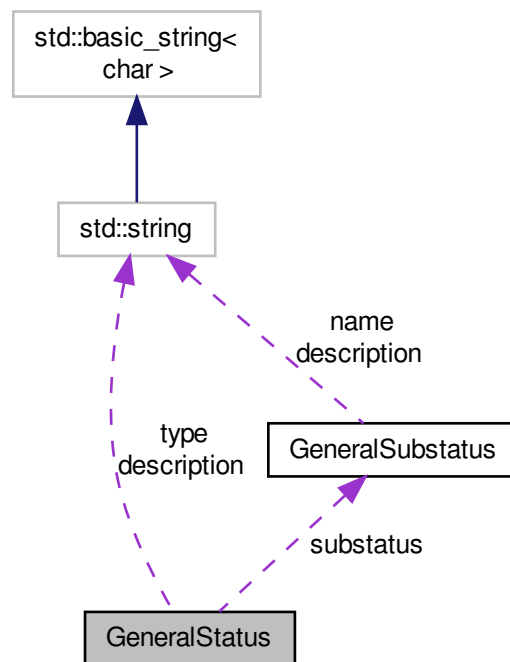
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h>

6.42 GeneralStatus Class Reference

The [GeneralStatus](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for GeneralStatus:



Public Member Functions

- [GeneralStatus](#) ()
Default constructor.

- [~GeneralStatus](#) ()
Class destructor.
- bool [isEqual](#) ([GeneralStatus](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [numberOfSubstatuses](#)
the number of substatuses
- std::string [type](#)
the type of status
- std::string [description](#)
the description of the status
- [GeneralSubstatus](#) ** [substatus](#)
the array of substatuses

6.42.1 Detailed Description

The [GeneralStatus](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A data structure class that corresponds to an xml element in the OSrL schema.

Definition at line 104 of file OSResult.h.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 [GeneralStatus::GeneralStatus](#) ()

Default constructor.

6.42.2.2 [GeneralStatus::~~GeneralStatus](#) ()

Class destructor.

6.42.3 Member Function Documentation

6.42.3.1 `bool GeneralStatus::isEqual (GeneralStatus * that)`

A function to check for the equality of two objects.

6.42.3.2 `bool GeneralStatus::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.42.4 Member Data Documentation

6.42.4.1 `int GeneralStatus::numberOfSubstatuses`

the number of substatuses

Definition at line 110 of file OSResult.h.

6.42.4.2 `std::string GeneralStatus::type`

the type of status

Definition at line 113 of file OSResult.h.

6.42.4.3 `std::string GeneralStatus::description`

the description of the status

Definition at line 116 of file OSResult.h.

6.42.4.4 `GeneralSubstatus** GeneralStatus::substatus`

the array of substatuses

Definition at line 119 of file OSResult.h.

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

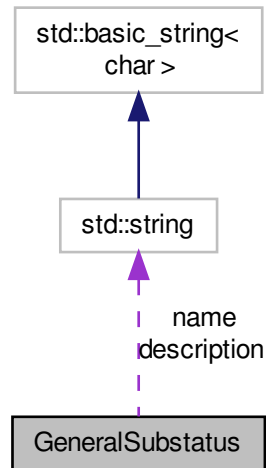
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.43 GeneralSubstatus Class Reference

The [GeneralSubstatus](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for GeneralSubstatus:



Public Member Functions

- [GeneralSubstatus](#) ()
Default constructor.
- [~GeneralSubstatus](#) ()
Class destructor.
- bool [isEqual](#) ([GeneralSubstatus](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string [name](#)
the name of the substatus
- std::string [description](#)
the description of the substatus

6.43.1 Detailed Description

The [GeneralSubstatus](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 11/03/2008

Since

OS 2.0

Remarks

A data structure class that corresponds to an xml element in the OSrL schema.

Definition at line 53 of file OSResult.h.

6.43.2 Constructor & Destructor Documentation**6.43.2.1 GeneralSubstatus::GeneralSubstatus ()**

Default constructor.

6.43.2.2 GeneralSubstatus::~~GeneralSubstatus ()

Class destructor.

6.43.3 Member Function Documentation**6.43.3.1 bool GeneralSubstatus::isEqual (GeneralSubstatus * *that*)**

A function to check for the equality of two objects.

6.43.3.2 bool GeneralSubstatus::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.43.4 Member Data Documentation**6.43.4.1 std::string GeneralSubstatus::name**

the name of the substatus

Definition at line 59 of file OSResult.h.

6.43.4.2 std::string GeneralSubstatus::description

the description of the substatus

Definition at line 62 of file OSResult.h.

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

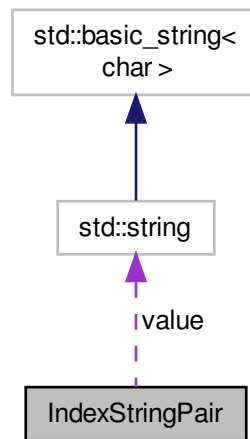
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.44 IndexStringPair Struct Reference

A commonly used structure holding an index-string pair This definition is based on the definition of [IndexValuePair](#) in [OSGeneral.h](#).

```
#include <OSResult.h>
```

Collaboration diagram for IndexStringPair:



Public Attributes

- `int` [idx](#)
idx holds the index of a string-valued entity (such as a variable, constraint, objective) that is part of a sparse vector
- `std::string` [value](#)
value is a string that holds the value of the entity

6.44.1 Detailed Description

A commonly used structure holding an index-string pair This definition is based on the definition of [IndexValuePair](#) in [OSGeneral.h](#).

Definition at line 28 of file `OSResult.h`.

6.44.2 Member Data Documentation

6.44.2.1 `int` `IndexStringPair::idx`

`idx` holds the index of a string-valued entity (such as a variable, constraint, objective) that is part of a sparse vector

Definition at line 33 of file OSResult.h.

6.44.2.2 std::string IndexStringPair::value

value is a string that holds the value of the entity

Definition at line 37 of file OSResult.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.45 IndexValuePair Struct Reference

A commonly used structure holding an index-value pair.

```
#include <OSGeneral.h>
```

Public Attributes

- int [idx](#)
idx holds the index of an entity (such as a variable, constraint, objective) that is part of a sparse vector
- double [value](#)
value is a double that holds the value of the entity

6.45.1 Detailed Description

A commonly used structure holding an index-value pair.

Definition at line 630 of file OSGeneral.h.

6.45.2 Member Data Documentation

6.45.2.1 int IndexValuePair::idx

idx holds the index of an entity (such as a variable, constraint, objective) that is part of a sparse vector

Definition at line 635 of file OSGeneral.h.

6.45.2.2 double IndexValuePair::value

value is a double that holds the value of the entity

Definition at line 638 of file OSGeneral.h.

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

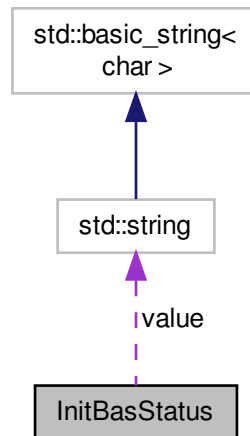
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.46 InitBasStatus Class Reference

the [InitBasStatus](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitBasStatus:



Public Member Functions

- `InitBasStatus ()`
Default constructor.
- `~InitBasStatus ()`
Class destructor.
- `bool IsEqual (InitBasStatus *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (InitBasStatus *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int idx`
variable index
- `std::string value`
initial value

6.46.1 Detailed Description

the `InitBasStatus` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been made redundant since OS version 2.3.

Definition at line 1481 of file OSOption.h.

6.46.2 Constructor & Destructor Documentation**6.46.2.1 InitBasStatus::InitBasStatus ()**

Default constructor.

6.46.2.2 InitBasStatus::~~InitBasStatus ()

Class destructor.

6.46.3 Member Function Documentation**6.46.3.1 bool InitBasStatus::isEqual (InitBasStatus * *that*)**

A function to check for the equality of two objects.

6.46.3.2 bool InitBasStatus::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.46.3.3 bool InitBasStatus::deepCopyFrom (InitBasStatus * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.46.4 Member Data Documentation**6.46.4.1 int InitBasStatus::idx**

variable index

Definition at line 1486 of file OSOption.h.

6.46.4.2 std::string InitBasStatus::value

initial value

Definition at line 1489 of file OSOption.h.

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

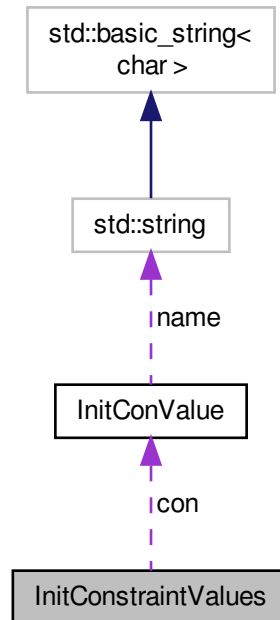
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.47 InitConstraintValues Class Reference

the [InitConstraintValues](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitConstraintValues:



Public Member Functions

- [InitConstraintValues](#) ()
Default constructor.
- [~InitConstraintValues](#) ()
Class destructor.
- bool [isEqual](#) ([InitConstraintValues](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InitConstraintValues](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setCon](#) (int [numberOfCon](#), [InitConValue](#) **con)
A function to set an array of <con> elements.
- bool [setCon](#) (int [numberOfCon](#), [InitConValue](#) **con, [ENUM_COMBINE_ARRAYS](#) disp)
Alternative signature for this function.
- bool [setCon](#) (int [numberOfCon](#), int *idx, double *value, std::string *name)
Another alternative signature for this function.
- bool [addCon](#) (int idx, double value)
A function to add a <con> element.
- bool [addCon](#) (int [numberOfCon](#), [InitConValue](#) **con)
Alternative signature for this function.

Public Attributes

- int [numberOfCon](#)
number of <con> children
- [InitConValue](#) ** [con](#)
initial value for each constraint

6.47.1 Detailed Description

the [InitConstraintValues](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2822 of file OSOption.h.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 InitConstraintValues::InitConstraintValues ()

Default constructor.

6.47.2.2 InitConstraintValues::~~InitConstraintValues ()

Class destructor.

6.47.3 Member Function Documentation

6.47.3.1 bool InitConstraintValues::isEqual (InitConstraintValues * *that*)

A function to check for the equality of two objects.

6.47.3.2 bool InitConstraintValues::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.47.3.3 bool InitConstraintValues::deepCopyFrom (InitConstraintValues * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.47.3.4 bool InitConstraintValues::setCon (int *numberOfCon*, InitConValue ** *con*)

A function to set an array of <con> elements.

Parameters

<i>numberOfCon,:</i>	number of <con> elements to be set
<i>con,:</i>	the array of <con> elements that are to be set

6.47.3.5 bool InitConstraintValues::setCon (int *numberOfCon*, InitConValue ** *con*, ENUM_COMBINE_ARRAYS *disp*)

Alternative signature for this function.

Parameters

<i>numberOfVar,:</i>	number of <i>elements to be set</i>
<i>var,:</i>	<i>the array of elements that are to be set</i>
<i>disp,:</i>	<i>method of disposition if previous data exist</i>

6.47.3.6 bool InitConstraintValues::setCon (int *numberOfCon*, int * *idx*, double * *value*, std::string * *name*)

Another alternative signature for this function.

Parameters

<i>numberOfCon,:</i>	number of <con> elements to be set
<i>idx,:</i>	the array of indices
<i>value,:</i>	the array of corresponding values
<i>name,:</i>	the array of constraint names

6.47.3.7 bool InitConstraintValues::addCon (int *idx*, double *value*)

A function to add a <con> element.

Parameters

<i>idx,:</i>	the index of the constraint to be given an initial value
<i>value,:</i>	the initial value to be added

6.47.3.8 bool InitConstraintValues::addCon (int *numberOfCon*, InitConValue ** *con*)

Alternative signature for this function.

A function to add an array of <con> elements simultaneously

Parameters

<i>numberOfCon,:</i>	number of <con> elements to be set
<i>obj,:</i>	the array of <con> elements that are to be set

6.47.4 Member Data Documentation

6.47.4.1 int InitConstraintValues::numberOfCon

number of <con> children

Definition at line 2827 of file OSOption.h.

6.47.4.2 InitConValue** InitConstraintValues::con

initial value for each constraint

Definition at line 2830 of file OSOption.h.

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

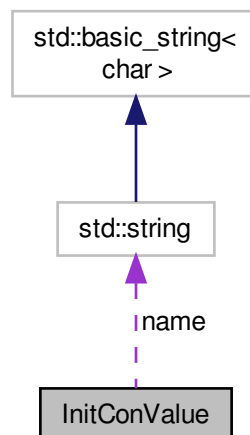
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.48 InitConValue Class Reference

the [InitConValue](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitConValue:



Public Member Functions

- [InitConValue](#) ()
Default constructor.
- [~InitConValue](#) ()
Class destructor.
- bool [IsEqual](#) ([InitConValue](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InitConValue](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
constraint index
- std::string [name](#)
optional variable name
- double [value](#)
initial value

6.48.1 Detailed Description

the [InitConValue](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2763 of file OSOption.h.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 InitConValue::InitConValue ()

Default constructor.

6.48.2.2 InitConValue::~~InitConValue ()

Class destructor.

6.48.3 Member Function Documentation

6.48.3.1 bool InitConValue::isEqual (InitConValue * *that*)

A function to check for the equality of two objects.

6.48.3.2 bool InitConValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.48.3.3 bool InitConValue::deepCopyFrom (InitConValue * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.48.4 Member Data Documentation

6.48.4.1 int InitConValue::idx

constraint index

Definition at line 2768 of file OSOption.h.

6.48.4.2 std::string InitConValue::name

optional variable name

Definition at line 2771 of file OSOption.h.

6.48.4.3 double InitConValue::value

initial value

Definition at line 2774 of file OSOption.h.

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

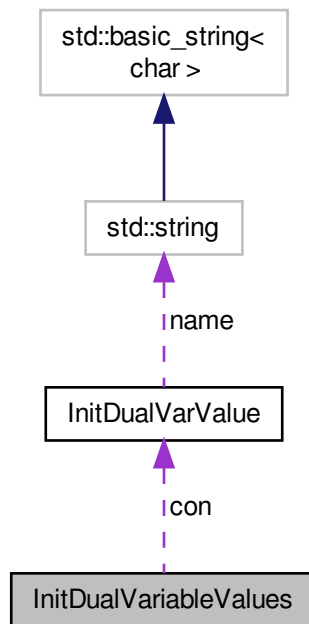
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.49 InitDualVariableValues Class Reference

the [InitDualVariableValues](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitDualVariableValues:



Public Member Functions

- [InitDualVariableValues](#) ()
Default constructor.
- [~InitDualVariableValues](#) ()
Class destructor.
- `bool` [IsEqual](#) ([InitDualVariableValues](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([InitDualVariableValues](#) *that)
A function to make a deep copy of an instance of this class.
- `bool` [setCon](#) (int numberOfCon, [InitDualVarValue](#) **con)
A function to set an array of <con> elements.
- `bool` [setCon](#) (int numberOfCon, [InitDualVarValue](#) **con, `ENUM_COMBINE_ARRAYS` disp)
Alternative signature for this function.

- bool [setCon](#) (int [numberOfCon](#), int *idx, double *lbValue, double *ubValue, std::string *name)
Another alternative signature for this function.
- bool [addCon](#) (int idx, double lbDualValue, double ubDualValue)
A function to add a <con> element.
- bool [addCon](#) (int [numberOfCon](#), [InitDualVarValue](#) **con)
Alternative signature for this function.

Public Attributes

- int [numberOfCon](#)
number of <con> children
- [InitDualVarValue](#) ** [con](#)
initial dual values for each constraint

6.49.1 Detailed Description

the [InitDualVariableValues](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2987 of file OSOption.h.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 InitDualVariableValues::InitDualVariableValues ()

Default constructor.

6.49.2.2 InitDualVariableValues::~~InitDualVariableValues ()

Class destructor.

6.49.3 Member Function Documentation

6.49.3.1 bool InitDualVariableValues::isEqual (InitDualVariableValues * that)

A function to check for the equality of two objects.

6.49.3.2 `bool InitDualVariableValues::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.49.3.3 `bool InitDualVariableValues::deepCopyFrom (InitDualVariableValues * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.49.3.4 `bool InitDualVariableValues::setCon (int numberOfCon, InitDualVarValue ** con)`

A function to set an array of <con> elements.

Parameters

<i>numberOfCon</i> ,:	number of <con> elements to be set
<i>con</i> ,:	the array of <con> elements that are to be set

6.49.3.5 `bool InitDualVariableValues::setCon (int numberOfCon, InitDualVarValue ** con, ENUM_COMBINE_ARRAYS disp)`

Alternative signature for this function.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements to be set</i>
<i>var</i> ,:	<i>the array of elements that are to be set</i>
<i>disp</i> ,:	<i>method of disposition if previous data exist</i>

6.49.3.6 `bool InitDualVariableValues::setCon (int numberOfCon, int * idx, double * lbValue, double * ubValue, std::string * name)`

Another alternative signature for this function.

Parameters

<i>numberOfCon</i> ,:	number of <con> elements to be set
<i>idx</i> ,:	the array of indices
<i>lbValue</i> ,:	the array of dual values for the lower bound
<i>ubValue</i> ,:	the array of dual values for the upper bound
<i>name</i> ,:	the array of constraint names

6.49.3.7 bool InitDualVariableValues::addCon (int *idx*, double *lbDualValue*, double *ubDualValue*)

A function to add a <con> element.

Parameters

<i>idx</i> ,:	the index of the constraint to be given initial dual variables
<i>lbDualValue</i> ,:	an initial value for the dual variable associated with the lower bound
<i>ubDualValue</i> ,:	an initial value for the dual variable associated with the upper bound

6.49.3.8 bool InitDualVariableValues::addCon (int *numberOfCon*, InitDualVarValue ** *con*)

Alternative signature for this function.

A function to add an array of <con> elements simultaneously

Parameters

<i>numberOfCon</i> ,:	number of <con> elements to be set
<i>obj</i> ,:	the array of <con> elements that are to be set

6.49.4 Member Data Documentation

6.49.4.1 int InitDualVariableValues::numberOfCon

number of <con> children

Definition at line 2992 of file OSOption.h.

6.49.4.2 InitDualVarValue** InitDualVariableValues::con

initial dual values for each constraint

Definition at line 2995 of file OSOption.h.

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

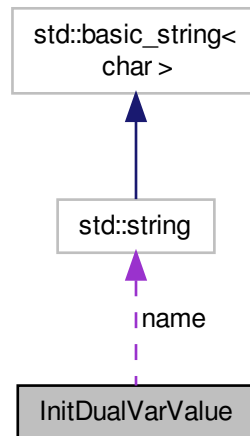
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.50 InitDualVarValue Class Reference

the [InitDualVarValue](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitDualVarValue:



Public Member Functions

- [InitDualVarValue](#) ()
Default constructor.
- [~InitDualVarValue](#) ()
Class destructor.
- bool [isEqual](#) ([InitDualVarValue](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InitDualVarValue](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
constraint index
- std::string [name](#)
optional variable name
- double [lbDualValue](#)
initial lower bound
- double [ubDualValue](#)
initial upper bound

6.50.1 Detailed Description

the [InitDualVarValue](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2924 of file OSOption.h.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 InitDualVarValue::InitDualVarValue ()

Default constructor.

6.50.2.2 InitDualVarValue::~~InitDualVarValue ()

Class destructor.

6.50.3 Member Function Documentation

6.50.3.1 bool InitDualVarValue::isEqual (InitDualVarValue * *that*)

A function to check for the equality of two objects.

6.50.3.2 bool InitDualVarValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.50.3.3 bool InitDualVarValue::deepCopyFrom (InitDualVarValue * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.50.4 Member Data Documentation

6.50.4.1 int InitDualVarValue::idx

constraint index

Definition at line 2929 of file OSOption.h.

6.50.4.2 std::string InitDualVarValue::name

optional variable name

Definition at line 2932 of file OSOption.h.

6.50.4.3 double InitDualVarValue::lbDualValue

initial lower bound

Definition at line 2935 of file OSOption.h.

6.50.4.4 double InitDualVarValue::ubDualValue

initial upper bound

Definition at line 2938 of file OSOption.h.

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

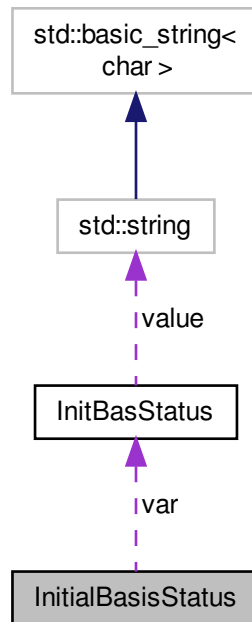
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.51 InitialBasisStatus Class Reference

the [InitialBasisStatus](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitialBasisStatus:



Public Member Functions

- [InitialBasisStatus](#) ()
Default constructor.
- [~InitialBasisStatus](#) ()
Class destructor.
- bool [isEqual](#) ([InitialBasisStatus](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InitialBasisStatus](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setVar](#) (int [numberOfVar](#), [InitBasStatus](#) **var)
A function to set an array of elements.
- bool [addVar](#) (int idx, std::string value)
A function to add a element.

Public Attributes

- int [numberOfVar](#)

number of children

- [InitBasStatus](#) **** var**

initial value for each variable

6.51.1 Detailed Description

the [InitialBasisStatus](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. As of OS version 2.3 this class has been superseded by the class [BasisStatus](#) (see [OSGeneral.h](#))

Definition at line 1540 of file OSOption.h.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 InitialBasisStatus::InitialBasisStatus ()

Default constructor.

6.51.2.2 InitialBasisStatus::~~InitialBasisStatus ()

Class destructor.

6.51.3 Member Function Documentation

6.51.3.1 bool InitialBasisStatus::IsEqual (InitialBasisStatus * that)

A function to check for the equality of two objects.

6.51.3.2 bool InitialBasisStatus::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.51.3.3 bool InitialBasisStatus::deepCopyFrom (InitialBasisStatus * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.51.3.4 bool InitialBasisStatus::setVar (int *numberOfVar*, InitBasStatus ** *var*)

A function to set an array of *elements*.

Parameters

<i>numberOfVar,:</i>	number of <i>elements</i> to be set
<i>var,:</i>	the array of <i>elements</i> to be that are to be set

6.51.3.5 bool InitialBasisStatus::addVar (int *idx*, std::string *value*)

A function to add a *element*.

Parameters

<i>idx,:</i>	the index of the variable to be given an initial basis status
<i>value,:</i>	the initial basis status to be added

6.51.4 Member Data Documentation

6.51.4.1 int InitialBasisStatus::numberOfVar

number of *children*

Definition at line 1545 of file OSOption.h.

6.51.4.2 InitBasStatus** InitialBasisStatus::var

initial value for each variable

Definition at line 1548 of file OSOption.h.

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

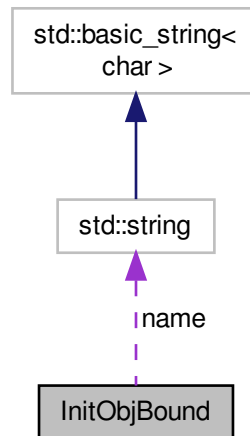
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.52 InitObjBound Class Reference

the [InitObjBound](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitObjBound:



Public Member Functions

- [InitObjBound \(\)](#)
Default constructor.
- [~InitObjBound \(\)](#)
Class destructor.
- [bool IsEqual \(InitObjBound *that\)](#)
A function to check for the equality of two objects.
- [bool setRandom \(double density, bool conformant\)](#)
A function to make a random instance of this class.
- [bool deepCopyFrom \(InitObjBound *that\)](#)
A function to make a deep copy of an instance of this class.

Public Attributes

- [int idx](#)
objective index
- [std::string name](#)
optional variable name
- [double lbValue](#)
initial lower bound
- [double ubValue](#)
initial upper bound

6.52.1 Detailed Description

the [InitObjBound](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2343 of file OSOption.h.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 InitObjBound::InitObjBound ()

Default constructor.

6.52.2.2 InitObjBound::~InitObjBound ()

Class destructor.

6.52.3 Member Function Documentation

6.52.3.1 bool InitObjBound::isEqual (InitObjBound * *that*)

A function to check for the equality of two objects.

6.52.3.2 bool InitObjBound::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.52.3.3 bool InitObjBound::deepCopyFrom (InitObjBound * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.52.4 Member Data Documentation

6.52.4.1 int InitObjBound::idx

objective index

Definition at line 2348 of file OSOption.h.

6.52.4.2 std::string InitObjBound::name

optional variable name

Definition at line 2351 of file OSOption.h.

6.52.4.3 double InitObjBound::lbValue

initial lower bound

Definition at line 2354 of file OSOption.h.

6.52.4.4 double InitObjBound::ubValue

initial upper bound

Definition at line 2357 of file OSOption.h.

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

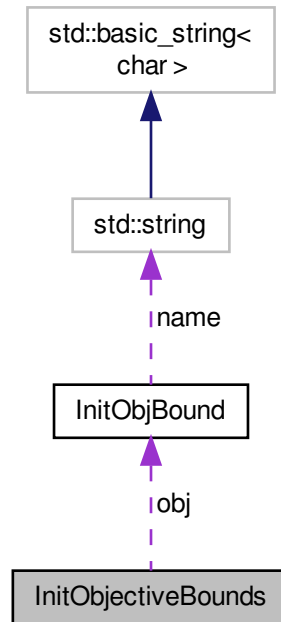
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.53 InitObjectiveBounds Class Reference

the [InitObjectiveBounds](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitObjectiveBounds:



Public Member Functions

- [InitObjectiveBounds](#) ()
Default constructor.
- [~InitObjectiveBounds](#) ()
Class destructor.
- bool [isEqual](#) ([InitObjectiveBounds](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InitObjectiveBounds](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setObj](#) (int [numberOfObj](#), [InitObjBound](#) **obj)
A function to set an array of <obj> elements.
- bool [setObj](#) (int [numberOfObj](#), [InitObjBound](#) **obj, [ENUM_COMBINE_ARRAYS](#) disp)
Alternative signature for this function.
- bool [setObj](#) (int [numberOfObj](#), int *idx, double *lbValue, double *ubValue, std::string *name)
Another alternative signature for this function.
- bool [addObj](#) (int idx, double lbValue, double ubValue)
A function to add a <obj> element.
- bool [addObj](#) (int [numberOfObj](#), [InitObjBound](#) **obj)
Alternative signature for this function.

Public Attributes

- int `numberOfObj`
number of <obj> children
- `InitObjBound ** obj`
initial bounds for each objective

6.53.1 Detailed Description

the `InitObjectiveBounds` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2405 of file OSOption.h.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 `InitObjectiveBounds::InitObjectiveBounds ()`

Default constructor.

6.53.2.2 `InitObjectiveBounds::~~InitObjectiveBounds ()`

Class destructor.

6.53.3 Member Function Documentation

6.53.3.1 `bool InitObjectiveBounds::isEqual (InitObjectiveBounds * that)`

A function to check for the equality of two objects.

6.53.3.2 `bool InitObjectiveBounds::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.53.3.3 `bool InitObjectiveBounds::deepCopyFrom (InitObjectiveBounds * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.53.3.4 `bool InitObjectiveBounds::setObj (int numberOfObj, InitObjBound ** obj)`

A function to set an array of <obj> elements.

Parameters

<i>numberOfObj,:</i>	number of <obj> elements to be set
<i>obj,:</i>	the array of <obj> elements that are to be set

6.53.3.5 `bool InitObjectiveBounds::setObj (int numberOfObj, InitObjBound ** obj, ENUM_COMBINE_ARRAYS disp)`

Alternative signature for this function.

Parameters

<i>numberOfVar,:</i>	number of <i>elements to be set</i>
<i>var,:</i>	<i>the array of elements that are to be set</i>
<i>disp,:</i>	<i>method of disposition if previous data exist</i>

6.53.3.6 `bool InitObjectiveBounds::setObj (int numberOfObj, int * idx, double * lbValue, double * ubValue, std::string * name)`

Another alternative signature for this function.

Parameters

<i>numberOfObj,:</i>	number of <obj> elements to be set
<i>idx,:</i>	the array of indices
<i>lbValue,:</i>	the array of corresponding lower bounds
<i>ubValue,:</i>	the array of corresponding upper bounds
<i>name,:</i>	the array of objective names

6.53.3.7 `bool InitObjectiveBounds::addObj (int idx, double lbValue, double ubValue)`

A function to add a <obj> element.

Parameters

<i>idx,:</i>	the index of the objective to be given initial bounds
<i>lbValue,:</i>	the initial lower bound for the objective
<i>ubValue,:</i>	the initial upper bound for the objective

6.53.3.8 `bool InitObjectiveBounds::addObj (int numberOfObj, InitObjBound ** obj)`

Alternative signature for this function.

A function to add an array of <obj> elements simultaneously

Parameters

<i>numberOfObj</i> :	number of <obj> elements to be set
<i>obj</i> :	the array of <obj> elements that are to be set

6.53.4 Member Data Documentation

6.53.4.1 `int InitObjectiveBounds::numberOfObj`

number of <obj> children

Definition at line 2410 of file OSOption.h.

6.53.4.2 `InitObjBound** InitObjectiveBounds::obj`

initial bounds for each objective

Definition at line 2413 of file OSOption.h.

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

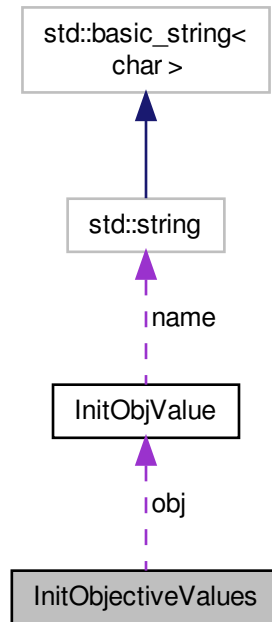
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.54 InitObjectiveValues Class Reference

the [InitObjectiveValues](#) class.

```
#include <OSOption.h>
```


Collaboration diagram for InitObjectiveValues:



Public Member Functions

- [InitObjectiveValues](#) ()
Default constructor.
- [~InitObjectiveValues](#) ()
Class destructor.
- `bool` [isEqual](#) ([InitObjectiveValues](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([InitObjectiveValues](#) *that)
A function to make a deep copy of an instance of this class.
- `bool` [setObj](#) (int [numberOfObj](#), [InitObjValue](#) **obj)
A function to set an array of <obj> elements.
- `bool` [setObj](#) (int [numberOfObj](#), [InitObjValue](#) **obj, [ENUM_COMBINE_ARRAYS](#) disp)
Alternative signature for this function.
- `bool` [setObj](#) (int [numberOfObj](#), int *idx, double *value, [std::string](#) *name)
Another alternative signature for this function.
- `bool` [addObj](#) (int idx, double value)
A function to add a <obj> element.
- `bool` [addObj](#) (int [numberOfObj](#), [InitObjValue](#) **obj)
Alternative signature for this function.

Public Attributes

- int `numberOfObj`
number of <obj> children
- `InitObjValue` ** `obj`
initial value for each objective

6.54.1 Detailed Description

the `InitObjectiveValues` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2241 of file OSOption.h.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 `InitObjectiveValues::InitObjectiveValues ()`

Default constructor.

6.54.2.2 `InitObjectiveValues::~~InitObjectiveValues ()`

Class destructor.

6.54.3 Member Function Documentation

6.54.3.1 `bool InitObjectiveValues::isEqual (InitObjectiveValues * that)`

A function to check for the equality of two objects.

6.54.3.2 `bool InitObjectiveValues::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.54.3.3 bool InitObjectiveValues::deepCopyFrom (InitObjectiveValues * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.54.3.4 bool InitObjectiveValues::setObj (int *numberOfObj*, InitObjValue ** *obj*)

A function to set an array of <obj> elements.

Parameters

<i>numberOfObj,:</i>	number of <obj> elements to be set
<i>obj,:</i>	the array of <obj> elements that are to be set

6.54.3.5 bool InitObjectiveValues::setObj (int *numberOfObj*, InitObjValue ** *obj*, ENUM_COMBINE_ARRAYS *disp*)

Alternative signature for this function.

Parameters

<i>numberOfVar,:</i>	number of <i>elements to be set</i>
<i>var,:</i>	<i>the array of elements that are to be set</i>
<i>disp,:</i>	<i>method of disposition if previous data exist</i>

6.54.3.6 bool InitObjectiveValues::setObj (int *numberOfObj*, int * *idx*, double * *value*, std::string * *name*)

Another alternative signature for this function.

@param *numberOfObj*: number of <obj> elements to be set

Parameters

<i>idx,:</i>	the array of indices
<i>value,:</i>	the array of corresponding values
<i>name,:</i>	the array of objective names

6.54.3.7 bool InitObjectiveValues::addObj (int *idx*, double *value*)

A function to add a <obj> element.

Parameters

<i>idx,:</i>	the index of the objective to be given an initial value
<i>value,:</i>	the initial value to be added

6.54.3.8 `bool InitObjectiveValues::addObj (int numberOfObj, InitObjValue ** obj)`

Alternative signature for this function.

A function to add an array of <obj> elements simultaneously

Parameters

<i>numberOfObj</i> :	number of <obj> elements to be set
<i>obj</i> :	the array of <obj> elements that are to be set

6.54.4 Member Data Documentation

6.54.4.1 `int InitObjectiveValues::numberOfObj`

number of <obj> children

Definition at line 2246 of file OSOption.h.

6.54.4.2 `InitObjValue** InitObjectiveValues::obj`

initial value for each objective

Definition at line 2249 of file OSOption.h.

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

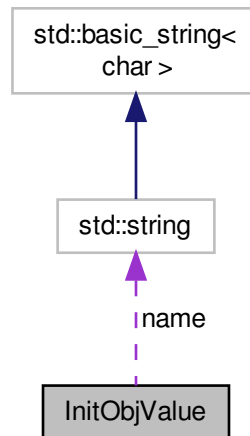
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.55 InitObjValue Class Reference

the [InitObjValue](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitObjValue:



Public Member Functions

- [InitObjValue](#) ()
Default constructor.
- [~InitObjValue](#) ()
Class destructor.
- bool [isEqual](#) (InitObjValue *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) (InitObjValue *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
objective index
- std::string [name](#)
optional objective name
- double [value](#)
initial value

6.55.1 Detailed Description

the [InitObjValue](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2182 of file OSOption.h.

6.55.2 Constructor & Destructor Documentation**6.55.2.1 InitObjValue::InitObjValue ()**

Default constructor.

6.55.2.2 InitObjValue::~~InitObjValue ()

Class destructor.

6.55.3 Member Function Documentation**6.55.3.1 bool InitObjValue::isEqual (InitObjValue * *that*)**

A function to check for the equality of two objects.

6.55.3.2 bool InitObjValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.55.3.3 bool InitObjValue::deepCopyFrom (InitObjValue * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.55.4 Member Data Documentation**6.55.4.1 int InitObjValue::idx**

objective index

Definition at line 2187 of file OSOption.h.

6.55.4.2 std::string InitObjValue::name

optional objective name

Definition at line 2190 of file OSOption.h.

6.55.4.3 double InitObjValue::value

initial value

Definition at line 2193 of file OSOption.h.

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

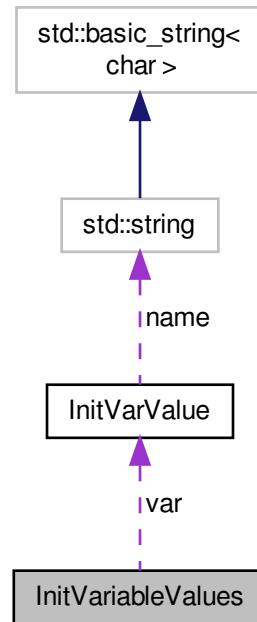
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.56 InitVariableValues Class Reference

the [InitVariableValues](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitVariableValues:



Public Member Functions

- `InitVariableValues ()`
Default constructor.
- `~InitVariableValues ()`
Class destructor.
- `bool isEqual (InitVariableValues *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (InitVariableValues *that)`
A function to make a deep copy of an instance of this class.
- `bool setVar (int numberOfVar, InitVarValue **var)`
A function to set an array of elements.
- `bool setVar (int numberOfVar, InitVarValue **var, ENUM_COMBINE_ARRAYS disp)`
Alternative signature for this function.
- `bool setVar (int numberOfVar, int *idx, double *value, std::string *name)`
Another alternative signature for this function.
- `bool addVar (int idx, double value)`
A function to add a element.
- `bool addVar (int numberOfVar, InitVarValue **var)`
Alternative signature for this function.

Public Attributes

- int `numberOfVar`
number of children
- `InitVarValue` ** `var`
initial value for each variable

6.56.1 Detailed Description

the `InitVariableValues` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1218 of file `OSOption.h`.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 `InitVariableValues::InitVariableValues ()`

Default constructor.

6.56.2.2 `InitVariableValues::~~InitVariableValues ()`

Class destructor.

6.56.3 Member Function Documentation

6.56.3.1 `bool InitVariableValues::isEqual (InitVariableValues * that)`

A function to check for the equality of two objects.

6.56.3.2 `bool InitVariableValues::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.56.3.3 `bool InitVariableValues::deepCopyFrom (InitVariableValues * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.56.3.4 `bool InitVariableValues::setVar (int numberOfVar, InitVarValue ** var)`

A function to set an array of *elements*.

Parameters

<i>numberOfVar,:</i>	number of <i>elements</i> to be set
<i>var,:</i>	the array of <i>elements</i> that are to be set

6.56.3.5 `bool InitVariableValues::setVar (int numberOfVar, InitVarValue ** var, ENUM_COMBINE_ARRAYS disp)`

Alternative signature for this function.

Parameters

<i>numberOfVar,:</i>	number of <i>elements</i> to be set
<i>var,:</i>	the array of <i>elements</i> that are to be set
<i>disp,:</i>	method of disposition if previous data exist

6.56.3.6 `bool InitVariableValues::setVar (int numberOfVar, int * idx, double * value, std::string * name)`

Another alternative signature for this function.

Parameters

<i>numberOfVar,:</i>	number of <i>elements</i> to be set
<i>idx,:</i>	the array of <i>indices</i>
<i>value,:</i>	the array of corresponding <i>values</i>
<i>name,:</i>	the array of corresponding <i>names</i>

6.56.3.7 `bool InitVariableValues::addVar (int idx, double value)`

A function to add a *element*.

Parameters

<i>idx,:</i>	the index of the variable to be given an initial value
<i>value,:</i>	the initial variable value to be added

6.56.3.8 `bool InitVariableValues::addVar (int numberOfVar, InitVarValue ** var)`

Alternative signature for this function.

A function to add an array of *elements simultaneously*

Parameters

numberOfVar,:	<i>number of elements to be set</i>
var,:	<i>the array of elements that are to be set</i>

6.56.4 Member Data Documentation**6.56.4.1 int InitVariableValues::numberOfVar**

number of *children*

Definition at line 1223 of file OSOption.h.

6.56.4.2 InitVarValue InitVariableValues::var**

initial value for each variable

Definition at line 1226 of file OSOption.h.

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

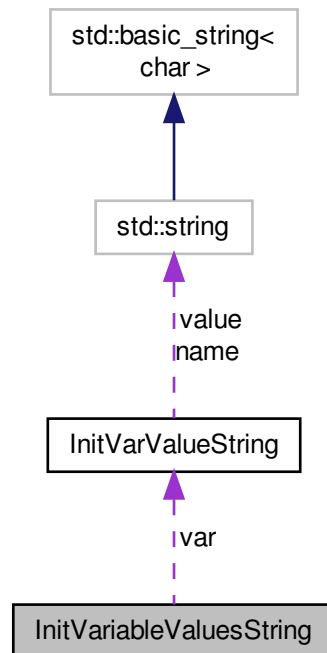
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.57 InitVariableValuesString Class Reference

the [InitVariableValuesString](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitVariableValuesString:



Public Member Functions

- [InitVariableValuesString](#) ()
Default constructor.
- [~InitVariableValuesString](#) ()
Class destructor.
- [bool IsEqual](#) ([InitVariableValuesString](#) *that)
A function to check for the equality of two objects.
- [bool setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- [bool deepCopyFrom](#) ([InitVariableValuesString](#) *that)
A function to make a deep copy of an instance of this class.
- [bool setVar](#) (int numberOfVar, [InitVarValueString](#) **var)
A function to set an array of elements.
- [bool setVar](#) (int numberOfVar, [InitVarValueString](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
Alternative signature for this function.
- [bool setVar](#) (int numberOfVar, int *idx, [std::string](#) *value, [std::string](#) *name)
Another alternative signature for this function.
- [bool addVar](#) (int idx, [std::string](#) value)
A function to add a element.

- bool `addVar` (int `numberOfVar`, `InitVarValueString` ***var*)

Alternative signature for this function.

Public Attributes

- int `numberOfVar`
number of children
- `InitVarValueString` *** var*
initial value for each variable

6.57.1 Detailed Description

the `InitVariableValuesString` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1379 of file OSOption.h.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `InitVariableValuesString::InitVariableValuesString ()`

Default constructor.

6.57.2.2 `InitVariableValuesString::~~InitVariableValuesString ()`

Class destructor.

6.57.3 Member Function Documentation

6.57.3.1 `bool InitVariableValuesString::isEqual (InitVariableValuesString * that)`

A function to check for the equality of two objects.

6.57.3.2 bool InitVariableValuesString::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.57.3.3 bool InitVariableValuesString::deepCopyFrom (InitVariableValuesString * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.57.3.4 bool InitVariableValuesString::setVar (int *numberOfVar*, InitVarValueString ** *var*)

A function to set an array of *elements*.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements to be set</i>
<i>var</i> ,:	<i>the array of elements that are to be set</i>

6.57.3.5 bool InitVariableValuesString::setVar (int *numberOfVar*, InitVarValueString ** *var*, ENUM_COMBINE_ARRAYS *disp*)

Alternative signature for this function.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements to be set</i>
<i>var</i> ,:	<i>the array of elements that are to be set</i>
<i>disp</i> ,:	<i>method of disposition if previous data exist</i>

6.57.3.6 bool InitVariableValuesString::setVar (int *numberOfVar*, int * *idx*, std::string * *value*, std::string * *name*)

Another alternative signature for this function.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements to be set</i>
<i>idx</i> ,:	<i>the array of indices</i>
<i>value</i> ,:	<i>the array of corresponding values</i>
<i>name</i> ,:	<i>the array of corresponding names</i>

6.57.3.7 bool InitVariableValuesString::addVar (int *idx*, std::string *value*)

A function to add a *element*.

Parameters

<i>idx</i> ,:	the index of the variable to be given an initial value
<i>value</i> ,:	the initial string value to be added

6.57.3.8 bool InitVariableValuesString::addVar (int *numberOfVar*, InitVarValueString ** *var*)

Alternative signature for this function.

A function to add an array of *elements simultaneously*

Parameters

<i>numberOfVar</i> ,:	<i>number of elements to be set</i>
<i>var</i> ,:	<i>the array of elements that are to be set</i>

6.57.4 Member Data Documentation

6.57.4.1 int InitVariableValuesString::numberOfVar

number of *children*

Definition at line 1384 of file OSOption.h.

6.57.4.2 InitVarValueString** InitVariableValuesString::var

initial value for each variable

Definition at line 1387 of file OSOption.h.

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

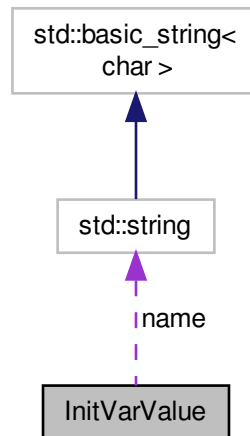
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.58 InitVarValue Class Reference

the [InitVarValue](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitVarValue:



Public Member Functions

- `InitVarValue ()`
Default constructor.
- `~InitVarValue ()`
Class destructor.
- `bool IsEqual (InitVarValue *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (InitVarValue *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int idx`
variable index
- `std::string name`
optional variable name
- `double value`
initial value

6.58.1 Detailed Description

the `InitVarValue` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1159 of file OSOption.h.

6.58.2 Constructor & Destructor Documentation**6.58.2.1 InitVarValue::InitVarValue ()**

Default constructor.

6.58.2.2 InitVarValue::~~InitVarValue ()

Class destructor.

6.58.3 Member Function Documentation**6.58.3.1 bool InitVarValue::isEqual (InitVarValue * *that*)**

A function to check for the equality of two objects.

6.58.3.2 bool InitVarValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.58.3.3 bool InitVarValue::deepCopyFrom (InitVarValue * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.58.4 Member Data Documentation**6.58.4.1 int InitVarValue::idx**

variable index

Definition at line 1164 of file OSOption.h.

6.58.4.2 std::string InitVarValue::name

optional variable name

Definition at line 1167 of file OSOption.h.

6.58.4.3 double InitVarValue::value

initial value

Definition at line 1170 of file OSOption.h.

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

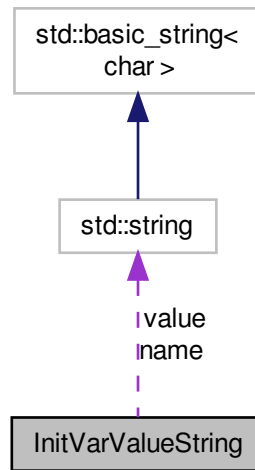
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.59 InitVarValueString Class Reference

the [InitVarValueString](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InitVarValueString:



Public Member Functions

- `InitVarValueString ()`
Default constructor.
- `~InitVarValueString ()`
Class destructor.
- `bool isEqual (InitVarValueString *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (InitVarValueString *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int idx`
variable index
- `std::string name`
optional variable name
- `std::string value`
initial value

6.59.1 Detailed Description

the [InitVarValueString](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1320 of file OSOption.h.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 InitVarValueString::InitVarValueString ()

Default constructor.

6.59.2.2 InitVarValueString::~InitVarValueString ()

Class destructor.

6.59.3 Member Function Documentation

6.59.3.1 bool InitVarValueString::isEqual (InitVarValueString * *that*)

A function to check for the equality of two objects.

6.59.3.2 bool InitVarValueString::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.59.3.3 bool InitVarValueString::deepCopyFrom (InitVarValueString * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.59.4 Member Data Documentation

6.59.4.1 int InitVarValueString::idx

variable index

Definition at line 1325 of file OSOption.h.

6.59.4.2 std::string InitVarValueString::name

optional variable name

Definition at line 1328 of file OSOption.h.

6.59.4.3 std::string InitVarValueString::value

initial value

Definition at line 1331 of file OSOption.h.

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

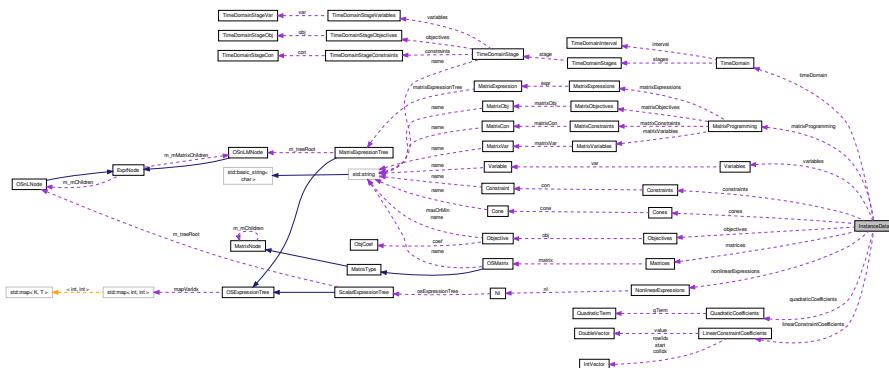
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h

6.60 InstanceData Class Reference

The in-memory representation of the <**instanceData**> element.

```
#include <OSInstance.h>
```

Collaboration diagram for InstanceData:



Public Member Functions

- [InstanceData\(\)](#)

The *InstanceData* class constructor.

- `~InstanceData ()`

The *InstanceData* class destructor.

- `bool IsEqual (InstanceData *that)`

A function to check for the equality of two objects.

Public Attributes

- `Variables * variables`

variables is a pointer to a *Variables* object

- `Objectives * objectives`

objectives is a pointer to a *Objectives* object

- `Constraints * constraints`

constraints is a pointer to a *Constraints* object

- `LinearConstraintCoefficients * linearConstraintCoefficients`

linearConstraintCoefficients is a pointer to a *LinearConstraintCoefficients* object

- `QuadraticCoefficients * quadraticCoefficients`

quadraticCoefficients is a pointer to a *QuadraticCoefficients* object

- `NonlinearExpressions * nonlinearExpressions`

nonlinearExpressions is a pointer to a *NonlinearExpressions* object

- `Matrices * matrices`

matrices is a pointer to a *Matrices* object

- `Cones * cones`

cones is a pointer to a *Cones* object

- `MatrixProgramming * matrixProgramming`

matrixProgramming is a pointer to a *MatrixProgramming* object

- `TimeDomain * timeDomain`

timeDomain is a pointer to a *TimeDomain* object

6.60.1 Detailed Description

The in-memory representation of the `<InstanceData>` element.

Remarks

The *InstanceData* object contains the objects that define the instance –

- **Variables** object
- **Objectives** object
- **Constraints** object
- **LinearConstraintCoefficients** object
- **QuadraticCoefficients** object
- **NonlinearExpressions/b>** object
- **TimeDomain/b>** object

Definition at line 2176 of file *OSInstance.h*.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 InstanceData::InstanceData ()

The [InstanceData](#) class constructor.

6.60.2.2 InstanceData::~~InstanceData ()

The [InstanceData](#) class destructor.

6.60.3 Member Function Documentation

6.60.3.1 bool InstanceData::isEqual (InstanceData * *that*)

A function to check for the equality of two objects.

6.60.4 Member Data Documentation

6.60.4.1 Variables* InstanceData::variables

variables is a pointer to a [Variables](#) object

Definition at line 2187 of file OSInstance.h.

6.60.4.2 Objectives* InstanceData::objectives

objectives is a pointer to a [Objectives](#) object

Definition at line 2190 of file OSInstance.h.

6.60.4.3 Constraints* InstanceData::constraints

constraints is a pointer to a [Constraints](#) object

Definition at line 2193 of file OSInstance.h.

6.60.4.4 LinearConstraintCoefficients* InstanceData::linearConstraintCoefficients

linearConstraintCoefficients is a pointer to a [LinearConstraintCoefficients](#) object

Definition at line 2198 of file OSInstance.h.

6.60.4.5 QuadraticCoefficients* InstanceData::quadraticCoefficients

quadraticCoefficients is a pointer to a [QuadraticCoefficients](#) object

Definition at line 2203 of file OSInstance.h.

6.60.4.6 NonlinearExpressions* InstanceData::nonlinearExpressions

nonlinearExpressions is a pointer to a [NonlinearExpressions](#) object

Definition at line 2208 of file OSInstance.h.

6.60.4.7 Matrices* InstanceData::matrices

matrices is a pointer to a [Matrices](#) object

Definition at line 2213 of file OSInstance.h.

6.60.4.8 Cones* InstanceData::cones

cones is a pointer to a [Cones](#) object

Definition at line 2218 of file OSInstance.h.

6.60.4.9 MatrixProgramming* InstanceData::matrixProgramming

matrixProgramming is a pointer to a [MatrixProgramming](#) object

Definition at line 2223 of file OSInstance.h.

6.60.4.10 TimeDomain* InstanceData::timeDomain

timeDomain is a pointer to a [TimeDomain](#) object

Definition at line 2228 of file OSInstance.h.

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

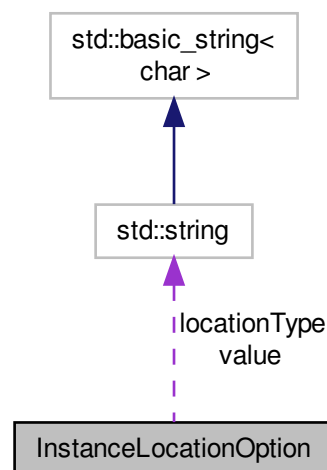
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.61 InstanceLocationOption Class Reference

the [InstanceLocationOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for InstanceLocationOption:



Public Member Functions

- [InstanceLocationOption](#) ()
Default constructor.
- [~InstanceLocationOption](#) ()
Class destructor.
- bool [IsEqual](#) ([InstanceLocationOption](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([InstanceLocationOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string [locationType](#)
the type of the location
- std::string [value](#)
the value of the <instanceLocation> element

6.61.1 Detailed Description

the [InstanceLocationOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to the instanceLocation element in the OSoL schema.

Definition at line 39 of file OSOption.h.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 [InstanceLocationOption::InstanceLocationOption](#) ()

Default constructor.

6.61.2.2 [InstanceLocationOption::~~InstanceLocationOption](#) ()

Class destructor.

6.61.3 Member Function Documentation

6.61.3.1 `bool InstanceLocationOption::isEqual (InstanceLocationOption * that)`

A function to check for the equality of two objects.

6.61.3.2 `bool InstanceLocationOption::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.61.3.3 `bool InstanceLocationOption::deepCopyFrom (InstanceLocationOption * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.61.4 Member Data Documentation

6.61.4.1 `std::string InstanceLocationOption::locationType`

the type of the location

Definition at line 44 of file `OSOption.h`.

6.61.4.2 `std::string InstanceLocationOption::value`

the value of the <instanceLocation> element

Definition at line 47 of file `OSOption.h`.

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

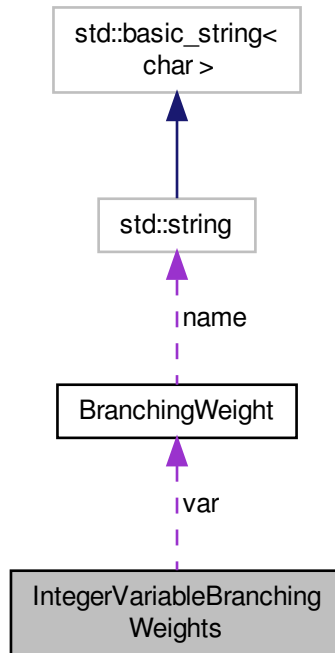
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.62 IntegerVariableBranchingWeights Class Reference

the [IntegerVariableBranchingWeights](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for IntegerVariableBranchingWeights:



Public Member Functions

- [IntegerVariableBranchingWeights](#) ()
Default constructor.
- [~IntegerVariableBranchingWeights](#) ()
Class destructor.
- bool [isEqual](#) ([IntegerVariableBranchingWeights](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([IntegerVariableBranchingWeights](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setVar](#) (int [numberOfVar](#), [BranchingWeight](#) **var)
A function to set an array of elements.
- bool [setVar](#) (int [numberOfVar](#), [BranchingWeight](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
Alternative signature for this function.
- bool [setVar](#) (int [numberOfVar](#), int *idx, double *value, std::string *name)
Another alternative signature for this function.
- bool [addVar](#) (int idx, double value)
A function to add a element.

- bool `addVar` (int `numberOfVar`, `BranchingWeight` **`var`)

Alternative signature for this function.

Public Attributes

- int `numberOfVar`
number of children
- `BranchingWeight` ** `var`
branching weight for each variable

6.62.1 Detailed Description

the `IntegerVariableBranchingWeights` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/11/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1671 of file `OSOption.h`.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 `IntegerVariableBranchingWeights::IntegerVariableBranchingWeights ()`

Default constructor.

6.62.2.2 `IntegerVariableBranchingWeights::~~IntegerVariableBranchingWeights ()`

Class destructor.

6.62.3 Member Function Documentation

6.62.3.1 `bool IntegerVariableBranchingWeights::isEqual (IntegerVariableBranchingWeights * that)`

A function to check for the equality of two objects.

6.62.3.2 `bool IntegerVariableBranchingWeights::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.62.3.3 `bool IntegerVariableBranchingWeights::deepCopyFrom (IntegerVariableBranchingWeights * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.62.3.4 `bool IntegerVariableBranchingWeights::setVar (int numberOfVar, BranchingWeight ** var)`

A function to set an array of *elements*.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements</i> to be set
<i>var</i> ,:	<i>the array of elements to be that are to be set</i>

6.62.3.5 `bool IntegerVariableBranchingWeights::setVar (int numberOfVar, BranchingWeight ** var, ENUM_COMBINE_ARRAYS disp)`

Alternative signature for this function.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements</i> to be set
<i>var</i> ,:	<i>the array of elements that are to be set</i>
<i>disp</i> ,:	<i>method of disposition if previous data exist</i>

6.62.3.6 `bool IntegerVariableBranchingWeights::setVar (int numberOfVar, int * idx, double * value, std::string * name)`

Another alternative signature for this function.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements</i> to be set
<i>idx</i> ,:	<i>the array of indices</i>
<i>value</i> ,:	<i>the array of corresponding values</i>
<i>name</i> ,:	<i>the array of corresponding names</i>

6.62.3.7 `bool IntegerVariableBranchingWeights::addVar (int idx, double value)`

A function to add a *element*.

Parameters

<i>idx</i> ,:	the index of the variable to be given a branching weight
<i>value</i> ,:	the branching weight to be added

6.62.3.8 `bool IntegerVariableBranchingWeights::addVar (int numberOfVar, BranchingWeight ** var)`

Alternative signature for this function.

A function to add an array of *elements simultaneously*

Parameters

<i>numberOfVar</i> ,:	<i>number of elements to be set</i>
<i>var</i> ,:	<i>the array of elements that are to be set</i>

6.62.4 Member Data Documentation

6.62.4.1 `int IntegerVariableBranchingWeights::numberOfVar`

number of *children*

Definition at line 1676 of file `OSOption.h`.

6.62.4.2 `BranchingWeight** IntegerVariableBranchingWeights::var`

branching weight for each variable

Definition at line 1679 of file `OSOption.h`.

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

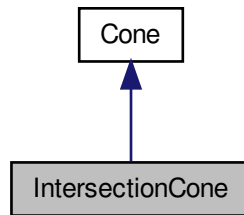
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h`

6.63 IntersectionCone Class Reference

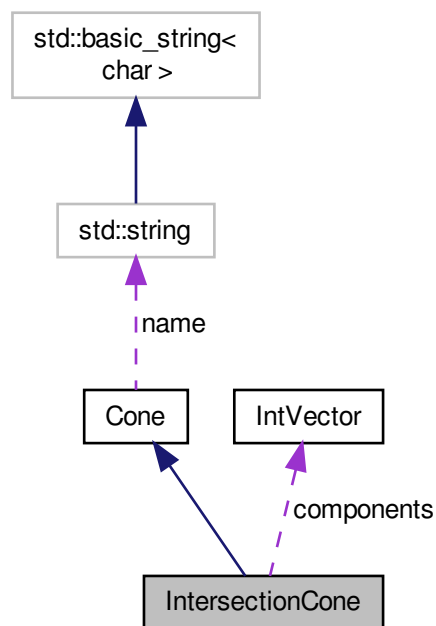
The in-memory representation of an intersection cone.

```
#include <OSInstance.h>
```

Inheritance diagram for IntersectionCone:



Collaboration diagram for IntersectionCone:



Public Member Functions

- [IntersectionCone](#) ()
The *IntersectionCone* class constructor.
- [~IntersectionCone](#) ()

The [IntersectionCone](#) class destructor.

- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()

Write an [IntersectionCone](#) object in XML format.

- bool [IsEqual](#) ([IntersectionCone](#) *that)
- A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
- A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([IntersectionCone](#) *that)
- A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
- Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
- Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
- The type of the cone (one of the values in `ENUM_CONE_TYPE`)
- int [idx](#)
- cones are referenced by an (automatically created) index
- [IntVector](#) * [components](#)
- the list of components contributing to the intersection each component contains a reference to a previously defined cone

6.63.1 Detailed Description

The in-memory representation of an intersection cone.

Definition at line 1325 of file `OSInstance.h`.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 [IntersectionCone::IntersectionCone](#) ()

The [IntersectionCone](#) class constructor.

6.63.2.2 [IntersectionCone::~~IntersectionCone](#) ()

The [IntersectionCone](#) class destructor.

6.63.3 Member Function Documentation

6.63.3.1 virtual std::string [IntersectionCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.63.3.2 `virtual std::string IntersectionCone::getConeInXML () [virtual]`

Write an [IntersectionCone](#) object in XML format.

This is used by [OSILWriter](#) to write a `<cone>` element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.63.3.3 `bool IntersectionCone::isEqual (IntersectionCone * that)`

A function to check for the equality of two objects.

6.63.3.4 `bool IntersectionCone::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <code><XXX></code> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.63.3.5 `bool IntersectionCone::deepCopyFrom (IntersectionCone * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.63.4 Member Data Documentation

6.63.4.1 `int IntersectionCone::numberOfRows`

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 1338 of file `OSInstance.h`.

6.63.4.2 `int IntersectionCone::numberOfColumns`

Definition at line 1339 of file `OSInstance.h`.

6.63.4.3 `int IntersectionCone::numberOfOtherIndexes`

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 1346 of file `OSInstance.h`.

6.63.4.4 `int* IntersectionCone::otherIndexes`

Definition at line 1347 of file `OSInstance.h`.

6.63.4.5 `int IntersectionCone::coneType`

The type of the cone (one of the values in `ENUM_CONE_TYPE`)

Definition at line 1350 of file `OSInstance.h`.

6.63.4.6 `int IntersectionCone::idx`

cones are referenced by an (automatically created) index

Definition at line 1353 of file `OSInstance.h`.

6.63.4.7 `IntVector* IntersectionCone::components`

the list of components contributing to the intersection each component contains a reference to a previously defined cone

Definition at line 1358 of file `OSInstance.h`.

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

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h`

6.64 Interval Class Reference

The in-memory representation of the `<interval>` element.

```
#include <OSInstance.h>
```

6.64.1 Detailed Description

The in-memory representation of the `<interval>` element.

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

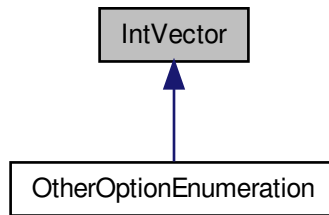
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h`

6.65 IntVector Class Reference

an integer Vector data structure

```
#include <OSGeneral.h>
```

Inheritance diagram for IntVector:



Public Member Functions

- [IntVector](#) ()
- [~IntVector](#) ()
- [IntVector](#) (int n)
alternate constructor
- bool [isEqual](#) ([IntVector](#) *that)
A method to compare two invectors.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([IntVector](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setIntVector](#) (int *i, int ni)
set values into an [IntVector](#)
- bool [extendIntVector](#) (int i)
append a value to an [IntVector](#)
- int [getNumberOfEI](#) ()
get the dimension of an [IntVector](#)
- int [getEI](#) (int j)
get an entry in the data array of an [IntVector](#)
- bool [getEI](#) (int *i)
Get the integer data array of an [IntVector](#).

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- int [numberOfEI](#)
- int * [el](#)

6.65.1 Detailed Description

an integer Vector data structure

Definition at line 469 of file OSGeneral.h.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 IntVector::IntVector ()

6.65.2.2 IntVector::~~IntVector ()

6.65.2.3 IntVector::IntVector (int *n*)

alternate constructor

6.65.3 Member Function Documentation

6.65.3.1 bool IntVector::isEqual (IntVector * *that*)

A method to compare two invectors.

6.65.3.2 bool IntVector::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest value (inclusive) that an entry in this vector can take
<i>iMax</i> ,:	greatest value (inclusive) that an entry in this vector can take

Reimplemented in [OtherOptionEnumeration](#).

6.65.3.3 bool IntVector::deepCopyFrom (IntVector * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.65.3.4 bool IntVector::setIntVector (int * *i*, int *ni*)

set values into an [IntVector](#)

Parameters

<i>ni</i>	contains the dimension of the IntVector
<i>i</i>	contains the array of values

6.65.3.5 bool IntVector::extendIntVector (int *i*)

append a value to an [IntVector](#)

Parameters

<i>i</i>	contains the value to be appended
----------	-----------------------------------

6.65.3.6 int IntVector::getNumberOfEI ()

get the dimension of an [IntVector](#)

6.65.3.7 int IntVector::getEI (int *j*)

get an entry in the data array of an [IntVector](#)

Parameters

<i>j</i>	is the index of the entry that is to be retrieved
----------	---

6.65.3.8 bool IntVector::getEI (int * *i*)

Get the integer data array of an [IntVector](#).

Parameters

<i>i</i>	is the location where the user wants to store the array
----------	---

Returns

the value

Note

it is the user's responsibility to reserve sufficient memory to hold the vector being returned.

6.65.4 Member Data Documentation

6.65.4.1 bool IntVector::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 482 of file OSGeneral.h.

6.65.4.2 int IntVector::numberOfEI

Definition at line 483 of file OSGeneral.h.

6.65.4.3 int* IntVector::ei

Definition at line 484 of file OSGeneral.h.

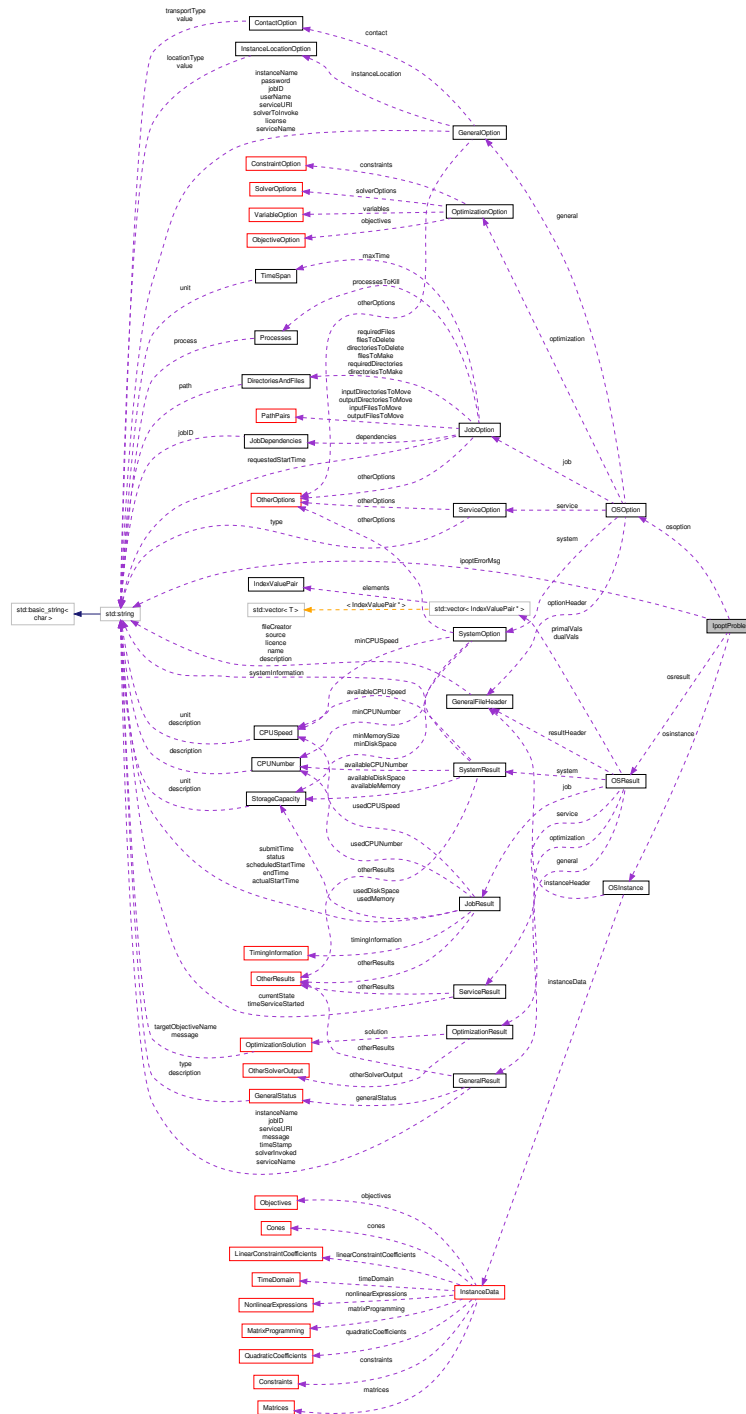
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.66 IpoptProblem Class Reference

```
#include <OSIpoptSolver.h>
```

Collaboration diagram for IpoptProblem:



Public Member Functions

- [IpoptProblem](#) ([OSInstance](#) *osinstance_, [OSOption](#) *osoption_, [OSResult](#) *osresult_, std::string *ipoptErrorMsg_)
the *IpoptProblem* class constructor
- virtual [~IpoptProblem](#) ()
the *IpoptProblem* class destructor
- virtual bool [get_nlp_info](#) (Ipopt::Index &n, Ipopt::Index &m, Ipopt::Index &nnz_jac_g, Ipopt::Index &nnz_h_lag, IndexStyleEnum &index_style)
IPOpt specific methods for defining the nlp problem.
- virtual bool [get_bounds_info](#) (Ipopt::Index n, Ipopt::Number *x_l, Ipopt::Number *x_u, Ipopt::Index m, Ipopt::Number *g_l, Ipopt::Number *g_u)
Method to return the bounds for my problem.
- virtual bool [get_starting_point](#) (Ipopt::Index n, bool init_x, Ipopt::Number *x, bool init_z, Ipopt::Number *z_L, Ipopt::Number *z_U, Ipopt::Index m, bool init_lambda, Ipopt::Number *lambda)
Method to return the starting point for the algorithm.
- virtual bool [eval_f](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number &obj_value)
Method to return the objective value.
- virtual bool [eval_grad_f](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number *grad_f)
Method to return the gradient of the objective.
- virtual bool [eval_g](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Index m, Ipopt::Number *g)
Method to return the constraint residuals.
- virtual bool [eval_jac_g](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Index m, Ipopt::Index nele_jac, Ipopt::Index *iRow, Ipopt::Index *jCol, Ipopt::Number *values)
Method to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)
- virtual bool [eval_h](#) (Ipopt::Index n, const Ipopt::Number *x, bool new_x, Ipopt::Number obj_factor, Ipopt::Index m, const Ipopt::Number *lambda, bool new_lambda, Ipopt::Index nele_hess, Ipopt::Index *iRow, Ipopt::Index *jCol, Ipopt::Number *values)
Method to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)
- virtual bool [get_scaling_parameters](#) (Ipopt::Number &obj_scaling, bool &use_x_scaling, Ipopt::Index n, Ipopt::Number *x_scaling, bool &use_g_scaling, Ipopt::Index m, Ipopt::Number *g_scaling)

Solution Methods

- virtual void [finalize_solution](#) (Ipopt::SolverReturn status, Ipopt::Index n, const Ipopt::Number *x, const Ipopt::Number *z_L, const Ipopt::Number *z_U, Ipopt::Index m, const Ipopt::Number *g, const Ipopt::Number *lambda, Ipopt::Number obj_value, const Ipopt::IpoptData *ip_data, Ipopt::IpoptCalculatedQuantities *ip_cq)

This method is called when the algorithm is complete so the TNLP can store/write the solution.

Public Attributes

- [OSInstance](#) * osinstance
- [OSOption](#) * osoption
- [OSResult](#) * osresult
- std::string * ipoptErrorMsg

6.66.1 Detailed Description

Definition at line 52 of file OSIpoptSolver.h.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `IpoptProblem::IpoptProblem (OSInstance * osinstance_, OSOption * osoption_, OSResult * osresult_, std::string * ipoptErrorMsg_)`

the IpoptProblemclass constructor

6.66.2.2 `virtual IpoptProblem::~~IpoptProblem () [virtual]`

the [IpoptProblem](#) class destructor

6.66.3 Member Function Documentation

6.66.3.1 `virtual bool IpoptProblem::get_nlp_info (Ipopt::Index & n, Ipopt::Index & m, Ipopt::Index & nnz_jac_g, Ipopt::Index & nnz_h_lag, IndexStyleEnum & index_style) [virtual]`

IPOpt specific methods for defining the nlp problem.

6.66.3.2 `virtual bool IpoptProblem::get_bounds_info (Ipopt::Index n, Ipopt::Number * x_l, Ipopt::Number * x_u, Ipopt::Index m, Ipopt::Number * g_l, Ipopt::Number * g_u) [virtual]`

Method to return the bounds for my problem.

6.66.3.3 `virtual bool IpoptProblem::get_starting_point (Ipopt::Index n, bool init_x, Ipopt::Number * x, bool init_z, Ipopt::Number * z_L, Ipopt::Number * z_U, Ipopt::Index m, bool init_lambda, Ipopt::Number * lambda) [virtual]`

Method to return the starting point for the algorithm.

6.66.3.4 `virtual bool IpoptProblem::eval_f (Ipopt::Index n, const Ipopt::Number * x, bool new_x, Ipopt::Number & obj_value) [virtual]`

Method to return the objective value.

6.66.3.5 `virtual bool IpoptProblem::eval_grad_f (Ipopt::Index n, const Ipopt::Number * x, bool new_x, Ipopt::Number * grad_f) [virtual]`

Method to return the gradient of the objective.

6.66.3.6 `virtual bool IpoptProblem::eval_g (Ipopt::Index n, const Ipopt::Number * x, bool new_x, Ipopt::Index m, Ipopt::Number * g) [virtual]`

Method to return the constraint residuals.

6.66.3.7 `virtual bool IpoptProblem::eval_jac_g (Ipopt::Index n, const Ipopt::Number * x, bool new_x, Ipopt::Index m, Ipopt::Index nele_jac, Ipopt::Index * iRow, Ipopt::Index * jCol, Ipopt::Number * values) [virtual]`

Method to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)


```
6.66.3.8 virtual bool IpoptProblem::eval_h ( Ipopt::Index n, const Ipopt::Number * x, bool new_x, Ipopt::Number obj_factor,
      Ipopt::Index m, const Ipopt::Number * lambda, bool new_lambda, Ipopt::Index nele_hess, Ipopt::Index * iRow,
      Ipopt::Index * jCol, Ipopt::Number * values ) [virtual]
```

Method to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)

```
6.66.3.9 virtual bool IpoptProblem::get_scaling_parameters ( Ipopt::Number & obj_scaling, bool & use_x_scaling, Ipopt::Index n,
      Ipopt::Number * x_scaling, bool & use_g_scaling, Ipopt::Index m, Ipopt::Number * g_scaling ) [virtual]
```

```
6.66.3.10 virtual void IpoptProblem::finalize_solution ( Ipopt::SolverReturn status, Ipopt::Index n, const Ipopt::Number * x, const
      Ipopt::Number * z_L, const Ipopt::Number * z_U, Ipopt::Index m, const Ipopt::Number * g, const Ipopt::Number
      * lambda, Ipopt::Number obj_value, const Ipopt::IpoptData * ip_data, Ipopt::IpoptCalculatedQuantities * ip_cq )
      [virtual]
```

This method is called when the algorithm is complete so the TNLP can store/write the solution.

6.66.4 Member Data Documentation

6.66.4.1 OSInstance* IpoptProblem::osinstance

Definition at line 63 of file OSIpoptSolver.h.

6.66.4.2 OSOption* IpoptProblem::osoption

Definition at line 65 of file OSIpoptSolver.h.

6.66.4.3 OSResult* IpoptProblem::osresult

Definition at line 67 of file OSIpoptSolver.h.

6.66.4.4 std::string* IpoptProblem::ipoptErrorMsg

Definition at line 69 of file OSIpoptSolver.h.

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

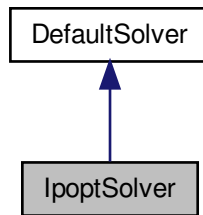
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSIpoptSolver.h](#)

6.67 IpoptSolver Class Reference

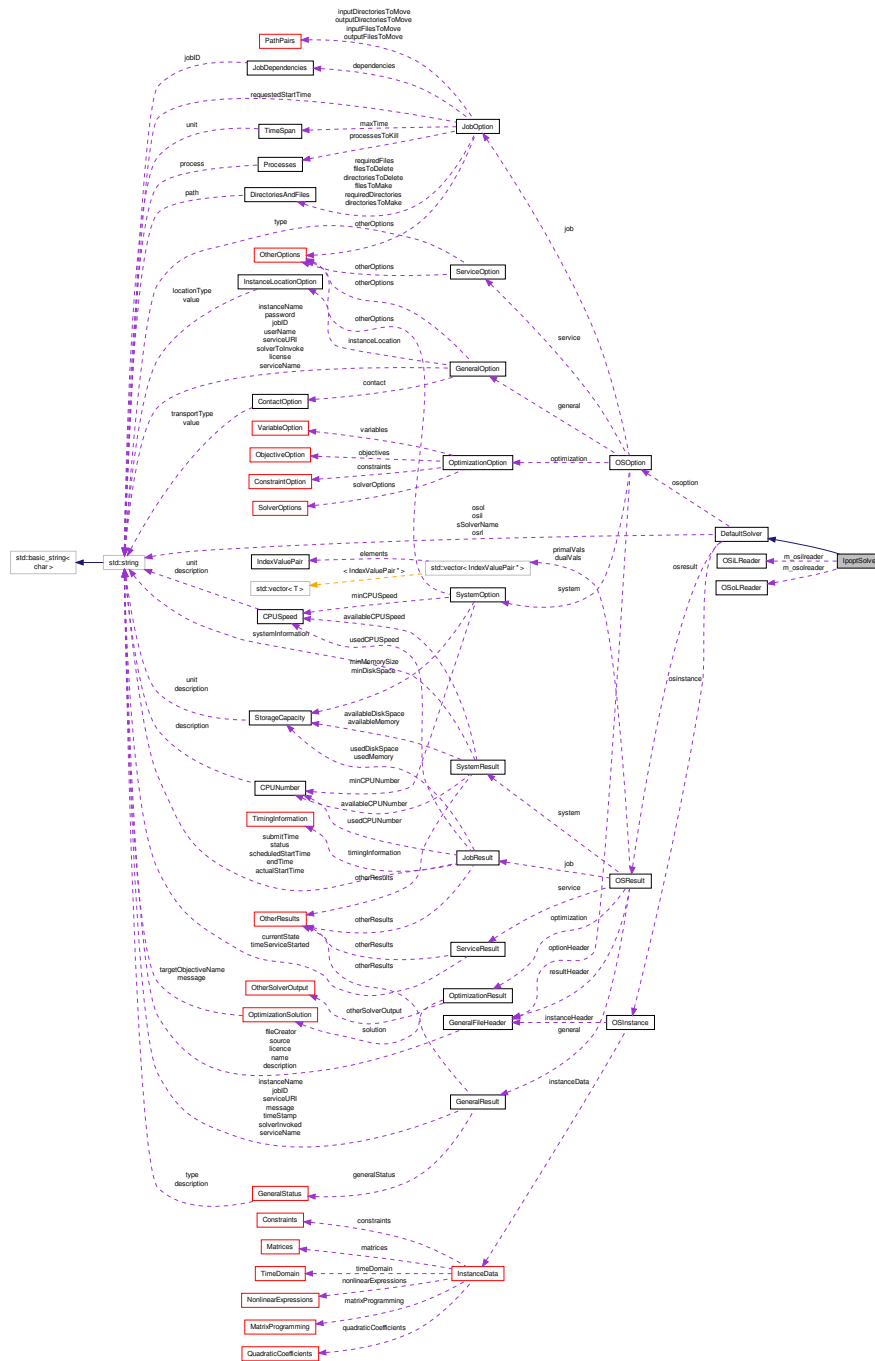
The [IpoptSolver](#) class solves problems using Ipopt.

```
#include <OSIpoptSolver.h>
```

Inheritance diagram for IpoptSolver:



Collaboration diagram for IpoptSolver:



Public Member Functions

- [IpoptSolver\(\)](#)
the *IpoptSolver* class constructor
- [~IpoptSolver\(\)](#)

the [IpoptSolver](#) class destructor

- virtual void [solve](#) () throw (ErrorClass)

solve results in an instance being read into the Ipopt data structures and optimize

- virtual void [buildSolverInstance](#) () throw (ErrorClass)

The implementation of the virtual functions.

- virtual void [setSolverOptions](#) () throw (ErrorClass)

The implementation of the virtual functions.

- void [dataEchoCheck](#) ()

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

Public Attributes

- Ipopt::SmartPtr< Ipopt::TNLP > [nlp](#)

- Ipopt::SmartPtr

< Ipopt::IpoptApplication > [app](#)

- [OSiLReader](#) * [m_osilreader](#)

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

- [OSoLReader](#) * [m_osolreader](#)

m_osolreader is an [OSoLReader](#) object used to create an ooption from an osol string if needed

6.67.1 Detailed Description

The [IpoptSolver](#) class solves problems using Ipopt.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

this class takes an OSiL instance and optimizes it using the COIN-OR Ipopt solver

Definition at line 167 of file OSIpoptSolver.h.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 IpoptSolver::IpoptSolver ()

the [IpoptSolver](#) class constructor

6.67.2.2 IpoptSolver::~~IpoptSolver ()

the [IpoptSolver](#) class destructor

6.67.3 Member Function Documentation

6.67.3.1 virtual void IpoptSolver::solve () throw (ErrorClass) [virtual]

solve results in an instance being read into the Ipopt data structures and optimize

Implements [DefaultSolver](#).

6.67.3.2 void IpoptSolver::buildSolverInstance () throw (ErrorClass) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.67.3.3 void IpoptSolver::setSolverOptions () throw (ErrorClass) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.67.3.4 void IpoptSolver::dataEchoCheck ()

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

6.67.4 Member Data Documentation

6.67.4.1 Ipopt::SmartPtr<Ipopt::TNLP> IpoptSolver::nlp

Definition at line 177 of file OSIpoptSolver.h.

6.67.4.2 Ipopt::SmartPtr<Ipopt::IpoptApplication> IpoptSolver::app

Definition at line 179 of file OSIpoptSolver.h.

6.67.4.3 OSiLReader* IpoptSolver::m_osilreader

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 209 of file OSIpoptSolver.h.

6.67.4.4 OSoLReader* IpoptSolver::m_osolreader

m_osolreader is an [OSoLReader](#) object used to create an osoption from an osol string if needed

Definition at line 215 of file OSIpoptSolver.h.

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

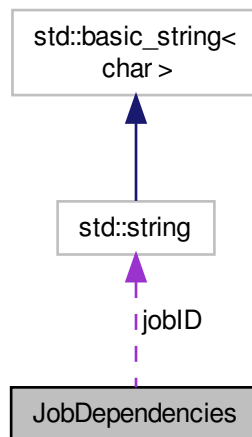
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/[OSIpoptSolver.h](#)

6.68 JobDependencies Class Reference

the [JobDependencies](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for JobDependencies:



Public Member Functions

- [JobDependencies](#) ()
Default constructor.
- [~JobDependencies](#) ()
Class destructor.
- bool [isEqual](#) ([JobDependencies](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([JobDependencies](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setJobID](#) (int [numberOfJobIDs](#), std::string *jobID)
A function to set an array of <jobID> elements.
- bool [addJobID](#) (std::string jobID)
A function to add an <jobID> element.

Public Attributes

- int [numberOfJobIDs](#)
the number of entries in the list of job dependencies

- `std::string * jobID`
the list of job IDs

6.68.1 Detailed Description

the [JobDependencies](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 709 of file OSOption.h.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 JobDependencies::JobDependencies ()

Default constructor.

6.68.2.2 JobDependencies::~~JobDependencies ()

Class destructor.

6.68.3 Member Function Documentation

6.68.3.1 bool JobDependencies::isEqual (JobDependencies * *that*)

A function to check for the equality of two objects.

6.68.3.2 bool JobDependencies::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.68.3.3 `bool JobDependencies::deepCopyFrom (JobDependencies * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.68.3.4 `bool JobDependencies::setJobID (int numberOfJobIDs, std::string * jobID)`

A function to set an array of <jobID> elements.

Parameters

<i>numberOfJobIDs,:</i>	number of <jobID> elements to be set
<i>jobID,:</i>	the array of <jobID> elements that are to be set

6.68.3.5 `bool JobDependencies::addJobID (std::string jobID)`

A function to add an <jobID> element.

Parameters

<i>jobID,:</i>	the name of the <jobID> element to be added
----------------	---

6.68.4 Member Data Documentation**6.68.4.1** `int JobDependencies::numberOfJobIDs`

the number of entries in the list of job dependencies

Definition at line 714 of file OSOption.h.

6.68.4.2 `std::string* JobDependencies::jobID`

the list of job IDs

Definition at line 717 of file OSOption.h.

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

- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.69 JobOption Class Reference

the [JobOption](#) class.

```
#include <OSOption.h>
```


The diagram illustrates the dependencies of various Boost.Job types on standard C++ types. The nodes and their dependencies are as follows:

- std::basic_string<char>** depends on **std::string** (via `std::basic_string<char>`).
- std::string** depends on **JobDependencies** (via `jobID`), **TimeSpan** (via `unit`), **Processes** (via `process`), **DirectoriesAndFiles** (via `path`), **PathPair** (via `from`), **PathPairs** (via `to`), **OtherOption** (via `value`), and **OtherOptions** (via `name`).
- JobDependencies** depends on **JobOption** (via `dependencies`).
- TimeSpan** depends on **JobOption** (via `maxTime`).
- Processes** depends on **JobOption** (via `processesToKill`).
- DirectoriesAndFiles** depends on **JobOption** (via `requiredFiles`, `filesToDelete`, `directoriesToDelete`, `filesToMake`, and `directoriesToMake`).
- PathPair** depends on **JobOption** (via `inputDirectoriesToMove`, `outputDirectoriesToMove`, `inputFilesToMove`, and `outputFilesToMove`).
- PathPairs** depends on **JobOption** (via `inputFilesToMove` and `outputFilesToMove`).
- OtherOption** depends on **JobOption** (via `otherOptions`).
- OtherOptions** depends on **JobOption** (via `otherOptions`).

- `JobOption ()`
Default constructor.
- `~JobOption ()`
Class destructor.
- `bool isEqual (JobOption *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (JobOption *that)`
A function to make a deep copy of an instance of this class.

- `TimeSpan * maxSpan`
the maximum time allowed
- `std::string requestedStartTime`
the requested time to start the job
- `JobDependencies * dependencies`
the dependency set
- `DirectoriesAndFiles * requiredDirectories`
directories required to run the job
- `DirectoriesAndFiles * requiredFiles`
files required to run the job
- `DirectoriesAndFiles * directoriesToMake`
directories to make during the job
- `DirectoriesAndFiles * filesToMake`
files to make during the job
- `PathPairs * inputDirectoriesToMove`
input directories to move or copy

- [PathPairs](#) * [inputFilesToMove](#)
input files to move or copy
- [PathPairs](#) * [outputFilesToMove](#)
output files to move or copy
- [PathPairs](#) * [outputDirectoriesToMove](#)
output directories to move or copy
- [DirectoriesAndFiles](#) * [filesToDelete](#)
files to delete upon completion
- [DirectoriesAndFiles](#) * [directoriesToDelete](#)
directories to delete upon completion
- [Processes](#) * [processesToKill](#)
processes to kill upon completion
- [OtherOptions](#) * [otherOptions](#)
list of other job options

6.69.1 Detailed Description

the [JobOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1064 of file OSOption.h.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 JobOption::JobOption ()

Default constructor.

6.69.2.2 JobOption::~~JobOption ()

Class destructor.

6.69.3 Member Function Documentation

6.69.3.1 bool JobOption::IsEqual (JobOption * *that*)

A function to check for the equality of two objects.

6.69.3.2 bool JobOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.69.3.3 bool JobOption::deepCopyFrom (JobOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.69.4 Member Data Documentation**6.69.4.1 TimeSpan* JobOption::maxTime**

the maximum time allowed

Definition at line 1069 of file OSOption.h.

6.69.4.2 std::string JobOption::requestedStartTime

the requested time to start the job

Definition at line 1072 of file OSOption.h.

6.69.4.3 JobDependencies* JobOption::dependencies

the dependency set

Definition at line 1075 of file OSOption.h.

6.69.4.4 DirectoriesAndFiles* JobOption::requiredDirectories

directories required to run the job

Definition at line 1078 of file OSOption.h.

6.69.4.5 DirectoriesAndFiles* JobOption::requiredFiles

files required to run the job

Definition at line 1081 of file OSOption.h.

6.69.4.6 DirectoriesAndFiles* JobOption::directoriesToMake

directories to make during the job

Definition at line 1084 of file OSOption.h.

6.69.4.7 DirectoriesAndFiles* JobOption::filesToMake

files to make during the job

Definition at line 1087 of file OOption.h.

6.69.4.8 PathPairs* JobOption::inputDirectoriesToMove

input directories to move or copy

Definition at line 1090 of file OOption.h.

6.69.4.9 PathPairs* JobOption::inputFilesToMove

input files to move or copy

Definition at line 1093 of file OOption.h.

6.69.4.10 PathPairs* JobOption::outputFilesToMove

output files to move or copy

Definition at line 1096 of file OOption.h.

6.69.4.11 PathPairs* JobOption::outputDirectoriesToMove

output directories to move or copy

Definition at line 1099 of file OOption.h.

6.69.4.12 DirectoriesAndFiles* JobOption::filesToDelete

files to delete upon completion

Definition at line 1102 of file OOption.h.

6.69.4.13 DirectoriesAndFiles* JobOption::directoriesToDelete

directories to delete upon completion

Definition at line 1105 of file OOption.h.

6.69.4.14 Processes* JobOption::processesToKill

processes to kill upon completion

Definition at line 1108 of file OOption.h.

6.69.4.15 OtherOptions* JobOption::otherOptions

list of other job options

Definition at line 1111 of file OOption.h.

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

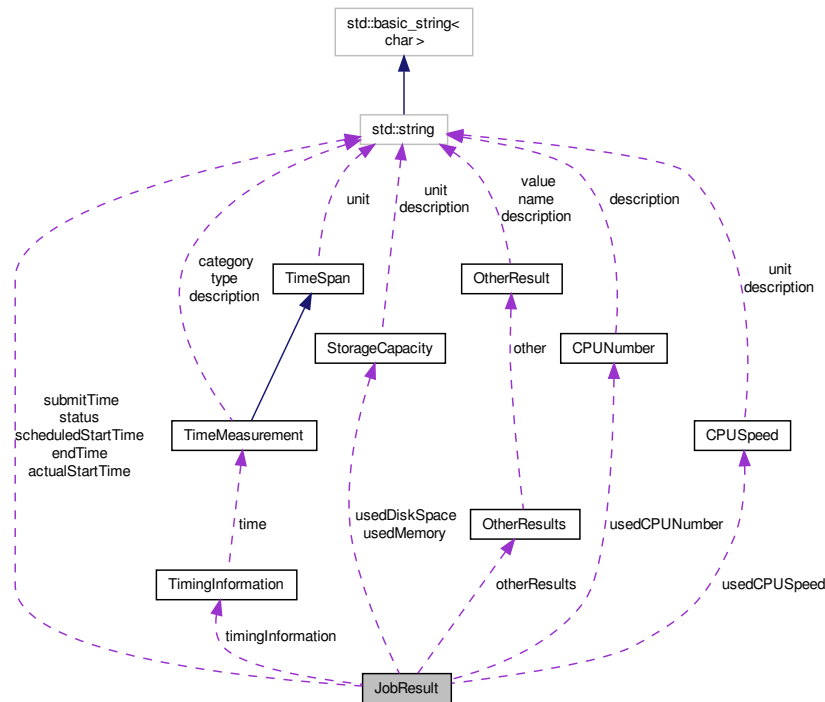
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OOption.h>

6.70 JobResult Class Reference

The [JobResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for JobResult:



Public Member Functions

- **JobResult** ()
Default constructor.
- **~JobResult** ()
Class destructor.
- bool **isEqual** (**JobResult** *that)
A function to check for the equality of two objects.
- bool **setRandom** (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string **status**
job status
- std::string **submitTime**
time the job was submitted
- std::string **scheduledStartTime**
the time when the job was supposed to start
- std::string **actualStartTime**

- the time when the job actually started*
- `std::string endTime`
the time when the job finished
- `TimingInformation * timingInformation`
a pointer to the [TimingInformation](#) class
- `StorageCapacity * usedDiskSpace`
a pointer to the [DiskSpace](#) class
- `StorageCapacity * usedMemory`
a pointer to the [MemorySize](#) class
- `CPU Speed * usedCPU Speed`
a pointer to the [CPU Speed](#) class
- `CPU Number * usedCPU Number`
a pointer to the [CPU Number](#) class
- `OtherResults * otherResults`
a pointer to the [OtherResults](#) class

6.70.1 Detailed Description

The [JobResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the system information that is defined in the OSrL schema.

Definition at line 658 of file OSResult.h.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 `JobResult::JobResult ()`

Default constructor.

6.70.2.2 `JobResult::~~JobResult ()`

Class destructor.

6.70.3 Member Function Documentation

6.70.3.1 `bool JobResult::isEqual (JobResult * that)`

A function to check for the equality of two objects.

6.70.3.2 `bool JobResult::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.70.4 Member Data Documentation

6.70.4.1 `std::string JobResult::status`

job status

Definition at line 663 of file OSResult.h.

6.70.4.2 `std::string JobResult::submitTime`

time the job was submitted

Definition at line 666 of file OSResult.h.

6.70.4.3 `std::string JobResult::scheduledStartTime`

the time when the job was supposed to start

Definition at line 669 of file OSResult.h.

6.70.4.4 `std::string JobResult::actualStartTime`

the time when the job actually started

Definition at line 672 of file OSResult.h.

6.70.4.5 `std::string JobResult::endTime`

the time when the job finished

Definition at line 675 of file OSResult.h.

6.70.4.6 `TimingInformation* JobResult::timingInformation`

a pointer to the [TimingInformation](#) class

Definition at line 678 of file OSResult.h.

6.70.4.7 `StorageCapacity* JobResult::usedDiskSpace`

a pointer to the [DiskSpace](#) class

Definition at line 682 of file OSResult.h.

6.70.4.8 StorageCapacity* JobResult::usedMemory

a pointer to the MemorySize class

Definition at line 686 of file OSResult.h.

6.70.4.9 CPUSpeed* JobResult::usedCPUSpeed

a pointer to the [CPUSpeed](#) class

Definition at line 690 of file OSResult.h.

6.70.4.10 CPUNumber* JobResult::usedCPUNumber

a pointer to the [CPUNumber](#) class

Definition at line 694 of file OSResult.h.

6.70.4.11 OtherResults* JobResult::otherResults

a pointer to the [OtherResults](#) class

Definition at line 698 of file OSResult.h.

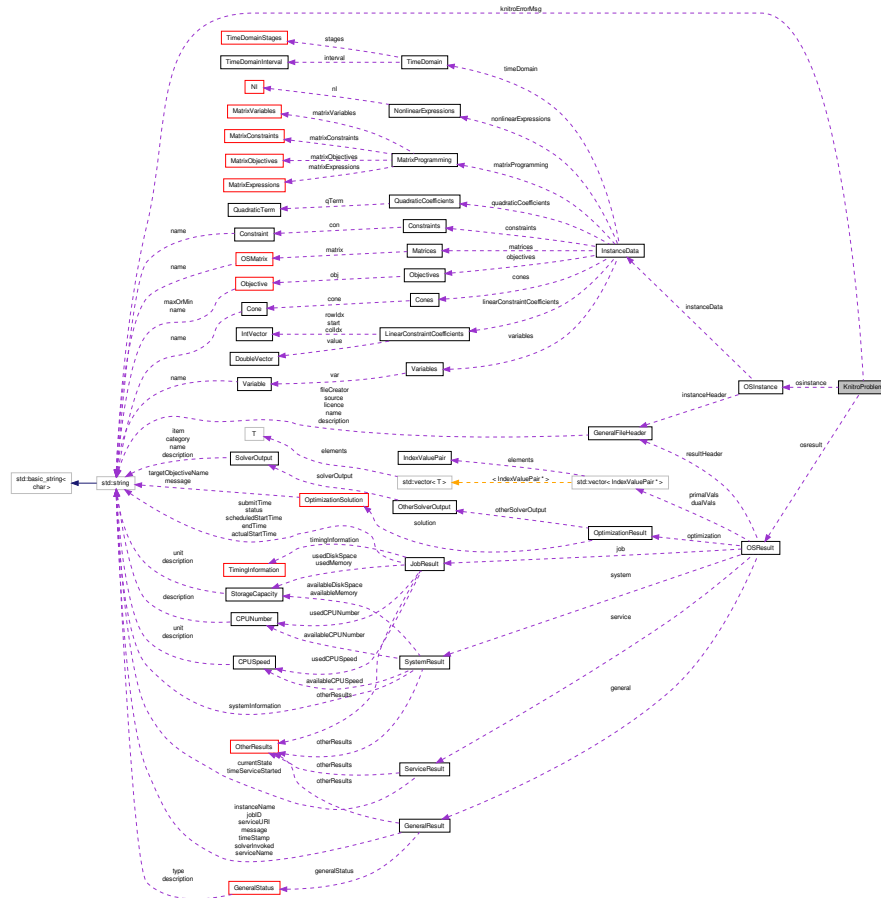
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.71 KnitroProblem Class Reference

```
#include <OSKnitroSolver.h>
```


Collaboration diagram for KnitroProblem:



Public Member Functions

- [KnitroProblem](#) ([OSInstance](#) *osinstance_, [OSResult](#) *osresult_)
the *IpoptProblem* class constructor
- virtual [~KnitroProblem](#) ()
the *IpoptProblem* class destructor
- int [getN](#) (void)
- int [getM](#) (void)
- void [getInitialX](#) (double *const daX)
- bool [loadProblemIntoKnitro](#) (KTR_context_ptr kc)
- bool [areDerivativesImplemented](#) (const DerivativesImplementedType nWhichDers)
- int [evalFC](#) (const double *const daX, double *const dObj, double *const daC, void *userParams)
- int [evalGA](#) (const double *const daX, double *const daG, double *const daJ, void *userParams)
- int [evalH](#) (const double *const daX, const double *const daLambda, double *const daH, void *userParams)
- int [evalHV](#) (const double *const daX, const double *const daLambda, double *const daHV, void *userParams)

Public Attributes

- [OSResult](#) * *osresult*
- [OSInstance](#) * *osinstance*
- std::string *knitroErrorMsg*

6.71.1 Detailed Description

Definition at line 87 of file OSKnitroSolver.h.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 KnitroProblem::KnitroProblem (OSInstance * *osinstance*_, OSResult * *osresult*_)

the IpoptProblemclass constructor

6.71.2.2 virtual KnitroProblem::~KnitroProblem () [virtual]

the [IpoptProblem](#) class destructor

6.71.3 Member Function Documentation

6.71.3.1 int KnitroProblem::getN (void)

6.71.3.2 int KnitroProblem::getM (void)

6.71.3.3 void KnitroProblem::getInitialX (double *const *daX*)6.71.3.4 bool KnitroProblem::loadProblemIntoKnitro (KTR_context_ptr *kc*)6.71.3.5 bool KnitroProblem::areDerivativesImplemented (const DerivativesImplementedType *nWhichDers*)6.71.3.6 int KnitroProblem::evalFC (const double *const *daX*, double *const *dObj*, double *const *daC*, void * *userParams*)6.71.3.7 int KnitroProblem::evalGA (const double *const *daX*, double *const *daG*, double *const *daJ*, void * *userParams*)6.71.3.8 int KnitroProblem::evalH (const double *const *daX*, const double *const *daLambda*, double *const *daH*, void * *userParams*)6.71.3.9 int KnitroProblem::evalHV (const double *const *daX*, const double *const *daLambda*, double *const *daHV*, void * *userParams*)

6.71.4 Member Data Documentation

6.71.4.1 OSResult* KnitroProblem::osresult

Definition at line 97 of file OSKnitroSolver.h.

6.71.4.2 OSInstance* KnitroProblem::osinstance

Definition at line 99 of file OSKnitroSolver.h.

6.71.4.3 `std::string KnitroProblem::knitroErrorMsg`

Definition at line 128 of file `OSKnitroSolver.h`.

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

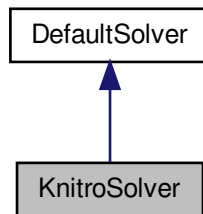
- `/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSKnitroSolver.h`

6.72 KnitroSolver Class Reference

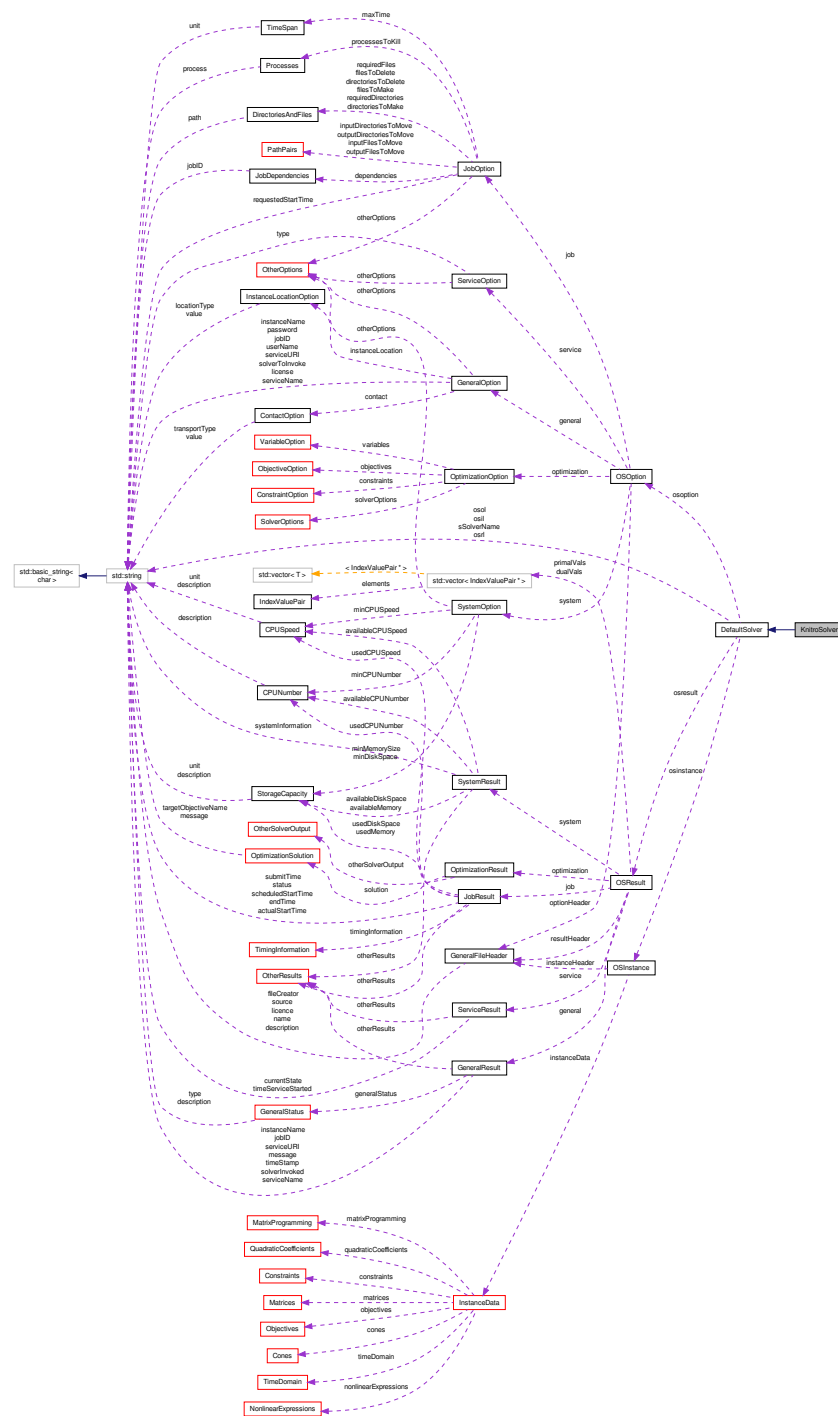
the [KnitroSolver](#) class solves problems using Knitro.

```
#include <OSKnitroSolver.h>
```

Inheritance diagram for KnitroSolver:



Collaboration diagram for KnitroSolver:



Public Member Functions

- [KnitroSolver](#) ()
the *KnitroSolver* class constructor

- [~KnitroSolver](#) ()
the [KnitroSolver](#) class constructor
- virtual void [buildSolverInstance](#) () throw (ErrorClass)
buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API
- virtual void [setSolverOptions](#) () throw (ErrorClass)
The implementation of the virtual functions.
- virtual void [solve](#) () throw (ErrorClass)
solve results in an instance being read into the Knitro data structures and optimized
- void [dataEchoCheck](#) ()
use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

Additional Inherited Members

6.72.1 Detailed Description

the [KnitroSolver](#) class solves problems using Knitro.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

this class takes an OSiL instance and optimizes it using the Knitro solver

Definition at line 144 of file OSKnitroSolver.h.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 [KnitroSolver::KnitroSolver](#) ()

the [KnitroSolver](#) class constructor

6.72.2.2 [KnitroSolver::~~KnitroSolver](#) ()

the [KnitroSolver](#) class constructor

6.72.3 Member Function Documentation

6.72.3.1 virtual void [KnitroSolver::buildSolverInstance](#) () throw (ErrorClass) [virtual]

[buildSolverInstance](#) is a virtual function – the actual solvers will implement their own [buildSolverInstance](#) method – the solver instance is the instance the individual solver sees in its API

Implements [DefaultSolver](#).

6.72.3.2 `void KnitroSolver::setSolverOptions () throw (ErrorClass) [virtual]`

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.72.3.3 `virtual void KnitroSolver::solve () throw (ErrorClass) [virtual]`

solve results in an instance being read into the Knitro data structures and optimized

Implements [DefaultSolver](#).

6.72.3.4 `void KnitroSolver::dataEchoCheck ()`

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

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

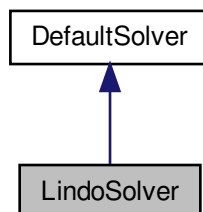
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSKnitroSolver.h](#)

6.73 LindoSolver Class Reference

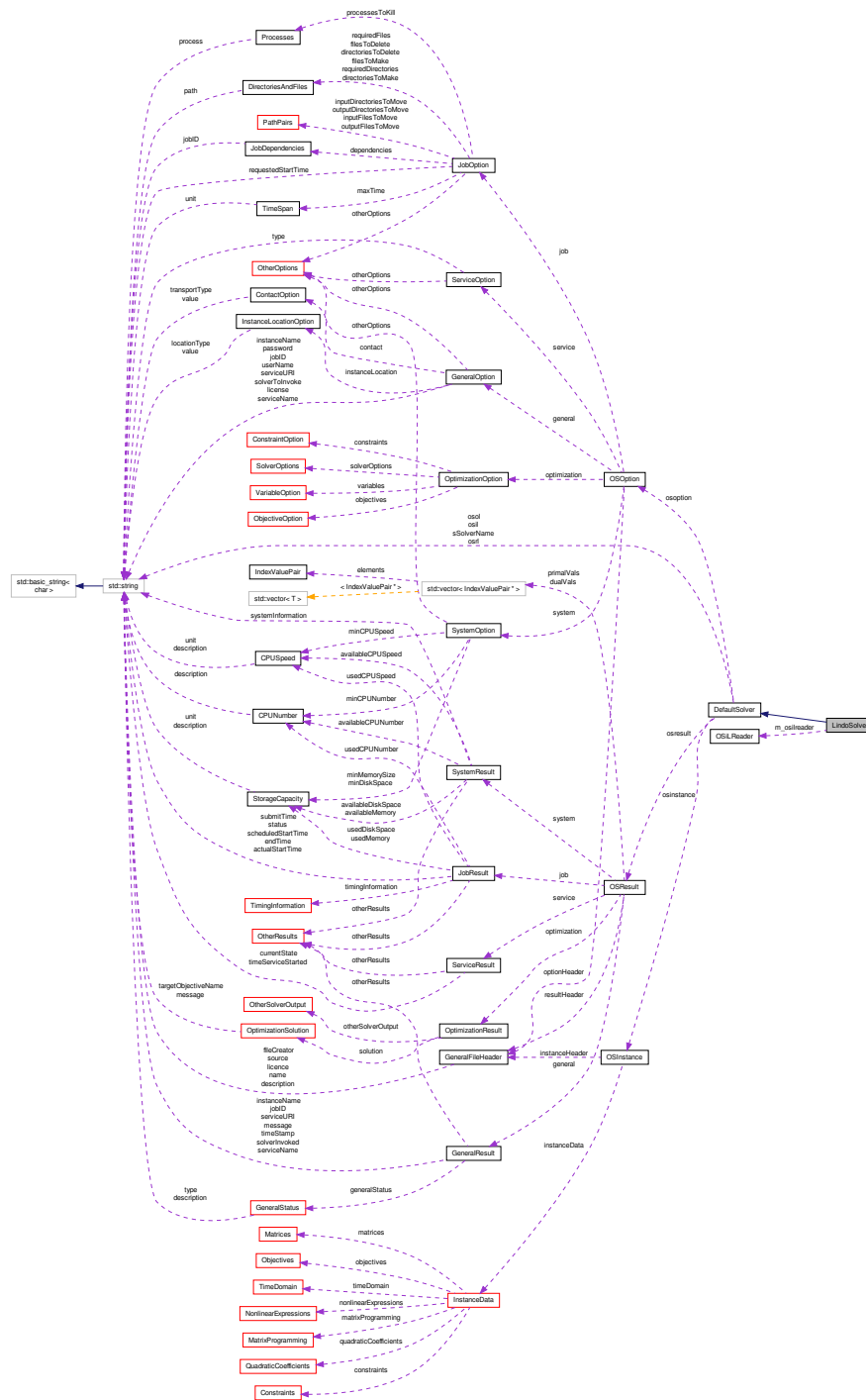
the [LindoSolver](#) class solves problems using Lindo.

```
#include <OSLindoSolver.h>
```

Inheritance diagram for LindoSolver:



Collaboration diagram for LingoSolver:



Public Member Functions

- [LingoSolver\(\)](#)
the *LingoSolver* class constructor

- [~LindoSolver \(\)](#)
the [LindoSolver](#) class destructor
- virtual void [solve \(\)](#)
solve results in an instance being read into the Lindo data structures and optimized
- virtual void [buildSolverInstance \(\)](#) throw (ErrorClass)
buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API
- virtual void [setSolverOptions \(\)](#) throw (ErrorClass)
The implementation of the virtual functions.
- bool [optimize \(\)](#)
invoke the Lindo API solver
- bool [processVariables \(\)](#)
read the OSiL instance variables and put these into the LINDO API variables
- bool [processConstraints \(\)](#)
read the OSiL instance constraints and put these into the LINDO API constraints
- bool [generateLindoModel \(\)](#)
create the LINDO environment and read the problem into the internal LINDO data structures
- bool [addSlackVars \(\)](#)
LINDO does not handle constraints with upper and lower bounds this method is part of kludge where we add a new variable to handle the bounds.
- bool [processQuadraticTerms \(\)](#)
read the quadratic terms in the model
- bool [processNonlinearExpressions \(\)](#)
read the nonlinear terms in the model
- void [dataEchoCheck \(\)](#)
use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

Public Attributes

- [OSiLReader * m_osilreader](#)
m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Protected Member Functions

- void [lindoAPIErrorCheck](#) (std::string errmsg)
Lindo's generalized error Reporting function.

6.73.1 Detailed Description

the [LindoSolver](#) class solves problems using Lindo.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

this class takes an OSiL instance and optimizes it using the Lindo API

Definition at line 49 of file OSLindoSolver.h.

6.73.2 Constructor & Destructor Documentation**6.73.2.1 LindoSolver::LindoSolver ()**

the [LindoSolver](#) class constructor

6.73.2.2 LindoSolver::~~LindoSolver ()

the [LindoSolver](#) class destructor

6.73.3 Member Function Documentation**6.73.3.1 virtual void LindoSolver::solve () [virtual]**

solve results in an instance being read into the Lindo data structures and optimized

Implements [DefaultSolver](#).

6.73.3.2 virtual void LindoSolver::buildSolverInstance () throw (ErrorClass) [virtual]

buildSolverInstance is a virtual function – the actual solvers will implement their own buildSolverInstance method – the solver instance is the instance the individual solver sees in its API

Implements [DefaultSolver](#).

6.73.3.3 void LindoSolver::setSolverOptions () throw (ErrorClass) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

6.73.3.4 bool LindoSolver::optimize ()

invoke the Lindo API solver

Returns

true if an exception is not thrown.

6.73.3.5 `bool LindoSolver::processVariables ()`

read the OSiL instance variables and put these into the LINDO API variables

Returns

true if an exception is not thrown.

6.73.3.6 `bool LindoSolver::processConstraints ()`

read the OSiL instance constraints and put these into the LINDO API constraints

Returns

true if an exception is not thrown.

6.73.3.7 `bool LindoSolver::generateLindoModel ()`

create the LINDO environment and read the problem into the internal LINDO data structures

Returns

true if an exception is not thrown.

6.73.3.8 `bool LindoSolver::addSlackVars ()`

LINDO does not handle constraints with upper and lower bounds this method is part of kludge where we add a new variable to handle the bounds.

Returns

true if an exception is not thrown.

6.73.3.9 `bool LindoSolver::processQuadraticTerms ()`

read the quadratic terms in the model

Returns

true if an exception is not thrown.

6.73.3.10 `bool LindoSolver::processNonlinearExpressions ()`

read the nonlinear terms in the model

Returns

true if an exception is not thrown.

6.73.3.11 `void LindoSolver::dataEchoCheck ()`

use this for debugging, print out the instance that the solver thinks it has and compare this with the OSiL file

6.73.3.12 `void LindoSolver::lindoAPIErrorCheck (std::string errmsg)` `[protected]`

Lindo's generalized error Reporting function.

6.73.4 Member Data Documentation

6.73.4.1 OSiLReader* LindoSolver::m_osilreader

m_osilreader is an [OSiLReader](#) object used to create an osinstance from an osil string if needed

Definition at line 129 of file OSLindoSolver.h.

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

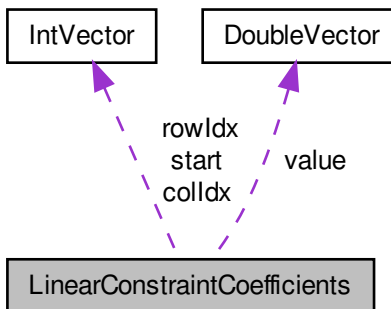
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/[OSLindoSolver.h](#)

6.74 LinearConstraintCoefficients Class Reference

The in-memory representation of the `<linearConstraintCoefficients>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for LinearConstraintCoefficients:



Public Member Functions

- [LinearConstraintCoefficients](#) ()
The [LinearConstraintCoefficients](#) class constructor.
- [~LinearConstraintCoefficients](#) ()
The [LinearConstraintCoefficients](#) class destructor.
- bool [isEqual](#) ([LinearConstraintCoefficients](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfValues](#)
numberOfValues is the number of nonzero elements stored in the `<linearConstraintCoefficients>` element
- [IntVector](#) * [start](#)

- a pointer to the start of each row or column stored in sparse format*
- [IntVector](#) * [rowIdx](#)
a pointer of row indices if the problem is stored by column
- [IntVector](#) * [colIdx](#)
a pointer of column indices if the problem is stored by row
- [DoubleVector](#) * [value](#)
a pointer to the array of nonzero values being stored
- int [iNumberOfStartElements](#)
iNumberOfStartElements counts the number of elements in the <start> section of <linearConstraintCoefficients>.

6.74.1 Detailed Description

The in-memory representation of the `<linearConstraintCoefficients>` element.

Remarks

if a large part of the problem is linear, then store this is the standard sparse format, either by column or row. There are three arrays, an array of nonzero values, an array of either column or row indices and then a pointer to the start of each column or row.

Definition at line 288 of file OSInstance.h.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 LinearConstraintCoefficients::LinearConstraintCoefficients ()

The [LinearConstraintCoefficients](#) class constructor.

6.74.2.2 LinearConstraintCoefficients::~~LinearConstraintCoefficients ()

The [LinearConstraintCoefficients](#) class destructor.

6.74.3 Member Function Documentation

6.74.3.1 bool LinearConstraintCoefficients::isEqual (LinearConstraintCoefficients * *that*)

A function to check for the equality of two objects.

6.74.4 Member Data Documentation

6.74.4.1 int LinearConstraintCoefficients::numberOfValues

numberOfValues is the number of nonzero elements stored in the `<linearConstraintCoefficients>` element

Definition at line 301 of file OSInstance.h.

6.74.4.2 IntVector* LinearConstraintCoefficients::start

a pointer to the start of each row or column stored in sparse format

Definition at line 306 of file OSInstance.h.

6.74.4.3 IntVector* LinearConstraintCoefficients::rowIdx

a pointer of row indices if the problem is stored by column

Definition at line 309 of file OSInstance.h.

6.74.4.4 IntVector* LinearConstraintCoefficients::colIdx

a pointer of column indices if the problem is stored by row

Definition at line 312 of file OSInstance.h.

6.74.4.5 DoubleVector* LinearConstraintCoefficients::value

a pointer to the array of nonzero values being stored

Definition at line 315 of file OSInstance.h.

6.74.4.6 int LinearConstraintCoefficients::iNumberOfStartElements

iNumberOfStartElements counts the number of elements in the <start> section of <linearConstraintCoefficients>.

This is useful for the parser in checking consistency of the number of start elements with variables or rows

Definition at line 322 of file OSInstance.h.

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

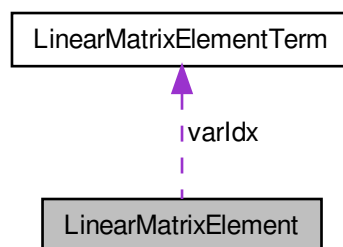
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.75 LinearMatrixElement Class Reference

a data structure to represent an expression in a linearMatrix element A [LinearMatrixElement](#) is a (finite) sum of LinearMatrixElementTerms, with an optional additive constant

```
#include <OSMatrix.h>
```

Collaboration diagram for LinearMatrixElement:

**Public Member Functions**

- [LinearMatrixElement](#) ()

- `~LinearMatrixElement ()`
- `bool IsEqual (LinearMatrixElement *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant, int iMin, int iMax)`
A function to make a random instance of this class.
- `bool deepCopyFrom (LinearMatrixElement *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int numberOfVarIdx`
- `double constant`
- `LinearMatrixElementTerm ** varIdx`

6.75.1 Detailed Description

a data structure to represent an expression in a linearMatrix element A `LinearMatrixElement` is a (finite) sum of `LinearMatrixElementTerms`, with an optional additive constant

Parameters

<code>numberOfVarIdx</code>	gives the number of terms in the expression
-----------------------------	---

Definition at line 411 of file `OSMatrix.h`.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 `LinearMatrixElement::LinearMatrixElement ()`

6.75.2.2 `LinearMatrixElement::~~LinearMatrixElement ()`

6.75.3 Member Function Documentation

6.75.3.1 `bool LinearMatrixElement::IsEqual (LinearMatrixElement * that)`

A function to check for the equality of two objects.

6.75.3.2 `bool LinearMatrixElement::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<code>density,:</code>	corresponds to the probability that a particular child element is created
<code>conformant,:</code>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<code>iMin,:</code>	lowest index value (inclusive) that a variable reference in this matrix can take
<code>iMax,:</code>	greatest index value (inclusive) that a variable reference in this matrix can take

6.75.3.3 `bool LinearMatrixElement::deepCopyFrom (LinearMatrixElement * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.75.4 Member Data Documentation

6.75.4.1 int LinearMatrixElement::numberOfVarIdx

Definition at line 414 of file OSMatrix.h.

6.75.4.2 double LinearMatrixElement::constant

Definition at line 415 of file OSMatrix.h.

6.75.4.3 LinearMatrixElementTerm** LinearMatrixElement::varIdx

Definition at line 417 of file OSMatrix.h.

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

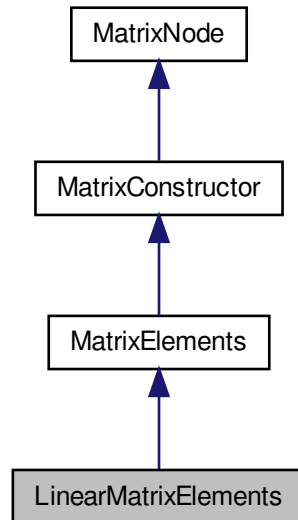
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSMatrix.h](#)

6.76 LinearMatrixElements Class Reference

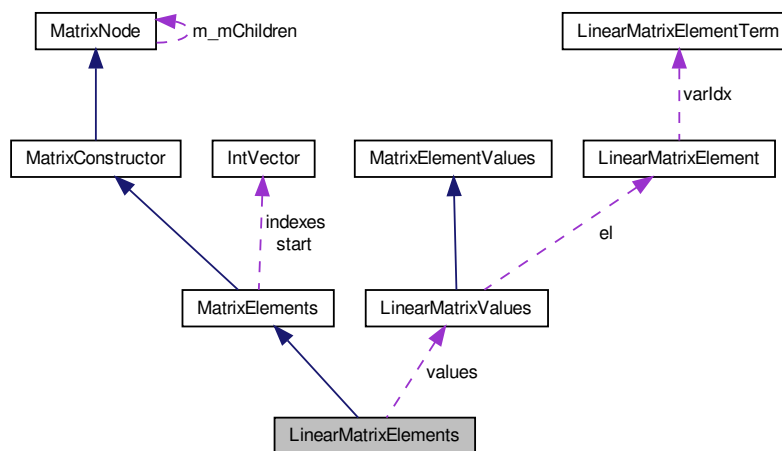
a data structure to represent the nonzero values in a linearMatrix element

```
#include <OSMatrix.h>
```

Inheritance diagram for LinearMatrixElements:



Collaboration diagram for LinearMatrixElements:



Public Member Functions

- [LinearMatrixElements\(\)](#)
- [~LinearMatrixElements\(\)](#)

- virtual `ENUM_MATRIX_CONSTRUCTOR_TYPE` `getNodeType ()`
- virtual `ENUM_MATRIX_TYPE` `getMatrixType ()`
- virtual `std::string` `getNodeName ()`
- virtual `std::string` `getMatrixNodeInXML ()`
- virtual `bool` `alignsOnBlockBoundary` (`int` firstRow, `int` firstColumn, `int` nRows, `int` nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual `LinearMatrixElements *` `cloneMatrixNode ()`
- `bool` `isEqual` (`LinearMatrixElements *`that)
A function to check for the equality of two objects.
- `bool` `setRandom` (`double` density, `bool` conformant, `int` iMin, `int` iMax)
A function to make a random instance of this class.
- `bool` `deepCopyFrom` (`LinearMatrixElements *`that)
A function to make a deep copy of an instance of this class.

Public Attributes

- `LinearMatrixValues *` `values`
*The values are expressions of the form $a_0 + a_1 x_{\{i_1\}} * a_2 x_{\{i_2\}} + \dots$*

6.76.1 Detailed Description

a data structure to represent the nonzero values in a linearMatrix element

Definition at line 918 of file OSMatrix.h.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 `LinearMatrixElements::LinearMatrixElements ()`

6.76.2.2 `LinearMatrixElements::~~LinearMatrixElements ()`

6.76.3 Member Function Documentation

6.76.3.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE` `LinearMatrixElements::getNodeType ()` `[virtual]`

Returns

the value of nType

Reimplemented from `MatrixNode`.

6.76.3.2 `virtual ENUM_MATRIX_TYPE` `LinearMatrixElements::getMatrixType ()` `[virtual]`

Returns

the type of the matrix elements

Implements `MatrixNode`.

6.76.3.3 `virtual std::string LinearMatrixElements::getNodeName () [virtual]`

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.76.3.4 `virtual std::string LinearMatrixElements::getMatrixNodeInXML () [virtual]`

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.76.3.5 `virtual bool LinearMatrixElements::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]`

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.76.3.6 `virtual LinearMatrixElements* LinearMatrixElements::cloneMatrixNode () [virtual]`

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.76.3.7 `bool LinearMatrixElements::isEqual (LinearMatrixElements * that)`

A function to check for the equality of two objects.

6.76.3.8 `bool LinearMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.76.3.9 bool LinearMatrixElements::deepCopyFrom (LinearMatrixElements * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.76.4 Member Data Documentation

6.76.4.1 LinearMatrixValues* LinearMatrixElements::values

The values are expressions of the form $a_0 + a_1 x_{i_1} * a_2 x_{i_2} + \dots$

Each term in this sum is stored as a separate [LinearMatrixValues](#) object

Definition at line 926 of file OSMatrix.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.77 LinearMatrixElementTerm Class Reference

a data structure to represent a term in a linearMatrix element A term has the form $c * x_{\{k\}}$, where c defaults to 1 and k is a valid index for a variable This is essentially an index-value pair, but with the presence of a default value

```
#include <OSMatrix.h>
```

Public Member Functions

- [LinearMatrixElementTerm \(\)](#)
- [~LinearMatrixElementTerm \(\)](#)
- bool [IsEqual \(LinearMatrixElementTerm *that\)](#)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom \(LinearMatrixElementTerm *that\)](#)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
- double [coef](#)

6.77.1 Detailed Description

a data structure to represent a term in a linearMatrix element A term has the form $c \cdot x_{\{k\}}$, where c defaults to 1 and k is a valid index for a variable This is essentially an index-value pair, but with the presence of a default value

Definition at line 373 of file OSMatrix.h.

6.77.2 Constructor & Destructor Documentation

6.77.2.1 LinearMatrixElementTerm::LinearMatrixElementTerm ()

6.77.2.2 LinearMatrixElementTerm::~~LinearMatrixElementTerm ()

6.77.3 Member Function Documentation

6.77.3.1 bool LinearMatrixElementTerm::isEqual (LinearMatrixElementTerm * *that*)

A function to check for the equality of two objects.

6.77.3.2 bool LinearMatrixElementTerm::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.77.3.3 bool LinearMatrixElementTerm::deepCopyFrom (LinearMatrixElementTerm * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.77.4 Member Data Documentation

6.77.4.1 int LinearMatrixElementTerm::idx

Definition at line 376 of file OSMatrix.h.

6.77.4.2 double LinearMatrixElementTerm::coef

Definition at line 377 of file OSMatrix.h.

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

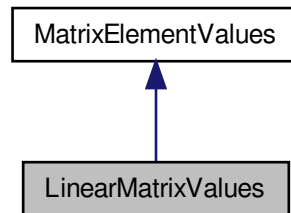
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.78 LinearMatrixValues Class Reference

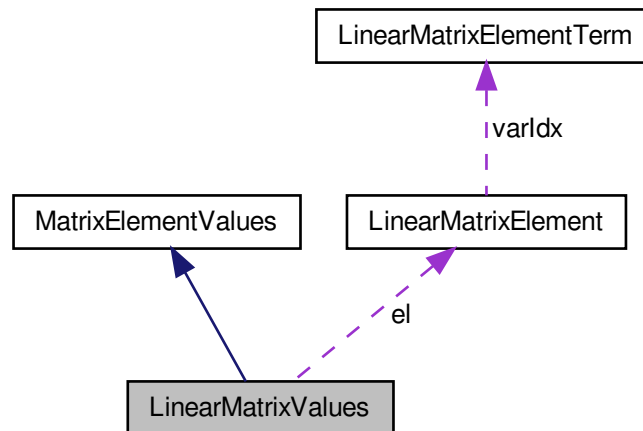
a data structure to represent the linear expressions in a [LinearMatrixElement](#) object

```
#include <OSMatrix.h>
```

Inheritance diagram for LinearMatrixValues:



Collaboration diagram for LinearMatrixValues:



Public Member Functions

- [LinearMatrixValues](#) ()
- [~LinearMatrixValues](#) ()
- `bool` [IsEqual](#) ([LinearMatrixValues](#) *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double *density*, bool *conformant*, int *iMin*, int *iMax*)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([LinearMatrixValues](#) **that*)
A function to make a deep copy of an instance of this class.

Public Attributes

- [LinearMatrixElement](#) ** *el*

6.78.1 Detailed Description

a data structure to represent the linear expressions in a [LinearMatrixElement](#) object
Definition at line 603 of file OSMatrix.h.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 [LinearMatrixValues::LinearMatrixValues](#) ()

6.78.2.2 [LinearMatrixValues::~~LinearMatrixValues](#) ()

6.78.3 Member Function Documentation

6.78.3.1 bool [LinearMatrixValues::isEqual](#) ([LinearMatrixValues](#) * *that*)

A function to check for the equality of two objects.

6.78.3.2 bool [LinearMatrixValues::setRandom](#) (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.78.3.3 bool [LinearMatrixValues::deepCopyFrom](#) ([LinearMatrixValues](#) * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.78.4 Member Data Documentation

6.78.4.1 LinearMatrixElement** LinearMatrixValues::el

Definition at line 606 of file OSMatrix.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.79 MathUtil Class Reference

this class has routines for linear algebra.

```
#include <OSMathUtil.h>
```

Public Member Functions

- [MathUtil](#) ()
the class constructor
- [~MathUtil](#) ()
the class destructor
- std::string [format_os_dtoa](#) (double x)

Static Public Member Functions

- static [SparseMatrix](#) * [convertLinearConstraintCoefficientMatrixToTheOtherMajor](#) (bool isColumnMajor, int startSize, int valueSize, int *start, int *index, double *value, int dimension)
Round a double number to the precision specified.

6.79.1 Detailed Description

this class has routines for linear algebra.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

This class will hold linear algebra routines used by other OS classes. Right now it has a routine to change the column/row storage of a sparse matrix

Definition at line 57 of file OSMathUtil.h.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 MathUtil::MathUtil ()

the class constructor

6.79.2.2 MathUtil::~~MathUtil ()

the class destructor

6.79.3 Member Function Documentation

6.79.3.1 **static SparseMatrix*** MathUtil::convertLinearConstraintCoefficientMatrixToTheOtherMajor (*bool isColumnMajor*, *int startSize*, *int valueSize*, *int * start*, *int * index*, *double * value*, *int dimension*) [static]

Round a double number to the precision specified.

Parameters

<i>x</i>	holds the number to be rounded.
<i>precision</i>	holds the number of digit after (or before if negative) the decimal point.

Returns

the rounded number. Calculation of x mod y.

Parameters

<i>x</i>	holds the number before the mod operator.
<i>y</i>	holds the number after the mod operator.

Returns

the result of x mod y.

Parameters

<i>isColumnMajor</i>	holds whether the coefMatrix (AMatrix) holding linear program data is stored by column. If false, the matrix is stored by row.
<i>start</i>	holds an integer array of start elements in coefMatrix (AMatrix), which points to the start of a column (row) of nonzero elements in coefMatrix (AMatrix).
<i>index</i>	holds an integer array of rowIdx (or colIdx) elements in coefMatrix (AMatrix). If the matrix is stored by column (row), rowIdx (colIdx) is the array of row (column) indices.
<i>value</i>	holds a double array of value elements in coefMatrix (AMatrix), which contains nonzero elements.
<i>dimension</i>	holds the column count if the input matrix is row major (row count = start.length-1) or the row number if the input matrix is column major (column count = start.length -1)

Returns

Linear constraint coefficient matrix in the other major of the input matrix. Return null if input matrix not valid.

6.79.3.2 std::string MathUtil::format_os_dtoa (*double x*)

Parameters

	<code>x</code>	is the double that gets converted into a string this takes the David Gay dtoa and converts to a formatted string
--	----------------	--

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

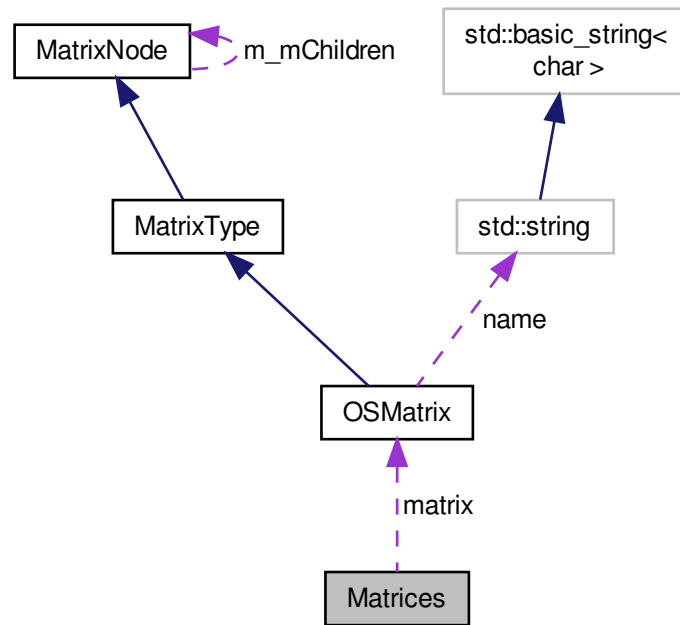
- </home/ted/COIN/trunk/OS/src/OSUtils/OSMathUtil.h>

6.80 Matrices Class Reference

The in-memory representation of the `<matrices>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for Matrices:



Public Member Functions

- [Matrices](#) ()
The [Matrices](#) class constructor.
- [~Matrices](#) ()
The [Matrices](#) class destructor.
- `bool IsEqual (Matrices *that)`
A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

- bool [deepCopyFrom](#) ([Matrices](#) *that)

A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfMatrices](#)

numberOfMatrices is the number of $\langle nl \rangle$ elements in the $\langle \text{matrices} \rangle$ element.

- [OSMatrix](#) ** [matrix](#)

matrix is a pointer to an array of [OSMatrix](#) object pointers

6.80.1 Detailed Description

The in-memory representation of the $\langle \text{matrices} \rangle$ element.

Definition at line 482 of file OSInstance.h.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 [Matrices::Matrices](#) ()

The [Matrices](#) class constructor.

6.80.2.2 [Matrices::~~Matrices](#) ()

The [Matrices](#) class destructor.

6.80.3 Member Function Documentation

6.80.3.1 bool [Matrices::isEqual](#) ([Matrices](#) * *that*)

A function to check for the equality of two objects.

6.80.3.2 bool [Matrices::setRandom](#) (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and $\langle XXX \rangle$ children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.80.3.3 bool [Matrices::deepCopyFrom](#) ([Matrices](#) * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.80.4 Member Data Documentation

6.80.4.1 int Matrices::numberOfMatrices

numberOfMatrices is the number of `<nl>` elements in the `<matrices>` element.

Definition at line 496 of file OSInstance.h.

6.80.4.2 OSMatrix** Matrices::matrix

matrix is a pointer to an array of [OSMatrix](#) object pointers

Definition at line 499 of file OSInstance.h.

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

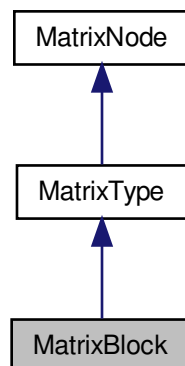
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.81 MatrixBlock Class Reference

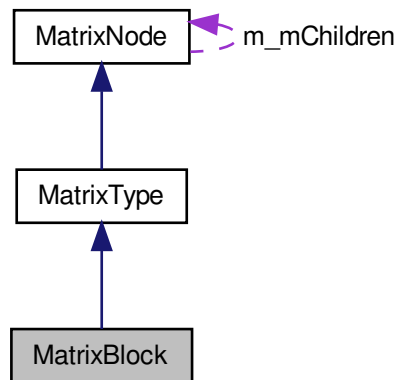
a data structure to represent a [MatrixBlock](#) object (derived from [MatrixType](#))

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixBlock:



Collaboration diagram for MatrixBlock:



Public Member Functions

- [MatrixBlock](#) ()
- [~MatrixBlock](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [MatrixBlock](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [isEqual](#) ([MatrixBlock](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([MatrixBlock](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [blockRowIdx](#)
- int [blockColIdx](#)

6.81.1 Detailed Description

a data structure to represent a [MatrixBlock](#) object (derived from [MatrixType](#))

Definition at line 2183 of file OSMatrix.h.

6.81.2 Constructor & Destructor Documentation

6.81.2.1 MatrixBlock::MatrixBlock ()

6.81.2.2 MatrixBlock::~~MatrixBlock ()

6.81.3 Member Function Documentation

6.81.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixBlock::getNodeTypes () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.81.3.2 virtual std::string MatrixBlock::getNodeName () [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

6.81.3.3 virtual ENUM_MATRIX_TYPE MatrixBlock::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.81.3.4 virtual std::string MatrixBlock::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.81.3.5 virtual bool MatrixBlock::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Reimplemented from [MatrixType](#).

6.81.3.6 `MatrixBlock * MatrixBlock::cloneMatrixNode () [virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [MatrixNode](#) of the proper type.

Implements [MatrixNode](#).

6.81.3.7 `bool MatrixBlock::isEqual (MatrixBlock * that)`

A function to check for the equality of two objects.

6.81.3.8 `bool MatrixBlock::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixType](#).

6.81.3.9 `bool MatrixBlock::deepCopyFrom (MatrixBlock * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.81.4 Member Data Documentation**6.81.4.1** `int MatrixBlock::blockRowIdx`

Definition at line 2186 of file OSMatrix.h.

6.81.4.2 `int MatrixBlock::blockColIdx`

Definition at line 2187 of file OSMatrix.h.

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

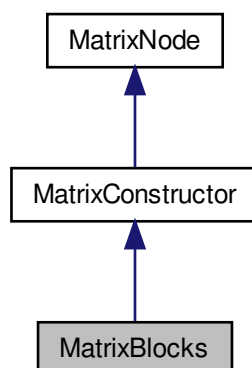
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h`

6.82 MatrixBlocks Class Reference

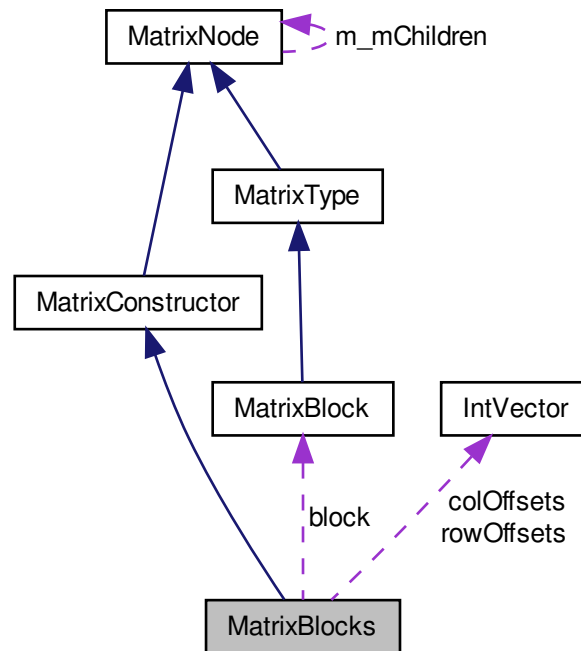
a data structure to represent the nonzeros of a matrix in a blockwise fashion.

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixBlocks:



Collaboration diagram for MatrixBlocks:



Public Member Functions

- [MatrixBlocks](#) ()
- [~MatrixBlocks](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [MatrixBlocks](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [isEqual](#) ([MatrixBlocks](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([MatrixBlocks](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [IntVector](#) * `colOffsets`
An array listing the leftmost column of each block within the larger matrix It is assumed that the blocks are neatly "stacked".
- [IntVector](#) * `rowOffsets`
An array listing the top row of each block within the larger matrix.
- `int` `numberOfBlocks`
*This integer gives the number of blocks for which values are provided Due to block-sparsity, this could be less than $\text{card}(\text{colOffsets}) * \text{card}(\text{rowOffsets})$*
- [MatrixBlock](#) ** `block`
The nonzeros in each block are held in this data structure.

6.82.1 Detailed Description

a data structure to represent the nonzerorees of a matrix in a blockwise fashion.

Each block can be given elementwise, through transformation, or by nested blocks, and so on, recursively.

Definition at line 1450 of file OSMatrix.h.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 `MatrixBlocks::MatrixBlocks ()`

6.82.2.2 `MatrixBlocks::~~MatrixBlocks ()`

6.82.3 Member Function Documentation

6.82.3.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixBlocks::getNodeType ()` [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.82.3.2 `virtual std::string MatrixBlocks::getNodeName ()` [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

6.82.3.3 `virtual ENUM_MATRIX_TYPE MatrixBlocks::getMatrixType ()` [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.82.3.4 virtual std::string MatrixBlocks::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.82.3.5 virtual bool MatrixBlocks::alignsOnBlockBoundary (int *firstRow*, int *firstColumn*, int *nRows*, int *nCols*) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Implements [MatrixNode](#).

6.82.3.6 MatrixBlocks * MatrixBlocks::cloneMatrixNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [MatrixNode](#) of the proper type.

Implements [MatrixNode](#).

6.82.3.7 bool MatrixBlocks::isEqual (MatrixBlocks * *that*)

A function to check for the equality of two objects.

6.82.3.8 bool MatrixBlocks::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.82.3.9 `bool MatrixBlocks::deepCopyFrom (MatrixBlocks * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.82.4 Member Data Documentation**6.82.4.1** `IntVector* MatrixBlocks::colOffsets`

An array listing the leftmost column of each block within the larger matrix It is assumed that the blocks are neatly "stacked".

Definition at line 1457 of file OSMatrix.h.

6.82.4.2 `IntVector* MatrixBlocks::rowOffsets`

An array listing the top row of each block within the larger matrix.

Definition at line 1462 of file OSMatrix.h.

6.82.4.3 `int MatrixBlocks::numberOfBlocks`

This integer gives the number of blocks for which values are provided Due to block-sparsity, this could be less than `card(colOffsets)*card(rowOffsets)`

Definition at line 1468 of file OSMatrix.h.

6.82.4.4 `MatrixBlock** MatrixBlocks::block`

The nonzeros in each block are held in this data structure.

Definition at line 1471 of file OSMatrix.h.

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

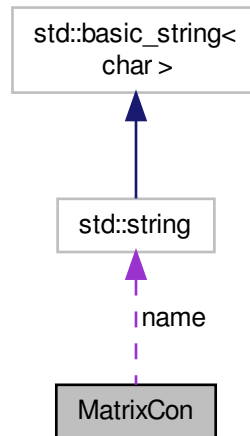
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h>

6.83 MatrixCon Class Reference

The in-memory representation of the `<matrixCon>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixCon:



Public Member Functions

- [MatrixCon](#) ()
The *MatrixCon* class constructor.
- [~MatrixCon](#) ()
The *MatrixCon* class destructor.
- `bool` [isEqual](#) ([MatrixCon](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `int` [numberOfRows](#)
numberOfRows gives the number of rows of this matrix
- `int` [numberOfColumns](#)
numberOfColumns gives the number of columns of this matrix
- `int` [templateMatrixIdx](#)
templateMatrixIdx refers to a matrix that describes the locations in this *matrixVar* that are allowed to be nonzero
- `int` [conReferenceMatrixIdx](#)
conReferenceMatrixIdx allows some or all of the components of this *matrixCon* to be copied from constraints defined in the core
- `int` [lbMatrixIdx](#)
lbMatrixIdx gives a lower bound for this *matrixCon*
- `int` [lbConelIdx](#)
lbConelIdx gives a cone that must contain *matrixCon* - *lbMatrix*
- `int` [ubMatrixIdx](#)
ubMatrixIdx gives an upper bound for this *matrixCon*

- int [ubConeldx](#)
ubConeldx gives a cone that must contain ubMatrix - matrixCon
- std::string [name](#)
an optional name to this [MatrixCon](#)

6.83.1 Detailed Description

The in-memory representation of the <**matrixCon**> element.

Definition at line 1737 of file OSInstance.h.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 MatrixCon::MatrixCon ()

The [MatrixCon](#) class constructor.

6.83.2.2 MatrixCon::~MatrixCon ()

The [MatrixCon](#) class destructor.

6.83.3 Member Function Documentation

6.83.3.1 bool MatrixCon::isEqual (**MatrixCon** * *that*)

A function to check for the equality of two objects.

6.83.4 Member Data Documentation

6.83.4.1 int MatrixCon::numberOfRows

numberOfRows gives the number of rows of this matrix

Definition at line 1741 of file OSInstance.h.

6.83.4.2 int MatrixCon::numberOfColumns

numberOfColumns gives the number of columns of this matrix

Definition at line 1744 of file OSInstance.h.

6.83.4.3 int MatrixCon::templateMatrixIdx

templateMatrixIdx refers to a matrix that describes the locations in this matrixVar that are allowed to be nonzero

Definition at line 1749 of file OSInstance.h.

6.83.4.4 int MatrixCon::conReferenceMatrixIdx

conReferenceMatrixIdx allows some or all of the components of this matrixCon to be copied from constraints defined in the core

Definition at line 1754 of file OSInstance.h.

6.83.4.5 int MatrixCon::lbMatrixIdx

lbMatrixIdx gives a lower bound for this matrixCon

Definition at line 1757 of file OSInstance.h.

6.83.4.6 int MatrixCon::lbConelIdx

lbConelIdx gives a cone that must contain matrixCon - lbMatrix

Definition at line 1760 of file OSInstance.h.

6.83.4.7 int MatrixCon::ubMatrixIdx

ubMatrixIdx gives an upper bound for this matrixCon

Definition at line 1763 of file OSInstance.h.

6.83.4.8 int MatrixCon::ubConelIdx

ubConelIdx gives a cone that must contain ubMatrix - matrixCon

Definition at line 1766 of file OSInstance.h.

6.83.4.9 std::string MatrixCon::name

an optional name to this [MatrixCon](#)

Definition at line 1769 of file OSInstance.h.

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

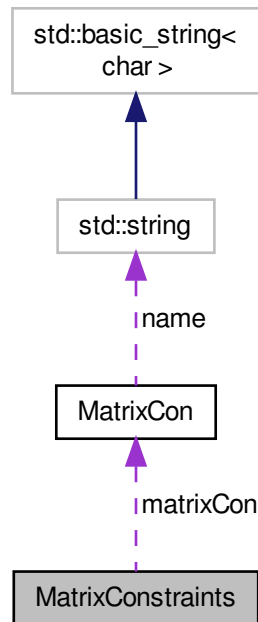
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.84 MatrixConstraints Class Reference

The in-memory representation of the `<matrixConstraints>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixConstraints:



Public Member Functions

- [MatrixConstraints \(\)](#)
The `MatrixConstraints` class constructor.
- [~MatrixConstraints \(\)](#)
The `MatrixConstraints` class destructor.
- `bool` [IsEqual \(MatrixConstraints *that\)](#)
A function to check for the equality of two objects.

Public Attributes

- `int` [numberOfMatrixCon](#)
numberOfMatrixCon gives the number of `<matrixCon>` children
- [MatrixCon ** matrixCon](#)
matrixCon is an array of pointers to the `<matrixCon>` children

6.84.1 Detailed Description

The in-memory representation of the `<matrixConstraints>` element.

Definition at line 1788 of file `OSInstance.h`.

6.84.2 Constructor & Destructor Documentation

6.84.2.1 MatrixConstraints::MatrixConstraints ()

The [MatrixConstraints](#) class constructor.

6.84.2.2 MatrixConstraints::~~MatrixConstraints ()

The [MatrixConstraints](#) class destructor.

6.84.3 Member Function Documentation

6.84.3.1 bool MatrixConstraints::isEqual (MatrixConstraints * *that*)

A function to check for the equality of two objects.

6.84.4 Member Data Documentation

6.84.4.1 int MatrixConstraints::numberOfMatrixCon

numberOfMatrixCon gives the number of <matrixCon> children

Definition at line 1799 of file OSInstance.h.

6.84.4.2 MatrixCon** MatrixConstraints::matrixCon

matrixCon is an array of pointers to the <matrixCon> children

Definition at line 1802 of file OSInstance.h.

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

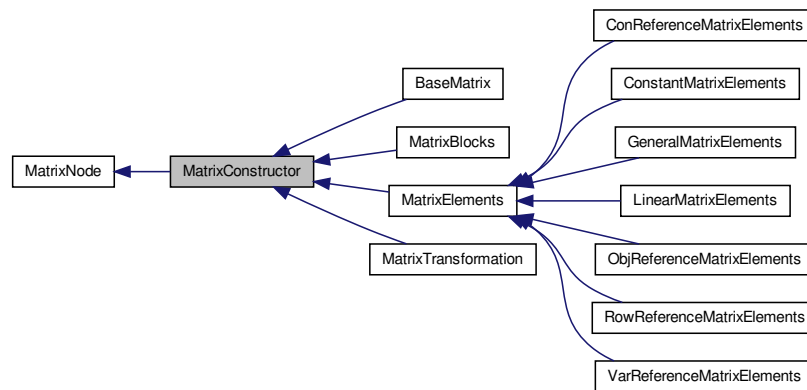
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.85 MatrixConstructor Class Reference

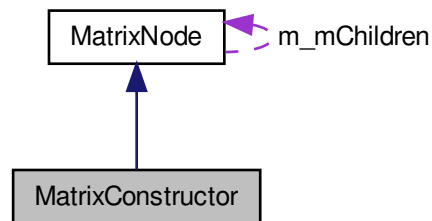
a data structure to describe one step in the construction of a matrix.

```
#include <OSMatrix.h>
```


Inheritance diagram for MatrixConstructor:



Collaboration diagram for MatrixConstructor:



Public Member Functions

- [MatrixConstructor](#) ()
constructor
- virtual [~MatrixConstructor](#) ()
destructor

Additional Inherited Members

6.85.1 Detailed Description

a data structure to describe one step in the construction of a matrix.

To facilitate parsing of complicated matrix constructors and the recursion implicit in the block structure, we distinguish the following types: 1 - [BaseMatrix](#) 2 - several types of Elements (e.g., constant, var reference, etc.) 3 - Transformation 4 - [MatrixBlocks](#) 5 - [MatrixBlock](#) 6 - [OSMatrix](#) Most of the logic of this representation is derived from the [OSnLNode](#) class.

Definition at line 209 of file `OSMatrix.h`.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 `MatrixConstructor::MatrixConstructor ()`

constructor

6.85.2.2 `virtual MatrixConstructor::~~MatrixConstructor () [virtual]`

destructor

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

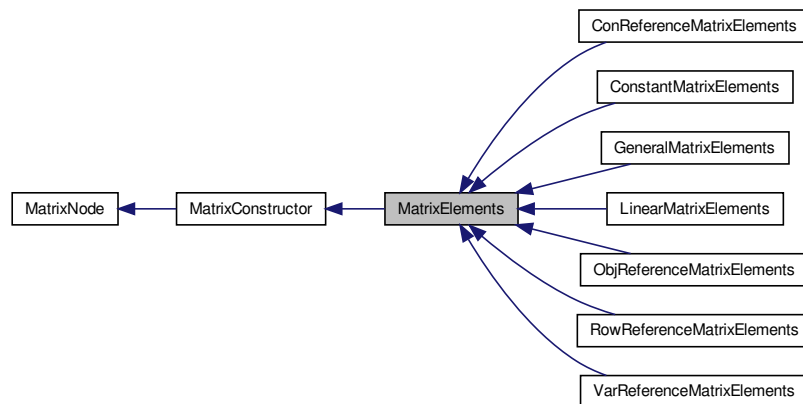
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h`

6.86 MatrixElements Class Reference

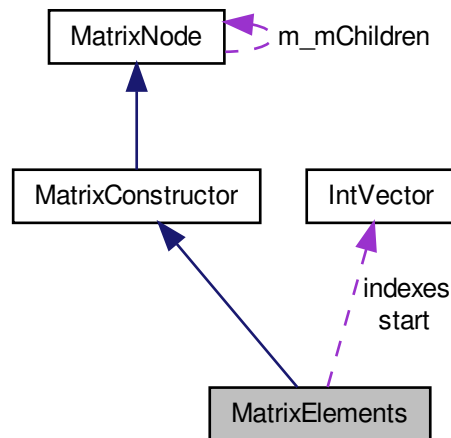
an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixElements:



Collaboration diagram for MatrixElements:



Public Member Functions

- [MatrixElements](#) ()
- virtual [~MatrixElements](#) ()
- bool [getRowMajor](#) ()
Returns whether the matrix is stored row-wise or column-wise.
- bool [isEqual](#) ([MatrixElements](#) *that)
A function to check for the equality of two objects.

Public Attributes

- bool [rowMajor](#)
To indicate whether the matrix elements are stored in row major form or column major form.
- int [numberOfValues](#)
numberOfValues records the number of entries in the arrays that make up the instance of nonzeros
- [IntVector](#) * [start](#)
A vector listing the row or column starts.
- [IntVector](#) * [indexes](#)
The indices of the (nonzero) elements.

6.86.1 Detailed Description

an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.

Definition at line 248 of file OSMatrix.h.

6.86.2 Constructor & Destructor Documentation**6.86.2.1 MatrixElements::MatrixElements ()****6.86.2.2 virtual MatrixElements::~~MatrixElements () [virtual]****6.86.3 Member Function Documentation****6.86.3.1 bool MatrixElements::getRowMajor ()**

Returns whether the matrix is stored row-wise or column-wise.

6.86.3.2 bool MatrixElements::isEqual (MatrixElements * *that*)

A function to check for the equality of two objects.

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

6.86.4 Member Data Documentation**6.86.4.1 bool MatrixElements::rowMajor**

To indicate whether the matrix elements are stored in row major form or column major form.

Definition at line 255 of file OSMatrix.h.

6.86.4.2 int MatrixElements::numberOfValues

numberOfValues records the number of entries in the arrays that make up the instance of nonzeros

Definition at line 261 of file OSMatrix.h.

6.86.4.3 IntVector* MatrixElements::start

A vector listing the row or column starts.

Definition at line 266 of file OSMatrix.h.

6.86.4.4 IntVector* MatrixElements::indexes

The indices of the (nonzero) elements.

Definition at line 269 of file OSMatrix.h.

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

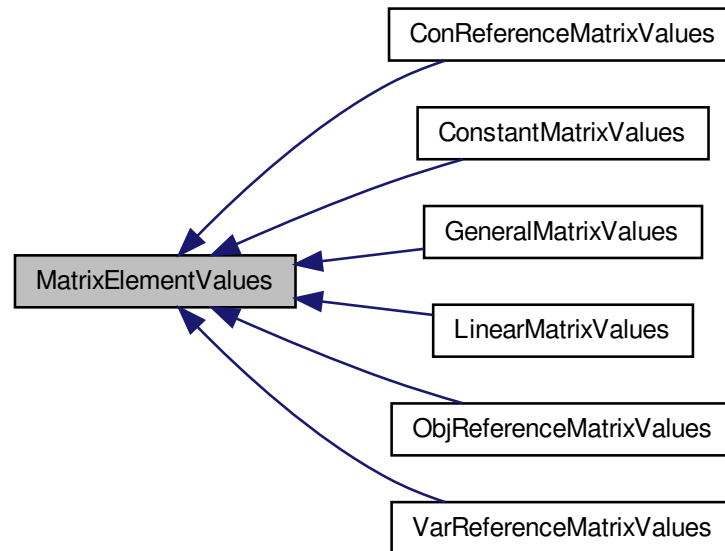
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.87 MatrixElementValues Class Reference

an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixElementValues:



Public Member Functions

- [MatrixElementValues](#) ()
- virtual [~MatrixElementValues](#) ()

Public Attributes

- int [numberOfEI](#)
each type of values is stored as an array named "el".

6.87.1 Detailed Description

an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc. Definition at line 321 of file OSMatrix.h.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 [MatrixElementValues::MatrixElementValues](#) ()

6.87.2.2 [virtual MatrixElementValues::~~MatrixElementValues](#) () [virtual]

6.87.3 Member Data Documentation

6.87.3.1 int MatrixElementValues::numberOfEI

each type of values is stored as an array named "el".

numberOfEI records the size of this array.

Definition at line 327 of file OSMatrix.h.

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

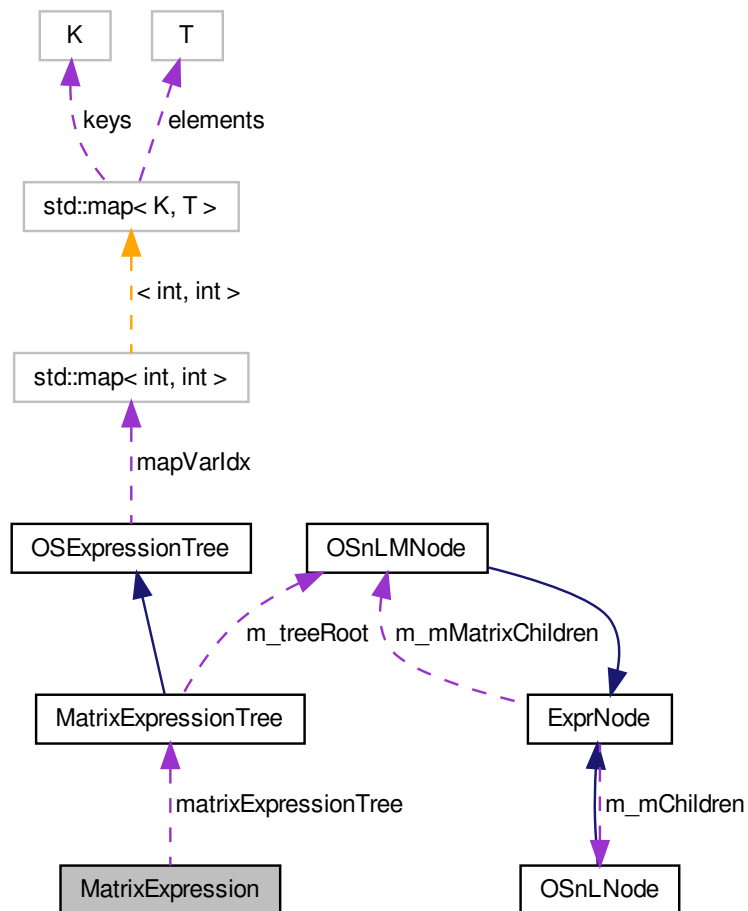
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.88 MatrixExpression Class Reference

The in-memory representation of the `<expr>` element, which is like a nonlinear expression, but since it involves matrices, the expression could be linear, so a "shape" attribute is added to distinguish linear and nonlinear expressions.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixExpression:



Public Member Functions

- [MatrixExpression](#) ()
The [MatrixExpression](#) class constructor.
- [~MatrixExpression](#) ()
The [MatrixExpression](#) class destructor.
- bool [IsEqual](#) ([MatrixExpression](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
idx holds the row index of the nonlinear expression
- [ENUM_NL_EXPR_SHAPE](#) shape
shape holds the shape of the nonlinear expression (linear/quadratic/convex/general) (see further up in this file).
- [MatrixExpressionTree](#) * [matrixExpressionTree](#)
matrixExpressionTree contains the root of the [MatrixExpressionTree](#)
- bool [m_bDeleteExpressionTree](#)
if [m_bDeleteExpressionTree](#) is true during garbage collection, we should delete the [osExpression](#) tree object, if the [OSInstance](#) class created a map of the expression trees, this should be false since the [osExpressionTree](#) is deleted by the [OSInstance](#) object

6.88.1 Detailed Description

The in-memory representation of the <expr> element, which is like a nonlinear expression, but since it involves matrices, the expression could be linear, so a "shape" attribute is added to distinguish linear and nonlinear expressions.

Definition at line 1817 of file [OSInstance.h](#).

6.88.2 Constructor & Destructor Documentation

6.88.2.1 [MatrixExpression::MatrixExpression \(\)](#)

The [MatrixExpression](#) class constructor.

6.88.2.2 [MatrixExpression::~~MatrixExpression \(\)](#)

The [MatrixExpression](#) class destructor.

6.88.3 Member Function Documentation

6.88.3.1 bool [MatrixExpression::IsEqual \(\[MatrixExpression\]\(#\) * that \)](#)

A function to check for the equality of two objects.

6.88.4 Member Data Documentation

6.88.4.1 int [MatrixExpression::idx](#)

idx holds the row index of the nonlinear expression

Definition at line 1821 of file OSInstance.h.

6.88.4.2 `ENUM_NL_EXPR_SHAPE` `MatrixExpression::shape`

`shape` holds the shape of the nonlinear expression (linear/quadratic/convex/general) (see further up in this file).

this might be useful in guiding solver selection.

Definition at line 1827 of file OSInstance.h.

6.88.4.3 `MatrixExpressionTree*` `MatrixExpression::matrixExpressionTree`

`matrixExpressionTree` contains the root of the [MatrixExpressionTree](#)

Definition at line 1830 of file OSInstance.h.

6.88.4.4 `bool` `MatrixExpression::m_bDeleteExpressionTree`

if `m_bDeleteExpressionTree` is true during garbage collection, we should delete the `osExpression` tree object, if the [OSInstance](#) class created a map of the expression trees, this should be false since the `osExpressionTree` is deleted by the [OSInstance](#) object

Definition at line 1837 of file OSInstance.h.

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

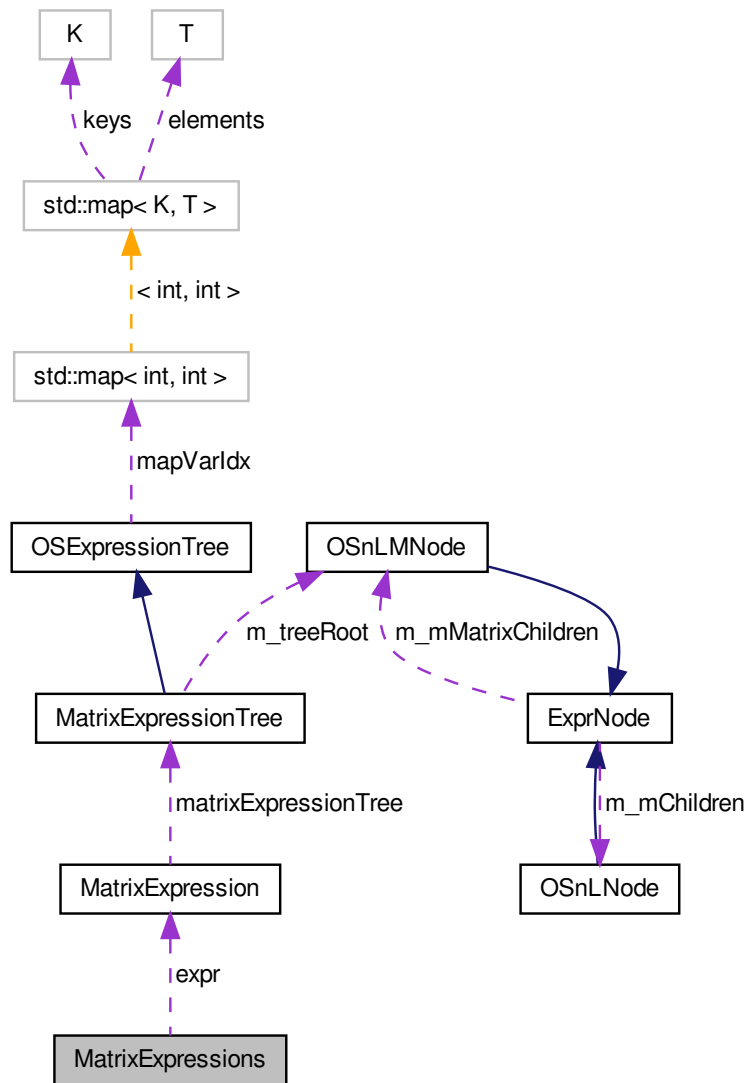
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h`

6.89 MatrixExpressions Class Reference

The in-memory representation of the `<matrixExpressions>` element.

```
#include <OSInstance.h>
```


Collaboration diagram for MatrixExpressions:



Public Member Functions

- [MatrixExpressions](#) ()
The *MatrixExpressions* class constructor.
- [~MatrixExpressions](#) ()
The *MatrixExpressions* class destructor.
- `bool` [isEqual](#) ([MatrixExpressions](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfExpr](#)
numberOfExpr gives the number of expressions
- [MatrixExpression](#) ** [expr](#)
a pointer to an array of linear and nonlinear expressions that evaluate to matrices

6.89.1 Detailed Description

The in-memory representation of the <**matrixExpressions**> element.

Definition at line 1856 of file OSInstance.h.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 MatrixExpressions::MatrixExpressions ()

The [MatrixExpressions](#) class constructor.

6.89.2.2 MatrixExpressions::~~MatrixExpressions ()

The [MatrixExpressions](#) class destructor.

6.89.3 Member Function Documentation

6.89.3.1 bool MatrixExpressions::isEqual ([MatrixExpressions](#) * *that*)

A function to check for the equality of two objects.

6.89.4 Member Data Documentation

6.89.4.1 int MatrixExpressions::numberOfExpr

numberOfExpr gives the number of expressions

Definition at line 1860 of file OSInstance.h.

6.89.4.2 [MatrixExpression](#) ** [MatrixExpressions::expr](#)

a pointer to an array of linear and nonlinear expressions that evaluate to matrices

Definition at line 1865 of file OSInstance.h.

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

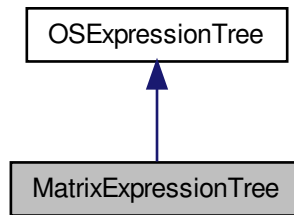
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.90 MatrixExpressionTree Class Reference

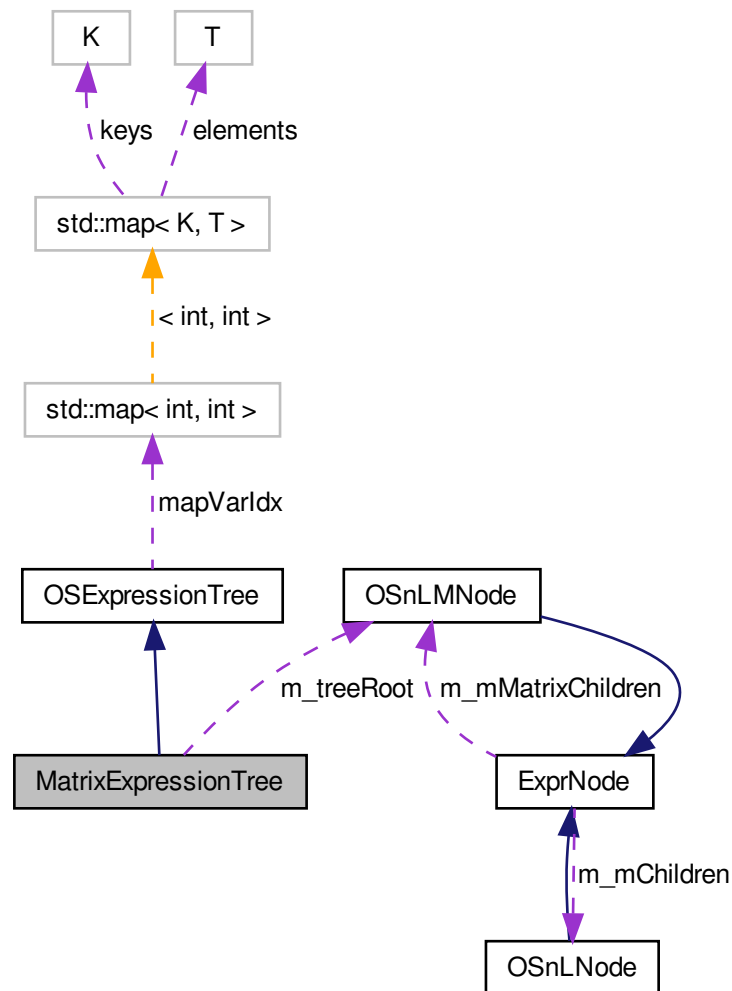
Used to hold the instance in memory.

```
#include <OSExpressionTree.h>
```

Inheritance diagram for MatrixExpressionTree:



Collaboration diagram for MatrixExpressionTree:



Public Member Functions

- [MatrixExpressionTree](#) ()
default constructor.
- [~MatrixExpressionTree](#) ()
default destructor.
- `bool` [IsEqual](#) ([MatrixExpressionTree](#) *that)
A function to check for the equality of two objects.
- `std::vector< ExprNode * >` [getPrefixFromExpressionTree](#) ()
Get a vector of pointers to ExprNodes that correspond to a scalar-valued OSExpressionTree in prefix format.
- `std::vector< ExprNode * >` [getPostfixFromExpressionTree](#) ()

Get a vector of pointers to ExprNodes that correspond to a scalar-valued [OSExpressionTree](#) in postfix format.

Public Attributes

- [OSnLMNode](#) * `m_treeRoot`

`m_treeRoot` holds the root node (of [OSnLMNode](#) type) of the expression tree.

6.90.1 Detailed Description

Used to hold the instance in memory.

Remarks

This class stores a matrix-valued linear or nonlinear expression in memory as an expression tree.

Definition at line 201 of file `OSExpressionTree.h`.

6.90.2 Constructor & Destructor Documentation

6.90.2.1 MatrixExpressionTree::MatrixExpressionTree ()

default constructor.

6.90.2.2 MatrixExpressionTree::~MatrixExpressionTree ()

default destructor.

6.90.3 Member Function Documentation

6.90.3.1 bool MatrixExpressionTree::isEqual (MatrixExpressionTree * that)

A function to check for the equality of two objects.

6.90.3.2 std::vector<ExprNode*> MatrixExpressionTree::getPrefixFromExpressionTree ()

Get a vector of pointers to ExprNodes that correspond to a scalar-valued [OSExpressionTree](#) in prefix format.

Returns

the expression tree as a vector of ExprNodes in prefix.

6.90.3.3 std::vector<ExprNode*> MatrixExpressionTree::getPostfixFromExpressionTree ()

Get a vector of pointers to ExprNodes that correspond to a scalar-valued [OSExpressionTree](#) in postfix format.

Returns

the expression tree as a vector of ExprNodes in postfix.

6.90.4 Member Data Documentation

6.90.4.1 OSnLMNode* MatrixExpressionTree::m_treeRoot

m_treeRoot holds the root node (of [OSnLMNode](#) type) of the expression tree.

Definition at line 207 of file OSExpressionTree.h.

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

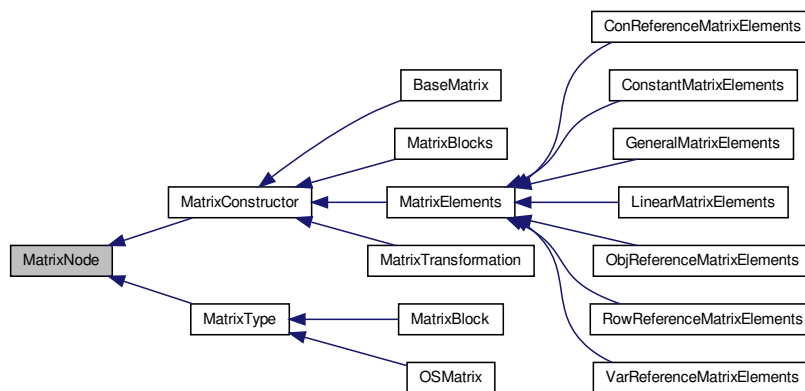
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTree.h](#)

6.91 MatrixNode Class Reference

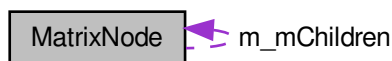
a generic class from which we derive matrix constructors ([BaseMatrix](#), [MatrixElements](#), [MatrixTransformation](#) and [MatrixBlocks](#)) as well as matrix types ([OSMatrix](#) and [MatrixBlock](#)).

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixNode:



Collaboration diagram for MatrixNode:



Public Member Functions

- [MatrixNode](#) ()

- default constructor*
- virtual [~MatrixNode](#) ()
- destructor*
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()=0
- virtual std::string [getNodeName](#) ()=0
- virtual std::string [getMatrixNodeInXML](#) ()=0
- std::vector< [MatrixNode](#) * > [getPrefixFromNodeTree](#) ()
- std::vector< [MatrixNode](#) * > [preOrderMatrixNodeTraversal](#) (std::vector< [MatrixNode](#) * > *prefixVector)
- std::vector< [MatrixNode](#) * > [getPostfixFromNodeTree](#) ()
- std::vector< [MatrixNode](#) * > [postOrderMatrixNodeTraversal](#) (std::vector< [MatrixNode](#) * > *postfixVector)
- virtual [MatrixNode](#) * [cloneMatrixNode](#) ()=0
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)=0
- Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.*
- virtual bool [isEqual](#) ([MatrixNode](#) *that)
- A function to check for the equality of two objects.*
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
- A function to make a random instance of this class.*
- bool [deepCopyFrom](#) ([MatrixNode](#) *that)
- A function to make a deep copy of an instance of this class.*

Public Attributes

- [ENUM_MATRIX_TYPE](#) [matrixType](#)
- matrixType tracks the type of elements contained in this [MatrixNode](#), which may be useful in solver selection For an enumeration of the possible types see [OSParameters.h](#)*
- [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [nType](#)
- nType is a unique integer assigned to each type of matrix node (see [OSParameters.h](#))*
- unsigned int [inumberOfChildren](#)
- inumberOfChildren is the number of [MatrixNode](#) child elements For the matrix types ([OSMatrix](#) and [MatrixBlock](#)) this number is not fixed and is temporarily set to 0*
- [MatrixNode](#) ** [m_mChildren](#)
- m_mChildren holds all the children, that is, nodes used in the definition or construction of the current node.*

6.91.1 Detailed Description

a generic class from which we derive matrix constructors ([BaseMatrix](#), [MatrixElements](#), [MatrixTransformation](#) and [MatrixBlocks](#)) as well as matrix types ([OSMatrix](#) and [MatrixBlock](#)).

Definition at line 50 of file [OSMatrix.h](#).

6.91.2 Constructor & Destructor Documentation

6.91.2.1 [MatrixNode::MatrixNode](#) ()

default constructor

6.91.2.2 `virtual MatrixNode::~~MatrixNode () [virtual]`

destructor

6.91.3 Member Function Documentation

6.91.3.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixNode::getNodeType () [virtual]`

Returns

the value of nType

Reimplemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.2 `virtual ENUM_MATRIX_TYPE MatrixNode::getMatrixType () [pure virtual]`

Returns

the type of the matrix elements

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.3 `virtual std::string MatrixNode::getNodeName () [pure virtual]`

Returns

the name of the matrix constructor

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.4 `virtual std::string MatrixNode::getMatrixNodeInXML () [pure virtual]`

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.5 `std::vector<MatrixNode*> MatrixNode::getPrefixFromNodeTree ()`

Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the [MatrixNode](#) tree in prefix format.

Returns

the node tree as a vector of MatrixNodes in prefix.

6.91.3.6 `std::vector<MatrixNode*> MatrixNode::preOrderMatrixNodeTraversal (std::vector< MatrixNode * > * prefixVector)`

6.91.3.7 `std::vector<MatrixNode*> MatrixNode::getPostfixFromNodeTree ()`

Get a vector of pointers to MatrixNodes that correspond to the [MatrixNode](#) tree in postfix format

Returns

the node tree as a vector of MatrixNodes in postfix.

6.91.3.8 `std::vector<MatrixNode*> MatrixNode::postOrderMatrixNodeTraversal (std::vector< MatrixNode * > * postfixVector)`

Called by [getPostfixFromNodeTree\(\)](#). This method calls itself recursively and generates a vector of pointers to MatrixNodes in postfix.

Parameters

<i>a</i>	pointer postfixVector to a vector of pointers of MatrixNodes
----------	--

Returns

a vector of pointers to MatrixNodes in postfix.

6.91.3.9 `virtual MatrixNode* MatrixNode::cloneMatrixNode () [pure virtual]`

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.10 `virtual bool MatrixNode::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [pure virtual]`

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implemented in [MatrixBlock](#), [OSMatrix](#), [MatrixType](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.11 `virtual bool MatrixNode::isEqual (MatrixNode * that) [virtual]`

A function to check for the equality of two objects.

6.91.3.12 `bool MatrixNode::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented in [MatrixBlock](#), [OSMatrix](#), [MatrixType](#), [MatrixBlocks](#), [MatrixTransformation](#), [RowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

6.91.3.13 `bool MatrixNode::deepCopyFrom (MatrixNode * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.91.4 Member Data Documentation

6.91.4.1 `ENUM_MATRIX_TYPE MatrixNode::matrixType`

`matrixType` tracks the type of elements contained in this [MatrixNode](#), which may be useful in solver selection For an enumeration of the possible types see [OSParameters.h](#)

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

6.91.4.2 `ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixNode::nType`

`nType` is a unique integer assigned to each type of matrix node (see [OSParameters.h](#))

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

6.91.4.3 `unsigned int MatrixNode::numberOfChildren`

`numberOfChildren` is the number of [MatrixNode](#) child elements For the matrix types ([OSMatrix](#) and [MatrixBlock](#)) this number is not fixed and is temporarily set to 0

Definition at line 70 of file [OSMatrix.h](#).

6.91.4.4 `MatrixNode** MatrixNode::m_mChildren`

`m_mChildren` holds all the children, that is, nodes used in the definition or construction of the current node.

Definition at line 76 of file OSMatrix.h.

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

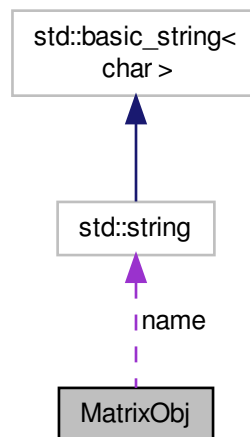
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h>

6.92 MatrixObj Class Reference

The in-memory representation of the `<matrixObj>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixObj:



Public Member Functions

- [MatrixObj \(\)](#)
The `MatrixVar` class constructor.
- [~MatrixObj \(\)](#)
The `MatrixVar` class destructor.
- `bool IsEqual (MatrixObj *that)`
A function to check for the equality of two objects.

Public Attributes

- `int numberOfRows`
numberOfRows gives the number of rows of this matrix
- `int numberOfColumns`
numberOfColumns gives the number of columns of this matrix
- `int templateMatrixIdx`

templateMatrixIdx refers to a matrix that describes the locations in this *matrixObj* that are allowed to be nonzero

- int [objReferenceMatrixIdx](#)

objReferenceMatrixIdx allows some or all of the components of this *matrixObj* to be copied from objectives defined in the *core*

- int [orderConeldx](#)

orderConeldx gives a cone that expresses preferences during the optimization *x* is (weakly) preferred to *y* if *obj(x) - obj(y)* lies in the cone.

- int [constantMatrixIdx](#)

constantMatrixIdx gives a constant added to the *matrixObj*

- std::string [name](#)

an optional name to this *matrixObj*

6.92.1 Detailed Description

The in-memory representation of the `<matrixObj>` element.

Definition at line 1662 of file `OSInstance.h`.

6.92.2 Constructor & Destructor Documentation

6.92.2.1 MatrixObj::MatrixObj ()

The [MatrixVar](#) class constructor.

6.92.2.2 MatrixObj::~~MatrixObj ()

The [MatrixVar](#) class destructor.

6.92.3 Member Function Documentation

6.92.3.1 bool MatrixObj::isEqual (MatrixObj * that)

A function to check for the equality of two objects.

6.92.4 Member Data Documentation

6.92.4.1 int MatrixObj::numberOfRows

numberOfRows gives the number of rows of this matrix

Definition at line 1666 of file `OSInstance.h`.

6.92.4.2 int MatrixObj::numberOfColumns

numberOfColumns gives the number of columns of this matrix

Definition at line 1669 of file `OSInstance.h`.

6.92.4.3 int MatrixObj::templateMatrixIdx

templateMatrixIdx refers to a matrix that describes the locations in this *matrixObj* that are allowed to be nonzero

Definition at line 1674 of file `OSInstance.h`.

6.92.4.4 int MatrixObj::objReferenceMatrixIdx

objReferenceMatrixIdx allows some or all of the components of this matrixObj to be copied from objectives defined in the core

Definition at line 1679 of file OSInstance.h.

6.92.4.5 int MatrixObj::orderConeldx

orderConeldx gives a cone that expresses preferences during the optimization x is (weakly) preferred to y if obj(x) - obj(y) lies in the cone.

Definition at line 1684 of file OSInstance.h.

6.92.4.6 int MatrixObj::constantMatrixIdx

constantMatrixIdx gives a constant added to the matrixObj

Definition at line 1687 of file OSInstance.h.

6.92.4.7 std::string MatrixObj::name

an optional name to this matrixObj

Definition at line 1690 of file OSInstance.h.

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

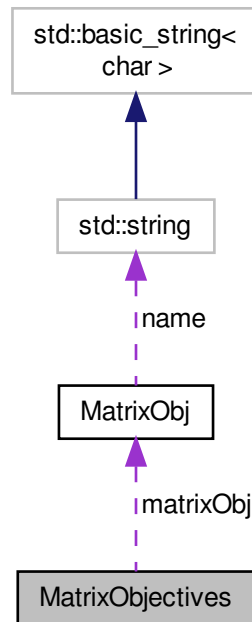
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.93 MatrixObjectives Class Reference

The in-memory representation of the `<matrixObjectives>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixObjectives:



Public Member Functions

- [MatrixObjectives](#) ()
The *MatrixObjectives* class constructor.
- [~MatrixObjectives](#) ()
The *MatrixObjectives* class destructor.
- `bool` [IsEqual](#) ([MatrixObjectives](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `int` [numberOfMatrixObj](#)
numberOfMatrixObj gives the number of `<matrixObj>` children
- [MatrixObj](#) ** [matrixObj](#)
matrixObj is an array of pointers to the `<matrixObj>` children

6.93.1 Detailed Description

The in-memory representation of the `<matrixObjectives>` element.

Definition at line 1709 of file `OSInstance.h`.

6.93.2 Constructor & Destructor Documentation

6.93.2.1 MatrixObjectives::MatrixObjectives ()

The [MatrixObjectives](#) class constructor.

6.93.2.2 MatrixObjectives::~~MatrixObjectives ()

The [MatrixObjectives](#) class destructor.

6.93.3 Member Function Documentation

6.93.3.1 bool MatrixObjectives::isEqual (MatrixObjectives * *that*)

A function to check for the equality of two objects.

6.93.4 Member Data Documentation

6.93.4.1 int MatrixObjectives::numberOfMatrixObj

numberOfMatrixObj gives the number of <matrixObj> children

Definition at line 1720 of file OSInstance.h.

6.93.4.2 MatrixObj** MatrixObjectives::matrixObj

matrixObj is an array of pointers to the <matrixObj> children

Definition at line 1723 of file OSInstance.h.

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

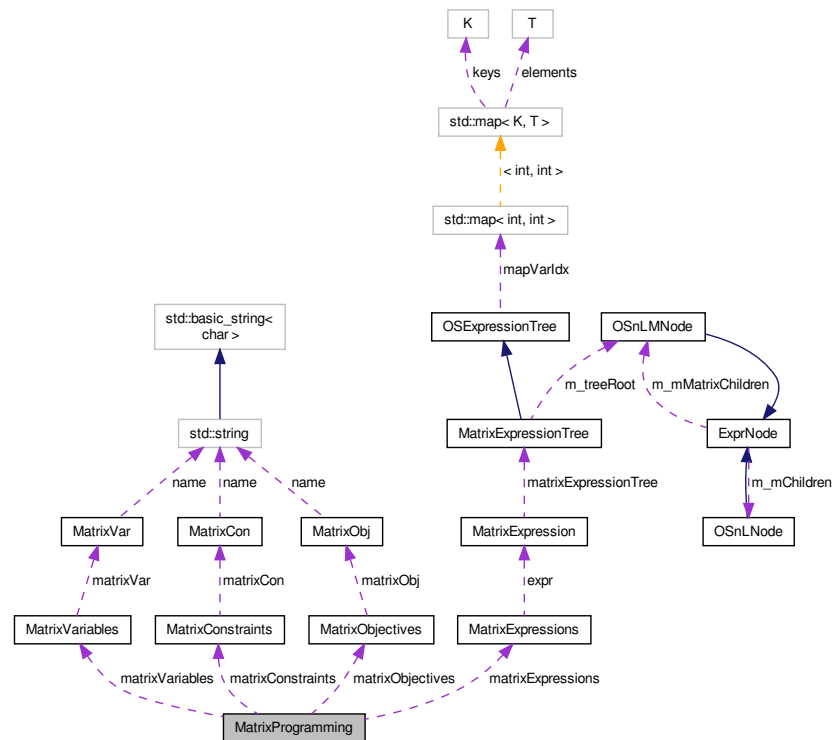
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.94 MatrixProgramming Class Reference

The in-memory representation of the <matrixProgramming> element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixProgramming:



Public Member Functions

- [MatrixProgramming \(\)](#)
The *MatrixProgramming* class constructor.
- [~MatrixProgramming \(\)](#)
The *MatrixProgramming* class destructor.
- [bool IsEqual \(MatrixProgramming *that\)](#)
A function to check for the equality of two objects.
- [bool setRandom \(double density, bool conformant, int iMin, int iMax\)](#)
A function to make a random instance of this class.
- [bool deepCopyFrom \(MatrixProgramming *that\)](#)
A function to make a deep copy of an instance of this class.

Public Attributes

- [MatrixVariables * matrixVariables](#)
a pointer to the *matrixVariables* object
- [MatrixObjectives * matrixObjectives](#)
a pointer to the *matrixObjectives* object
- [MatrixConstraints * matrixConstraints](#)

a pointer to the matrixConstraints object

- [MatrixExpressions](#) * [matrixExpressions](#)

a pointer to the matrixExpressions object

6.94.1 Detailed Description

The in-memory representation of the `<matrixProgramming>` element.

Definition at line 1883 of file OSInstance.h.

6.94.2 Constructor & Destructor Documentation

6.94.2.1 MatrixProgramming::MatrixProgramming ()

The [MatrixProgramming](#) class constructor.

6.94.2.2 MatrixProgramming::~~MatrixProgramming ()

The [MatrixProgramming](#) class destructor.

6.94.3 Member Function Documentation

6.94.3.1 bool MatrixProgramming::isEqual (MatrixProgramming * that)

A function to check for the equality of two objects.

6.94.3.2 bool MatrixProgramming::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

6.94.3.3 bool MatrixProgramming::deepCopyFrom (MatrixProgramming * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.94.4 Member Data Documentation

6.94.4.1 MatrixVariables* MatrixProgramming::matrixVariables

a pointer to the matrixVariables object

Definition at line 1893 of file OSInstance.h.

6.94.4.2 MatrixObjectives* MatrixProgramming::matrixObjectives

a pointer to the matrixObjectives object

Definition at line 1896 of file OSInstance.h.

6.94.4.3 MatrixConstraints* MatrixProgramming::matrixConstraints

a pointer to the matrixConstraints object

Definition at line 1899 of file OSInstance.h.

6.94.4.4 MatrixExpressions* MatrixProgramming::matrixExpressions

a pointer to the matrixExpressions object

Definition at line 1902 of file OSInstance.h.

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

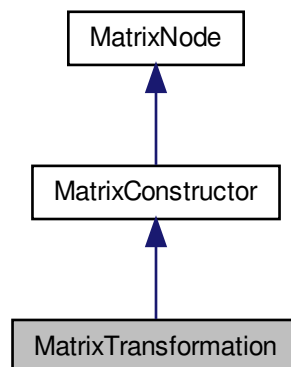
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.95 MatrixTransformation Class Reference

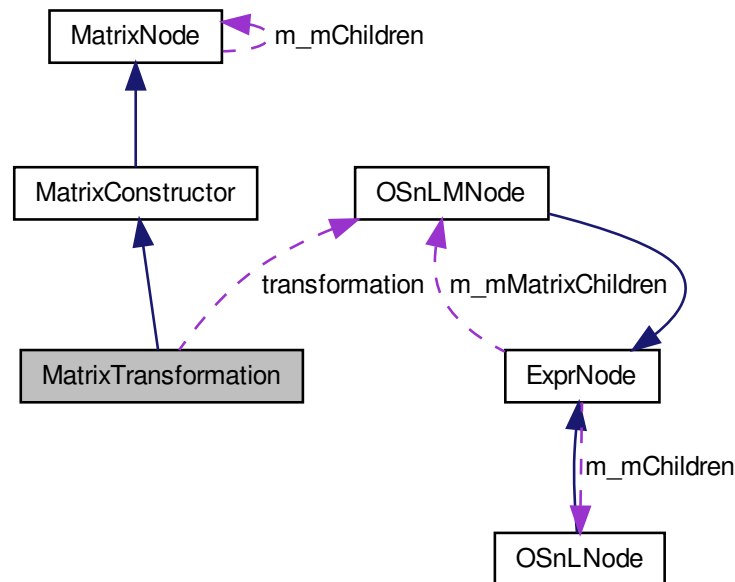
a data structure to represent the nonzeros of a matrix by transformation from other (previously defined) matrices

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixTransformation:



Collaboration diagram for MatrixTransformation:



Public Member Functions

- [MatrixTransformation](#) ()
- [~MatrixTransformation](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual `std::string` [getNodeName](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual `std::string` [getMatrixNodeInXML](#) ()
- virtual `bool` [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [MatrixTransformation](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- `bool` [isEqual](#) ([MatrixTransformation](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([MatrixTransformation](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [OSnLMNode](#) * [transformation](#)
A transformation is essentially an expression tree that evaluates to a matrix.
- [ENUM_NL_EXPR_SHAPE](#) [shape](#)
shape can be used to specify linearity etc.

6.95.1 Detailed Description

a data structure to represent the nonzerose of a matrix by transformation from other (previously defined) matrices
Definition at line 1359 of file OSMatrix.h.

6.95.2 Constructor & Destructor Documentation

6.95.2.1 `MatrixTransformation::MatrixTransformation ()`

6.95.2.2 `MatrixTransformation::~~MatrixTransformation ()`

6.95.3 Member Function Documentation

6.95.3.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixTransformation::getNodeType ()` [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.95.3.2 `virtual std::string MatrixTransformation::getNodeName ()` [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

6.95.3.3 `virtual ENUM_MATRIX_TYPE MatrixTransformation::getMatrixType ()` [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.95.3.4 `virtual std::string MatrixTransformation::getMatrixNodeInXML ()` [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.95.3.5 `virtual bool MatrixTransformation::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols)`
`[virtual]`

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Implements [MatrixNode](#).

6.95.3.6 `MatrixTransformation * MatrixTransformation::cloneMatrixNode ()` `[virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [MatrixNode](#) of the proper type.

Implements [MatrixNode](#).

6.95.3.7 `bool MatrixTransformation::isEqual (MatrixTransformation * that)`

A function to check for the equality of two objects.

6.95.3.8 `bool MatrixTransformation::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.95.3.9 `bool MatrixTransformation::deepCopyFrom (MatrixTransformation * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.95.4 Member Data Documentation**6.95.4.1 OSnLMNode* MatrixTransformation::transformation**

A transformation is essentially an expression tree that evaluates to a matrix.

Definition at line 1365 of file OSMatrix.h.

6.95.4.2 ENUM_NL_EXPR_SHAPE MatrixTransformation::shape

shape can be used to specify linearity etc.

of an expression For possible values, see OSParamaters.h

Definition at line 1371 of file OSMatrix.h.

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

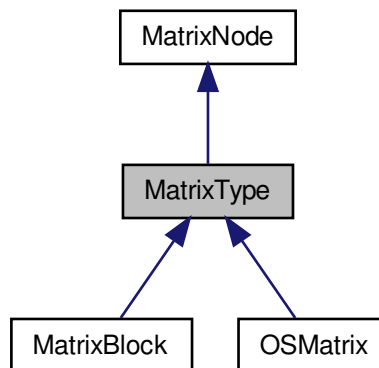
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSMatrix.h](#)

6.96 MatrixType Class Reference

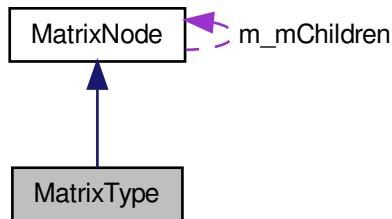
a data structure to represent a [MatrixType](#) object (from which we derive [OSMatrix](#) and [MatrixBlock](#))

```
#include <OSMatrix.h>
```

Inheritance diagram for MatrixType:



Collaboration diagram for MatrixType:



Public Member Functions

- [MatrixType](#) ()
- virtual [~MatrixType](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
A method to check whether a matrix or block is diagonal.
- bool [matrixHasBase](#) ()
Several tools to parse the constructor list of a matrix.
- bool [matrixHasElements](#) ()
- bool [matrixHasTransformations](#) ()
- bool [matrixHasBlocks](#) ()
- int [getNumberOfElementConstructors](#) ()
- int [getNumberOfTransformationConstructors](#) ()
- int [getNumberOfBlocksConstructors](#) ()
- [GeneralSparseMatrix](#) * [getMatrixInColumnMajorForm](#) ()
- [GeneralSparseMatrix](#) * [getMatrixInRowMajorForm](#) ()
- [GeneralSparseMatrix](#) * [getMatrixBlockInColumnMajorForm](#) (int columnIdx, int rowIdx)
- bool [processBlocks](#) (int *rowOffsets, int *colOffsets, bool rowMajor, [ENUM_MATRIX_SYMMETRY](#) symmetry)
A method to expand a matrix or block The result is a [GeneralSparseMatrix](#) object of constant matrix elements, variable references, linear or nonlinear expressions, or objective and constraint references (possibly mixed).
- [GeneralSparseMatrix](#) * [extractBlock](#) (int firstrow, int firstcol, int nrows, int ncols, bool rowMajor, [ENUM_MATRIX_SYMMETRY](#) symmetry)
A method to extract a block from a larger matrix The result is a sparse matrix object, depending on the matrixType, of constant matrix elements, variable references, linear or nonlinear expressions, or objective and constraint references (possibly mixed).
- bool [isEqual](#) ([MatrixType](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([MatrixType](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ENUM_MATRIX_SYMMETRY](#) *symmetry*
To track the type of symmetry present in the matrix or block.
- [ENUM_MATRIX_TYPE](#) *type*
To track the type of values present in the matrix or block.
- int [numberOfRows](#)
- int [numberOfColumns](#)
- bool [haveExpandedForm](#)
The expanded form of the matrix is held in four sparse matrix objects: *m_mmConstantElements* *m_mmVariableReferences* *m_mmGeneralElements* *m_mmObjAndConReferences*.

6.96.1 Detailed Description

a data structure to represent a [MatrixType](#) object (from which we derive [OSMatrix](#) and [MatrixBlock](#))

Definition at line 1849 of file [OSMatrix.h](#).

6.96.2 Constructor & Destructor Documentation

6.96.2.1 [MatrixType::MatrixType \(\)](#)6.96.2.2 [virtual MatrixType::~MatrixType \(\)](#) [[virtual](#)]

6.96.3 Member Function Documentation

6.96.3.1 [virtual bool MatrixType::alignsOnBlockBoundary \(int *firstRow*, int *firstColumn*, int *nRows*, int *nCols* \)](#) [[virtual](#)]

A method to check whether a matrix or block is diagonal.

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Implements [MatrixNode](#).

Reimplemented in [MatrixBlock](#), and [OSMatrix](#).

6.96.3.2 [bool MatrixType::matrixHasBase \(\)](#)

Several tools to parse the constructor list of a matrix.

6.96.3.3 [bool MatrixType::matrixHasElements \(\)](#)6.96.3.4 [bool MatrixType::matrixHasTransformations \(\)](#)

6.96.3.5 `bool MatrixType::matrixHasBlocks ()`

6.96.3.6 `int MatrixType::getNumberOfElementConstructors ()`

6.96.3.7 `int MatrixType::getNumberOfTransformationConstructors ()`

6.96.3.8 `int MatrixType::getNumberOfBlocksConstructors ()`

6.96.3.9 `GeneralSparseMatrix* MatrixType::getMatrixInColumnMajorForm ()`

6.96.3.10 `GeneralSparseMatrix* MatrixType::getMatrixInRowMajorForm ()`

6.96.3.11 `GeneralSparseMatrix* MatrixType::getMatrixBlockInColumnMajorForm (int columnIndex, int rowIdx)`

6.96.3.12 `bool MatrixType::processBlocks (int * rowOffsets, int * colOffsets, bool rowMajor, ENUM_MATRIX_SYMMETRY symmetry)`

A method to expand a matrix or block The result is a [GeneralSparseMatrix](#) object of constant matrix elements, variable references, linear or nonlinear expressions, or objective and constraint references (possibly mixed).

(Values depend on the matrixType.) Duplicate elements are removed according to the rules formulated in the OSiL schema.

Parameters

<i>rowMajor</i>	can be used to store the objects in row major form.
-----------------	---

Returns

whether the operation was successful or not. A method to process a matrixType into a specific block structure.

Parameters

<i>rowOffsets</i>	defines a partition of the matrix rows into the blocks
<i>colOffsets</i>	defines a partition of the matrix columns into the blocks
<i>symmetry</i>	can be used to store only the upper or lower triangle, depending on the parameter value — see OSParameters.h for definitions

Returns

whether the operation was successful

Remarks

The blocks are stored into a `std::vector` of type `expandedMatrixBlocks` so that they can be retrieved later using `extractBlock` (see below). It is possible (though probably not advisable) to maintain multiple decompositions with different row and column partitions

6.96.3.13 `GeneralSparseMatrix* MatrixType::extractBlock (int firstrow, int firstcol, int nrows, int ncols, bool rowMajor, ENUM_MATRIX_SYMMETRY symmetry)`

A method to extract a block from a larger matrix The result is a sparse matrix object, depending on the matrixType, of constant matrix elements, variable references, linear or nonlinear expressions, or objective and constraint references (possibly mixed).

Duplicate elements are removed according to the rules formulated in the OSiL schema.

Parameters

<i>firstrow</i>	gives the first row of the block
<i>firstcol</i>	gives the first column of the block
<i>nrows</i>	gives the number of rows in the block
<i>ncols</i>	gives the number of columns in the block
<i>rowMajor</i>	can be used to store the objects in row major form.
<i>symmetry</i>	can be used to store only the upper or lower triangle, depending on the parameter value — see OSParameters.h for definitions

Returns

the block as a general sparse matrix

6.96.3.14 `bool MatrixType::isEqual (MatrixType * that)`

A function to check for the equality of two objects.

6.96.3.15 `bool MatrixType::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

Reimplemented in [MatrixBlock](#), and [OSMatrix](#).

6.96.3.16 `bool MatrixType::deepCopyFrom (MatrixType * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.96.4 Member Data Documentation

6.96.4.1 `ENUM_MATRIX_SYMMETRY MatrixType::symmetry`

To track the type of symmetry present in the matrix or block.

Remarks

for definitions, see [OSParameters.h](#)

Definition at line 1856 of file OSMatrix.h.

6.96.4.2 ENUM_MATRIX_TYPE MatrixType::type

To track the type of values present in the matrix or block.

Remarks

for definitions, see [OSParameters.h](#)

Definition at line 1862 of file OSMatrix.h.

6.96.4.3 int MatrixType::numberOfRows

Definition at line 1864 of file OSMatrix.h.

6.96.4.4 int MatrixType::numberOfColumns

Definition at line 1865 of file OSMatrix.h.

6.96.4.5 bool MatrixType::haveExpandedForm

The expanded form of the matrix is held in four sparse matrix objects: m_mmConstantElements m_mmVariableReferences m_mmGeneralElements m_mmObjAndConReferences.

to track whether the expanded form of the matrix is available

Definition at line 1878 of file OSMatrix.h.

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

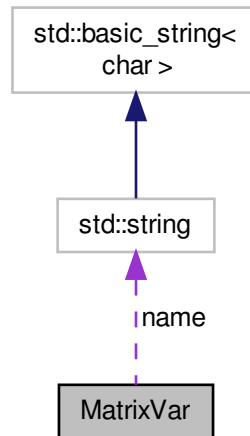
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.97 MatrixVar Class Reference

The in-memory representation of the `<matrixVar>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixVar:



Public Member Functions

- [MatrixVar](#) ()
The [MatrixVar](#) class constructor.
- [~MatrixVar](#) ()
The [MatrixVar](#) class destructor.
- `bool` [IsEqual](#) ([MatrixVar](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `int` [numberOfRows](#)
numberOfRows gives the number of rows of this matrix
- `int` [numberOfColumns](#)
numberOfColumns gives the number of columns of this matrix
- `int` [templateMatrixIdx](#)
templateMatrixIdx refers to a matrix that describes the locations in this matrixVar that are allowed to be nonzero
- `int` [varReferenceMatrixIdx](#)
varReferenceMatrixIdx allows some or all of the components of this matrix variable to be copied from variables defined in the core
- `int` [lbMatrixIdx](#)
lbMatrixIdx gives a lower bound for this matrixVar
- `int` [lbConelIdx](#)
lbConelIdx gives a cone that must contain matrixVar - lbMatrix
- `int` [ubMatrixIdx](#)
ubMatrixIdx gives an upper bound for this matrixVar

- int [ubConeldx](#)
ubConeldx gives a cone that must contain ubMatrix - matrixVar
- std::string [name](#)
an optional name to this matrixVar
- char [varType](#)
an optional variable type (C, B, I, D, J, S).

6.97.1 Detailed Description

The in-memory representation of the <**matrixVar**> element.

Definition at line 1580 of file OSInstance.h.

6.97.2 Constructor & Destructor Documentation

6.97.2.1 MatrixVar::MatrixVar ()

The [MatrixVar](#) class constructor.

6.97.2.2 MatrixVar::~~MatrixVar ()

The [MatrixVar](#) class destructor.

6.97.3 Member Function Documentation

6.97.3.1 bool MatrixVar::isEqual (**MatrixVar** * *that*)

A function to check for the equality of two objects.

6.97.4 Member Data Documentation

6.97.4.1 int MatrixVar::numberOfRows

numberOfRows gives the number of rows of this matrix

Definition at line 1584 of file OSInstance.h.

6.97.4.2 int MatrixVar::numberOfColumns

numberOfColumns gives the number of columns of this matrix

Definition at line 1587 of file OSInstance.h.

6.97.4.3 int MatrixVar::templateMatrixIdx

templateMatrixIdx refers to a matrix that describes the locations in this matrixVar that are allowed to be nonzero

Definition at line 1592 of file OSInstance.h.

6.97.4.4 int MatrixVar::varReferenceMatrixIdx

varReferenceMatrixIdx allows some or all of the components of this matrix variable to be copied from variables defined in the core

Definition at line 1597 of file OSInstance.h.

6.97.4.5 int MatrixVar::lbMatrixIdx

lbMatrixIdx gives a lower bound for this matrixVar

Definition at line 1600 of file OSInstance.h.

6.97.4.6 int MatrixVar::lbConelIdx

lbConelIdx gives a cone that must contain matrixVar - lbMatrix

Definition at line 1603 of file OSInstance.h.

6.97.4.7 int MatrixVar::ubMatrixIdx

ubMatrixIdx gives an upper bound for this matrixVar

Definition at line 1606 of file OSInstance.h.

6.97.4.8 int MatrixVar::ubConelIdx

ubConelIdx gives a cone that must contain ubMatrix - matrixVar

Definition at line 1609 of file OSInstance.h.

6.97.4.9 std::string MatrixVar::name

an optional name to this matrixVar

Definition at line 1612 of file OSInstance.h.

6.97.4.10 char MatrixVar::varType

an optional variable type (C, B, I, D, J, S).

Remarks

must be the same for each component of this matrixVar

Definition at line 1617 of file OSInstance.h.

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

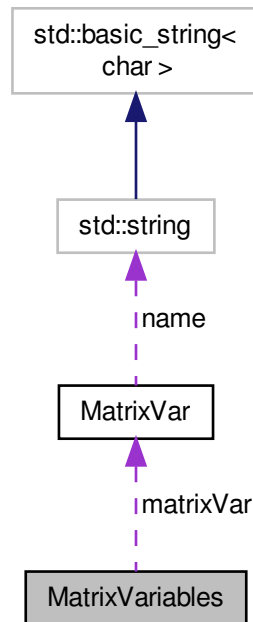
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.98 MatrixVariables Class Reference

The in-memory representation of the `<matrixVariables>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for MatrixVariables:



Public Member Functions

- [MatrixVariables](#) ()
The [MatrixVariables](#) class constructor.
- [~MatrixVariables](#) ()
The [MatrixVariables](#) class destructor.
- `bool` [IsEqual](#) ([MatrixVariables](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `int` [numberOfMatrixVar](#)
numberOfMatrixVar gives the number of `<matrixVar>` children
- [MatrixVar](#) ** [matrixVar](#)
matrixVar is an array of pointers to the `<matrixVar>` children

6.98.1 Detailed Description

The in-memory representation of the `<matrixVariables>` element.

Definition at line 1636 of file `OSInstance.h`.

6.98.2 Constructor & Destructor Documentation

6.98.2.1 MatrixVariables::MatrixVariables ()

The [MatrixVariables](#) class constructor.

6.98.2.2 MatrixVariables::~~MatrixVariables ()

The [MatrixVariables](#) class destructor.

6.98.3 Member Function Documentation

6.98.3.1 bool MatrixVariables::isEqual ([MatrixVariables](#) * *that*)

A function to check for the equality of two objects.

6.98.4 Member Data Documentation

6.98.4.1 int MatrixVariables::numberOfMatrixVar

numberOfMatrixVar gives the number of <matrixVar> children

Definition at line 1640 of file OSInstance.h.

6.98.4.2 [MatrixVar](#)** MatrixVariables::matrixVar

matrixVar is an array of pointers to the <matrixVar> children

Definition at line 1643 of file OSInstance.h.

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

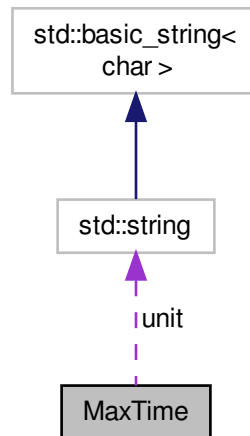
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.99 MaxTime Class Reference

the [MaxTime](#) class.

```
#include <OSOption.h>
```


Collaboration diagram for MaxTime:



Public Member Functions

- `MaxTime ()`
Default constructor.
- `~MaxTime ()`
Class destructor.
- `bool isEqual (MaxTime *that)`
A function to check for the equality of two objects.

Public Attributes

- `std::string unit`
the unit in which time is measured
- `double value`
the maximum time allowed

6.99.1 Detailed Description

the `MaxTime` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been superseded as of version 2.3 by the class [TimeSpan](#) (see [OSGeneral.h](#))

Definition at line 668 of file `OSOption.h`.

6.99.2 Constructor & Destructor Documentation**6.99.2.1 `MaxTime::MaxTime ()`**

Default constructor.

6.99.2.2 `MaxTime::~~MaxTime ()`

Class destructor.

6.99.3 Member Function Documentation**6.99.3.1 `bool MaxTime::IsEqual (MaxTime * that)`**

A function to check for the equality of two objects.

6.99.4 Member Data Documentation**6.99.4.1 `std::string MaxTime::unit`**

the unit in which time is measured

Definition at line 673 of file `OSOption.h`.

6.99.4.2 `double MaxTime::value`

the maximum time allowed

Definition at line 676 of file `OSOption.h`.

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

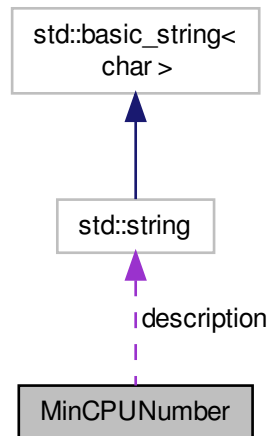
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h`

6.100 MinCPUNumber Class Reference

the [MinCPUNumber](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for MinCPUNumber:



Public Member Functions

- `MinCPUNumber ()`
Default constructor.
- `~MinCPUNumber ()`
Class destructor.
- `bool IsEqual (MinCPUNumber *that)`
A function to check for the equality of two objects.

Public Attributes

- `std::string description`
additional description about the CPU
- `int value`
the minimum number of CPUs required

6.100.1 Detailed Description

the `MinCPUNumber` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been superseded as of version 2.3 by the class [CPUNumber](#) (see [OSGeneral.h](#))

Definition at line 504 of file [OSOption.h](#).

6.100.2 Constructor & Destructor Documentation**6.100.2.1 MinCPUNumber::MinCPUNumber ()**

Default constructor.

6.100.2.2 MinCPUNumber::~~MinCPUNumber ()

Class destructor.

6.100.3 Member Function Documentation**6.100.3.1 bool MinCPUNumber::isEqual (MinCPUNumber * *that*)**

A function to check for the equality of two objects.

6.100.4 Member Data Documentation**6.100.4.1 std::string MinCPUNumber::description**

additional description about the CPU

Definition at line 509 of file [OSOption.h](#).

6.100.4.2 int MinCPUNumber::value

the minimum number of CPUs required

Definition at line 512 of file [OSOption.h](#).

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

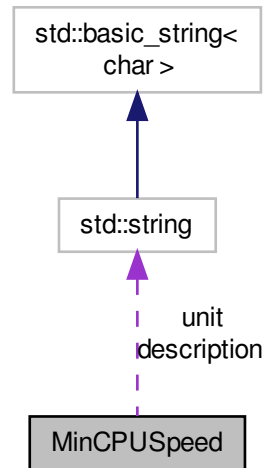
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.101 MinCPUSpeed Class Reference

the [MinCPUSpeed](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for MinCPUSpeed:



Public Member Functions

- `MinCPUSpeed ()`
Default constructor.
- `~MinCPUSpeed ()`
Class destructor.
- `bool IsEqual (MinCPUSpeed *that)`
A function to check for the equality of two objects.

Public Attributes

- `std::string unit`
the unit in which CPU speed is measured
- `std::string description`
additional description about the CPU speed
- `double value`
the minimum CPU speed required

6.101.1 Detailed Description

the `MinCPUSpeed` class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been superseded as of version 2.3 by the class [CPUSpeed](#) (see [OSGeneral.h](#))

Definition at line 459 of file OSOption.h.

6.101.2 Constructor & Destructor Documentation**6.101.2.1 MinCPUSpeed::MinCPUSpeed ()**

Default constructor.

6.101.2.2 MinCPUSpeed::~MinCPUSpeed ()

Class destructor.

6.101.3 Member Function Documentation**6.101.3.1 bool MinCPUSpeed::isEqual (MinCPUSpeed * that)**

A function to check for the equality of two objects.

6.101.4 Member Data Documentation**6.101.4.1 std::string MinCPUSpeed::unit**

the unit in which CPU speed is measured

Definition at line 464 of file OSOption.h.

6.101.4.2 std::string MinCPUSpeed::description

additional description about the CPU speed

Definition at line 467 of file OSOption.h.

6.101.4.3 double MinCPUSpeed::value

the minimum CPU speed required

Definition at line 470 of file OSOption.h.

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

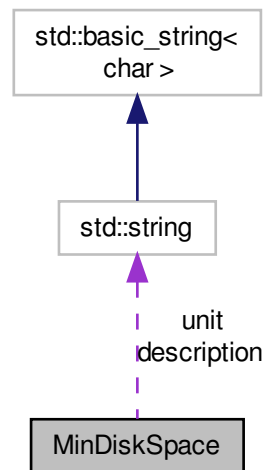
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.102 MinDiskSpace Class Reference

the [MinDiskSpace](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for MinDiskSpace:



Public Member Functions

- [MinDiskSpace](#) ()
Default constructor.
- [~MinDiskSpace](#) ()
Class destructor.
- `bool` [IsEqual](#) ([MinDiskSpace](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `std::string` [unit](#)
the unit in which disk space is measured
- `std::string` [description](#)
additional description about the disk space
- `double` [value](#)
the minimum disk space required

6.102.1 Detailed Description

the [MinDiskSpace](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been superseded as of version 2.3 by the class [StorageCapacity](#) (see [OSGeneral.h](#))

Definition at line 369 of file `OSOption.h`.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 `MinDiskSpace::MinDiskSpace ()`

Default constructor.

6.102.2.2 `MinDiskSpace::~~MinDiskSpace ()`

Class destructor.

6.102.3 Member Function Documentation

6.102.3.1 `bool MinDiskSpace::isEqual (MinDiskSpace * that)`

A function to check for the equality of two objects.

6.102.4 Member Data Documentation

6.102.4.1 `std::string MinDiskSpace::unit`

the unit in which disk space is measured

Definition at line 374 of file `OSOption.h`.

6.102.4.2 `std::string MinDiskSpace::description`

additional description about the disk space

Definition at line 377 of file `OSOption.h`.

6.102.4.3 double MinDiskSpace::value

the minimum disk space required

Definition at line 380 of file OOption.h.

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

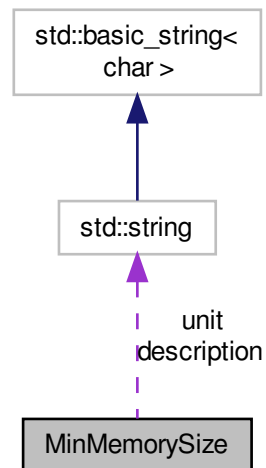
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OOption.h>

6.103 MinMemorySize Class Reference

the [MinMemorySize](#) class.

```
#include <OOption.h>
```

Collaboration diagram for MinMemorySize:



Public Member Functions

- [MinMemorySize](#) ()
Default constructor.
- [~MinMemorySize](#) ()
Class destructor.
- `bool` [IsEqual](#) ([MinMemorySize](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `std::string` [unit](#)

- the unit in which memory size is measured*
 - `std::string` [description](#)
 - additional description about the memory*
 - `double` [value](#)
 - the minimum memory size required*

6.103.1 Detailed Description

the [MinMemorySize](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema. This class has been superseded as of version 2.3 by the class [StorageCapacity](#) (see [OSGeneral.h](#))

Definition at line 414 of file `OSOption.h`.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 `MinMemorySize::MinMemorySize ()`

Default constructor.

6.103.2.2 `MinMemorySize::~~MinMemorySize ()`

Class destructor.

6.103.3 Member Function Documentation

6.103.3.1 `bool MinMemorySize::isEqual (MinMemorySize * that)`

A function to check for the equality of two objects.

6.103.4 Member Data Documentation

6.103.4.1 `std::string MinMemorySize::unit`

the unit in which memory size is measured

Definition at line 419 of file `OSOption.h`.

6.103.4.2 `std::string MinMemorySize::description`

additional description about the memory

Definition at line 422 of file `OSOption.h`.

6.103.4.3 `double MinMemorySize::value`

the minimum memory size required

Definition at line 425 of file `OSOption.h`.

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

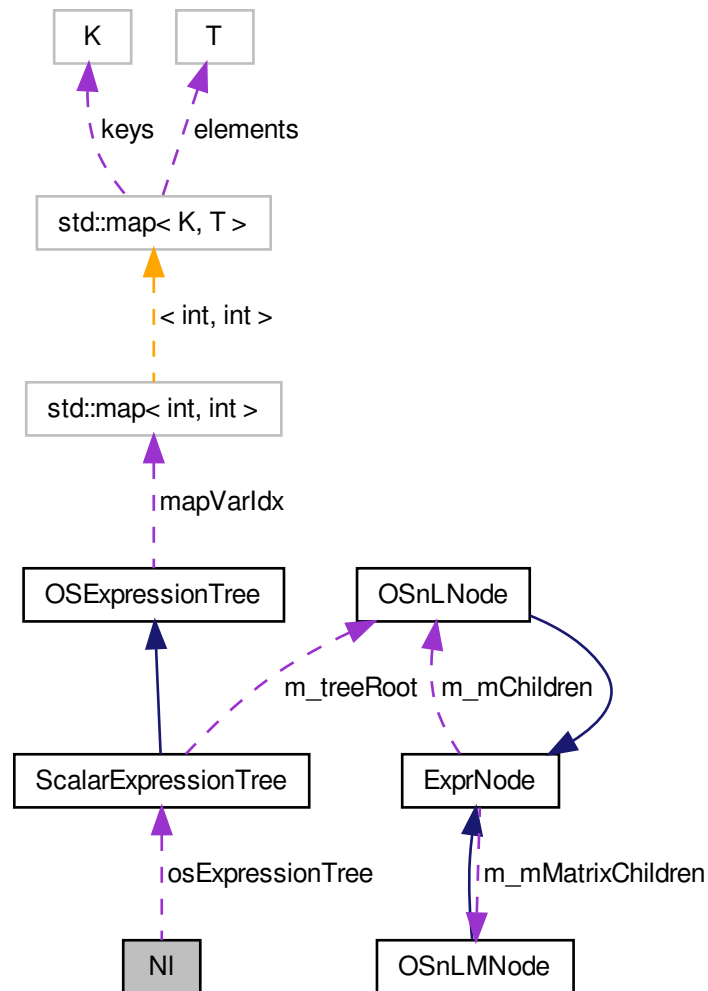
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h`

6.104 NI Class Reference

The in-memory representation of the `<nl>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for NI:



Public Member Functions

- `NI()`
default constructor.
- `~NI()`
default destructor.
- `bool IsEqual (NI *that)`
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
idx holds the row index of the nonlinear expression
- [ENUM_NL_EXPR_SHAPE](#) [shape](#)
shape holds the shape of the nonlinear expression (linear/quadratic/convex/general) (see further up in this file).
- bool [m_bDeleteExpressionTree](#)
m_bDeleteExpressionTree is true, if in garbage collection, we should delete the osExpression tree object, if the [OSInstance](#) class created a map of the expression trees this should be false since the osExpressionTree is deleted by the [OSInstance](#) object
- [ScalarExpressionTree](#) * [osExpressionTree](#)
osExpressionTree contains the root of the [ScalarExpressionTree](#)

6.104.1 Detailed Description

The in-memory representation of the <nl> element.

Definition at line 410 of file OSInstance.h.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 [NI::NI \(\)](#)

default constructor.

6.104.2.2 [NI::~~NI \(\)](#)

default destructor.

6.104.3 Member Function Documentation

6.104.3.1 [bool NI::isEqual \(NI * that \)](#)

A function to check for the equality of two objects.

6.104.4 Member Data Documentation

6.104.4.1 [int NI::idx](#)

idx holds the row index of the nonlinear expression

Definition at line 414 of file OSInstance.h.

6.104.4.2 [ENUM_NL_EXPR_SHAPE NI::shape](#)

shape holds the shape of the nonlinear expression (linear/quadratic/convex/general) (see further up in this file).

this might be useful in guiding solver selection.

Definition at line 420 of file OSInstance.h.

6.104.4.3 bool NI::m_bDeleteExpressionTree

m_bDeleteExpressionTree is true, if in garbage collection, we should delete the osExpression tree object, if the [OSInstance](#) class created a map of the expression trees this should be false since the osExpressionTree is deleted by the [OSInstance](#) object

Definition at line 427 of file OSInstance.h.

6.104.4.4 ScalarExpressionTree* NI::osExpressionTree

osExpressionTree contains the root of the [ScalarExpressionTree](#)

Definition at line 430 of file OSInstance.h.

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

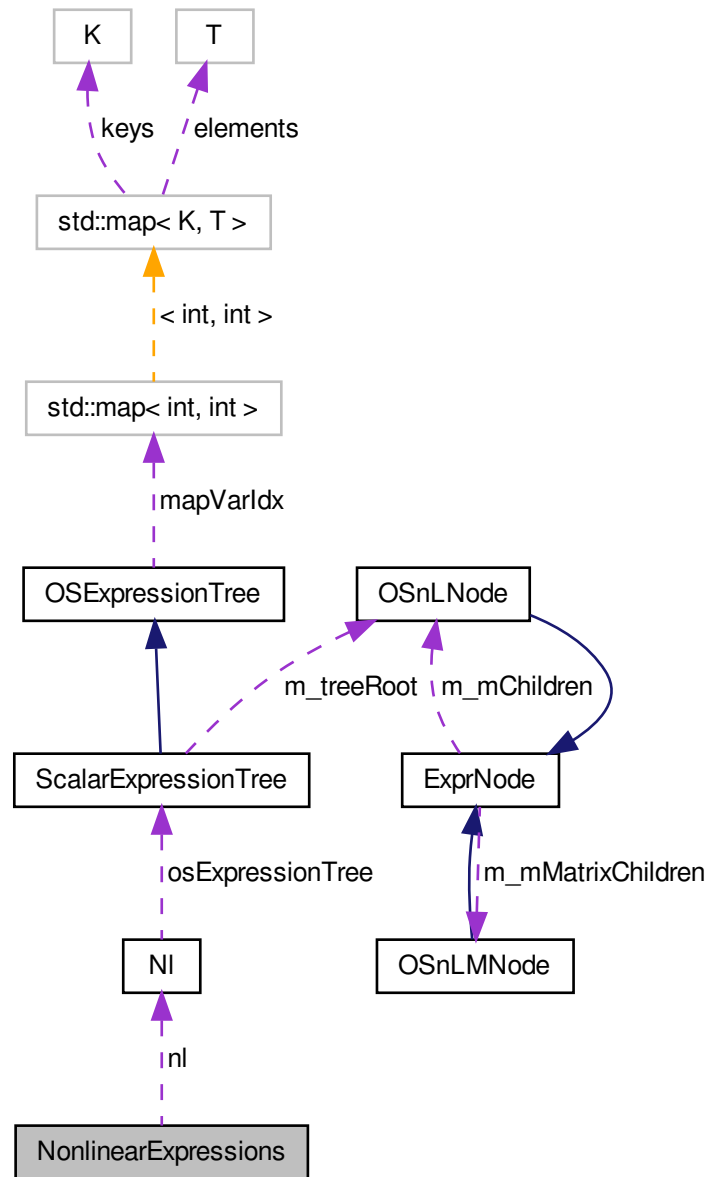
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.105 NonlinearExpressions Class Reference

The in-memory representation of the `<nonlinearExpressions>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for NonlinearExpressions:



Public Member Functions

- [NonlinearExpressions](#) ()
The *NonlinearExpressions* class constructor.
- [~NonlinearExpressions](#) ()
The *NonlinearExpressions* class destructor.

- bool [IsEqual](#) ([NonlinearExpressions](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfNonlinearExpressions](#)
*numberOfNonlinearExpressions is the number of <nl> elements in the <**nonlinearExpressions**> element.*
- [NI](#) ** nl
nl is pointer to an array of [NI](#) object pointers

6.105.1 Detailed Description

The in-memory representation of the <**nonlinearExpressions**> element.
Definition at line 452 of file OSInstance.h.

6.105.2 Constructor & Destructor Documentation

6.105.2.1 NonlinearExpressions::NonlinearExpressions ()

The [NonlinearExpressions](#) class constructor.

6.105.2.2 NonlinearExpressions::~~NonlinearExpressions ()

The [NonlinearExpressions](#) class destructor.

6.105.3 Member Function Documentation

6.105.3.1 bool NonlinearExpressions::IsEqual ([NonlinearExpressions](#) * that)

A function to check for the equality of two objects.

6.105.4 Member Data Documentation

6.105.4.1 int NonlinearExpressions::numberOfNonlinearExpressions

numberOfNonlinearExpressions is the number of <nl> elements in the <**nonlinearExpressions**> element.
Definition at line 466 of file OSInstance.h.

6.105.4.2 [NI](#)** NonlinearExpressions::nl

nl is pointer to an array of [NI](#) object pointers

Definition at line 469 of file OSInstance.h.

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

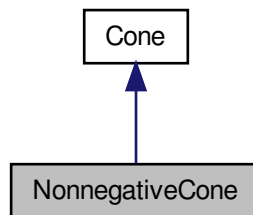
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.106 NonnegativeCone Class Reference

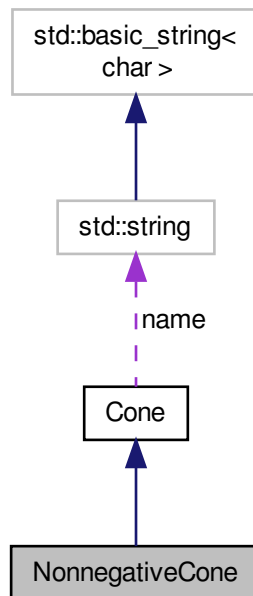
The [NonnegativeCone](#) Class.

```
#include <OSInstance.h>
```

Inheritance diagram for NonnegativeCone:



Collaboration diagram for NonnegativeCone:



Public Member Functions

- [NonnegativeCone](#) ()
default constructor.
- [~NonnegativeCone](#) ()
default destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [NonnegativeCone](#) object in XML format.
- bool [IsEqual](#) ([NonnegativeCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([NonnegativeCone](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

6.106.1 Detailed Description

The [NonnegativeCone](#) Class.

Remarks

The in-memory representation of the OSiL element <nonnegativeCone>

Definition at line 609 of file OSInstance.h.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 [NonnegativeCone::NonnegativeCone](#) ()

default constructor.

6.106.2.2 [NonnegativeCone::~~NonnegativeCone](#) ()

default destructor.

6.106.3 Member Function Documentation

6.106.3.1 virtual std::string [NonnegativeCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.106.3.2 `virtual std::string NonnegativeCone::getConeInXML () [virtual]`

Write a [NonnegativeCone](#) object in XML format.

This is used by [OSILWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.106.3.3 `bool NonnegativeCone::isEqual (NonnegativeCone * that)`

A function to check for the equality of two objects.

6.106.3.4 `bool NonnegativeCone::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.106.3.5 `bool NonnegativeCone::deepCopyFrom (NonnegativeCone * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

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

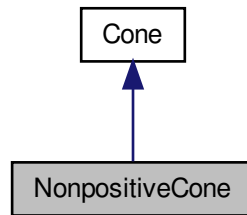
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.107 NonpositiveCone Class Reference

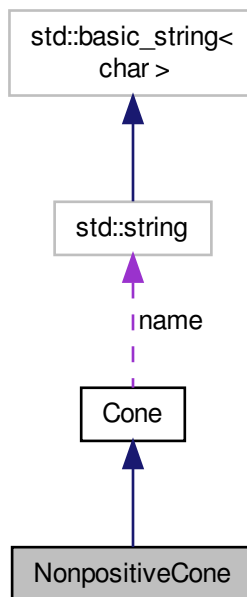
The [NonpositiveCone](#) Class.

```
#include <OSInstance.h>
```

Inheritance diagram for NonpositiveCone:



Collaboration diagram for NonpositiveCone:



Public Member Functions

- [NonpositiveCone](#) ()
default constructor.
- [~NonpositiveCone](#) ()
default destructor.

- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [NonpositiveCone](#) object in XML format.
- bool [IsEqual](#) ([NonpositiveCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([NonpositiveCone](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

6.107.1 Detailed Description

The [NonpositiveCone](#) Class.

Remarks

The in-memory representation of the OSiL element <nonpositiveCone>

Definition at line 667 of file OSInstance.h.

6.107.2 Constructor & Destructor Documentation

6.107.2.1 NonpositiveCone::NonpositiveCone ()

default constructor.

6.107.2.2 NonpositiveCone::~~NonpositiveCone ()

default destructor.

6.107.3 Member Function Documentation

6.107.3.1 virtual std::string NonpositiveCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.107.3.2 virtual std::string NonpositiveCone::getConeInXML () [virtual]

Write a [NonpositiveCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.107.3.3 bool NonpositiveCone::isEqual (NonpositiveCone * that)

A function to check for the equality of two objects.

6.107.3.4 bool NonpositiveCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.107.3.5 bool NonpositiveCone::deepCopyFrom (NonpositiveCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.108 ObjCoef Class Reference

The in-memory representation of the objective function <coef> element.

```
#include <OSInstance.h>
```

Public Member Functions

- [ObjCoef](#) ()
The *ObjCoef* class constructor.
- [~ObjCoef](#) ()
The *ObjCoef* class destructor.
- bool [isEqual](#) (ObjCoef *that)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)

idx is the index of the variable corresponding to the coefficient

- double [value](#)

value is the value of the objective function coefficient corresponding to the variable with index idx

6.108.1 Detailed Description

The in-memory representation of the objective function `<coef>` element.

Definition at line 110 of file `OSInstance.h`.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 `ObjCoef::ObjCoef ()`

The [ObjCoef](#) class constructor.

6.108.2.2 `ObjCoef::~~ObjCoef ()`

The [ObjCoef](#) class destructor.

6.108.3 Member Function Documentation

6.108.3.1 `bool ObjCoef::isEqual (ObjCoef * that)`

A function to check for the equality of two objects.

6.108.4 Member Data Documentation

6.108.4.1 `int ObjCoef::idx`

`idx` is the index of the variable corresponding to the coefficient

Definition at line 123 of file `OSInstance.h`.

6.108.4.2 `double ObjCoef::value`

`value` is the value of the objective function coefficient corresponding to the variable with index `idx`

Definition at line 128 of file `OSInstance.h`.

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

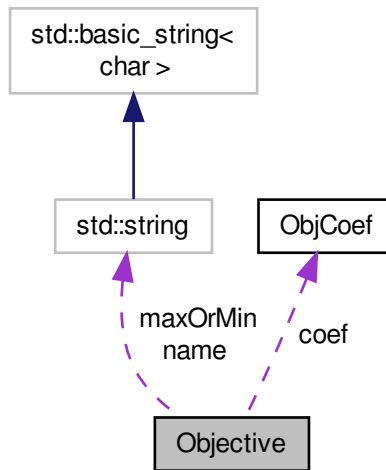
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h`

6.109 Objective Class Reference

The in-memory representation of the `<obj>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for Objective:



Public Member Functions

- **Objective** ()
The *Objective* class constructor.
- **~Objective** ()
The *Objective* class destructor.
- **bool IsEqual (Objective *that)**
A function to check for the equality of two objects.

Public Attributes

- **std::string name**
the name of the objective function
- **std::string maxOrMin**
declare the objective function to be a max or a min
- **double constant**
constant is the constant term added to the objective function, 0 by default
- **double weight**
weight is the weight applied to the given objective function, 1.0 by default
- **int numberOfObjCoef**
numberOfObjCoef is the number of variables with a nonzero objective function coefficient
- **ObjCoef ** coef**
coef is pointer to an array of *ObjCoef* object pointers

6.109.1 Detailed Description

The in-memory representation of the `<obj>` element.

Definition at line 141 of file `OSInstance.h`.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 `Objective::Objective ()`

The `Objective` class constructor.

6.109.2.2 `Objective::~~Objective ()`

The `Objective` class destructor.

6.109.3 Member Function Documentation

6.109.3.1 `bool Objective::IsEqual (Objective * that)`

A function to check for the equality of two objects.

6.109.4 Member Data Documentation

6.109.4.1 `std::string Objective::name`

the name of the objective function

Definition at line 152 of file `OSInstance.h`.

6.109.4.2 `std::string Objective::maxOrMin`

declare the objective function to be a max or a min

Definition at line 157 of file `OSInstance.h`.

6.109.4.3 `double Objective::constant`

constant is the constant term added to the objective function, 0 by default

Definition at line 162 of file `OSInstance.h`.

6.109.4.4 `double Objective::weight`

weight is the weight applied to the given objective function, 1.0 by default

Definition at line 167 of file `OSInstance.h`.

6.109.4.5 `int Objective::numberOfObjCoef`

`numberOfObjCoef` is the number of variables with a nonzero objective function coefficient

Definition at line 172 of file `OSInstance.h`.

6.109.4.6 `ObjCoef** Objective::coef`

`coef` is pointer to an array of `ObjCoef` object pointers

Definition at line 176 of file OSInstance.h.

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

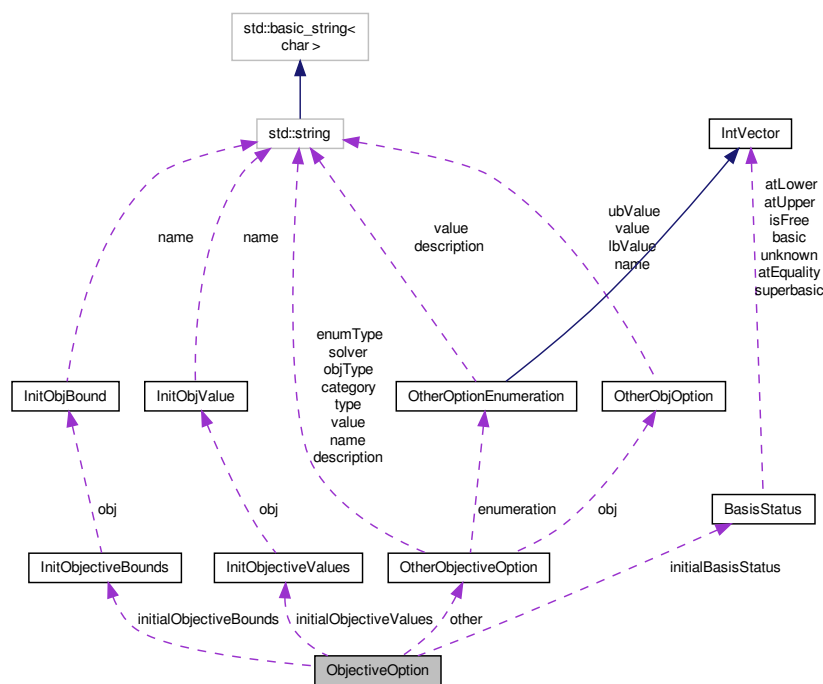
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h

6.110 ObjectiveOption Class Reference

the [ObjectiveOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for ObjectiveOption:



Public Member Functions

- [ObjectiveOption](#) ()
Default constructor.
- [~ObjectiveOption](#) ()
Class destructor.
- `bool` [IsEqual](#) ([ObjectiveOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([ObjectiveOption](#) *that)
A function to make a deep copy of an instance of this class.

- bool [setOther](#) (int numberOfOptions, [OtherObjectiveOption](#) **other)
A function to set an array of <other> elements.
- bool [addOther](#) ([OtherObjectiveOption](#) *other)
A function to add an <other> element.

Public Attributes

- int [numberOfOtherObjectiveOptions](#)
number of <other> child elements
- [InitObjectiveValues](#) * [initialObjectiveValues](#)
initial values for the objectives
- [InitObjectiveBounds](#) * [initialObjectiveBounds](#)
initial bounds for the objectives
- [BasisStatus](#) * [initialBasisStatus](#)
initial basis status for the objectives
- [OtherObjectiveOption](#) ** [other](#)
other information about the objectives

6.110.1 Detailed Description

the [ObjectiveOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2681 of file OSOption.h.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 [ObjectiveOption::ObjectiveOption \(\)](#)

Default constructor.

6.110.2.2 [ObjectiveOption::~~ObjectiveOption \(\)](#)

Class destructor.

6.110.3 Member Function Documentation

6.110.3.1 bool ObjectiveOption::isEqual (ObjectiveOption * *that*)

A function to check for the equality of two objects.

6.110.3.2 bool ObjectiveOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.110.3.3 bool ObjectiveOption::deepCopyFrom (ObjectiveOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.110.3.4 bool ObjectiveOption::setOther (int *numberOfOptions*, OtherObjectiveOption ** *other*)

A function to set an array of <other> elements.

Parameters

<i>numberOfOptions</i> ,:	number of <other> elements to be set
<i>other</i> ,:	the array of <other> elements that are to be set

6.110.3.5 bool ObjectiveOption::addOther (OtherObjectiveOption * *other*)

A function to add an <other> element.

Parameters

<i>other</i> ,:	the content of the <other> element to be added
-----------------	--

6.110.4 Member Data Documentation

6.110.4.1 int ObjectiveOption::numberOfOtherObjectiveOptions

number of <other> child elements

Definition at line 2686 of file OSOption.h.

6.110.4.2 InitObjectiveValues* ObjectiveOption::initialObjectiveValues

initial values for the objectives

Definition at line 2689 of file OSOption.h.

6.110.4.3 InitObjectiveBounds* ObjectiveOption::initialObjectiveBounds

initial bounds for the objectives

Definition at line 2692 of file OSOption.h.

6.110.4.4 BasisStatus* ObjectiveOption::initialBasisStatus

initial basis status for the objectives

Definition at line 2695 of file OSOption.h.

6.110.4.5 OtherObjectiveOption ObjectiveOption::other**

other information about the objectives

Definition at line 2698 of file OSOption.h.

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

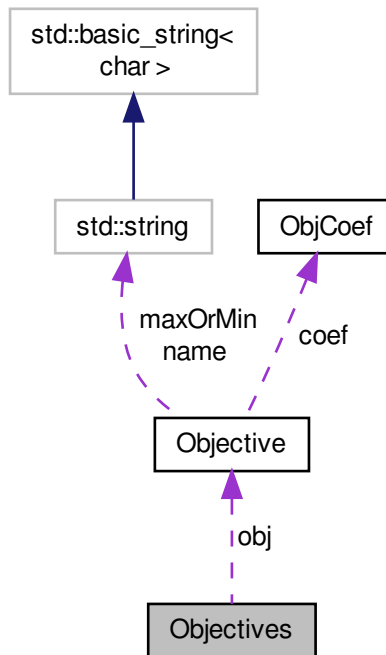
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.111 Objectives Class Reference

The in-memory representation of the **<objectives>** element.

```
#include <OSInstance.h>
```

Collaboration diagram for Objectives:



Public Member Functions

- [Objectives](#) ()
The [Objectives](#) class constructor.
- [~Objectives](#) ()
The [Objectives](#) class destructor.
- bool [IsEqual](#) ([Objectives](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfObjectives](#)
numberOfObjectives is the number of objective functions in the instance
- [Objective](#) ** [obj](#)
coef is pointer to an array of [ObjCoef](#) object pointers

6.111.1 Detailed Description

The in-memory representation of the **<objectives>** element.

Definition at line 188 of file OSInstance.h.

6.111.2 Constructor & Destructor Documentation

6.111.2.1 Objectives::Objectives ()

The [Objectives](#) class constructor.

6.111.2.2 Objectives::~~Objectives ()

The [Objectives](#) class destructor.

6.111.3 Member Function Documentation

6.111.3.1 bool Objectives::isEqual ([Objectives](#) * *that*)

A function to check for the equality of two objects.

6.111.4 Member Data Documentation

6.111.4.1 int Objectives::numberOfObjectives

numberOfObjectives is the number of objective functions in the instance
Definition at line 201 of file OSInstance.h.

6.111.4.2 [Objective](#)** Objectives::obj

coef is pointer to an array of [ObjCoef](#) object pointers

Definition at line 205 of file OSInstance.h.

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

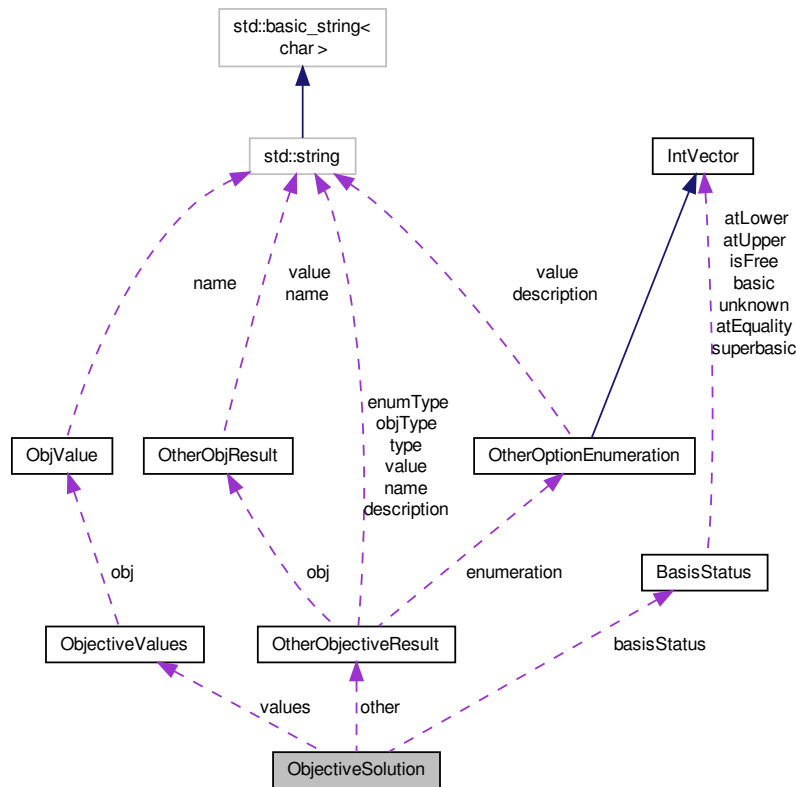
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.112 ObjectiveSolution Class Reference

The [ObjectiveSolution](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for ObjectiveSolution:



Public Member Functions

- `ObjectiveSolution ()`
Default constructor.
- `~ObjectiveSolution ()`
Class destructor.
- `bool IsEqual (ObjectiveSolution *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfOtherObjectiveResults`
the number of types of objective function results other than the basic objective function values
- `ObjectiveValues * values`
a pointer to an array of `ObjectiveValues` objects
- `BasisStatus * basisStatus`

a pointer to a [BasisStatus](#) object

- [OtherObjectiveResult](#) ** [other](#)

a pointer to an array of other pointer objects for objective functions

6.112.1 Detailed Description

The [ObjectiveSolution](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting all of the types of solution values associated with objective functions.

Definition at line 1523 of file OSResult.h.

6.112.2 Constructor & Destructor Documentation

6.112.2.1 ObjectiveSolution::ObjectiveSolution ()

Default constructor.

6.112.2.2 ObjectiveSolution::~~ObjectiveSolution ()

Class destructor.

6.112.3 Member Function Documentation

6.112.3.1 bool ObjectiveSolution::isEqual (ObjectiveSolution * *that*)

A function to check for the equality of two objects.

6.112.3.2 bool ObjectiveSolution::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.112.4 Member Data Documentation

6.112.4.1 `int ObjectiveSolution::numberOfOtherObjectiveResults`

the number of types of objective function results other than the basic objective function values

Definition at line 1530 of file OSResult.h.

6.112.4.2 `ObjectiveValues* ObjectiveSolution::values`

a pointer to an array of [ObjectiveValues](#) objects

Definition at line 1533 of file OSResult.h.

6.112.4.3 `BasisStatus* ObjectiveSolution::basisStatus`

a pointer to a [BasisStatus](#) object

Definition at line 1536 of file OSResult.h.

6.112.4.4 `OtherObjectiveResult** ObjectiveSolution::other`

a pointer to an array of other pointer objects for objective functions

Definition at line 1541 of file OSResult.h.

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

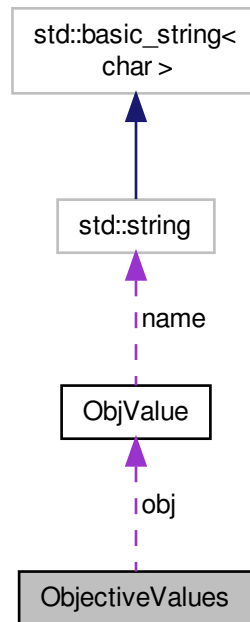
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.113 ObjectiveValues Class Reference

The [ObjectiveValues](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for ObjectiveValues:



Public Member Functions

- `ObjectiveValues ()`
Default constructor.
- `~ObjectiveValues ()`
Class destructor.
- `bool IsEqual (ObjectiveValues *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfObj`
record the number of objective rows for which values are given
- `ObjValue ** obj`
obj is a pointer to an array of `ObjValue` objects that give an index and objective function value for each objective function.

6.113.1 Detailed Description

The [ObjectiveValues](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting objective function values

Definition at line 1324 of file OSResult.h.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 ObjectiveValues::ObjectiveValues ()

Default constructor.

6.113.2.2 ObjectiveValues::~~ObjectiveValues ()

Class destructor.

6.113.3 Member Function Documentation

6.113.3.1 bool ObjectiveValues::isEqual (ObjectiveValues * *that*)

A function to check for the equality of two objects.

6.113.3.2 bool ObjectiveValues::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.113.4 Member Data Documentation

6.113.4.1 int ObjectiveValues::numberOfObj

record the number of objective rows for which values are given

Definition at line 1330 of file OSResult.h.

6.113.4.2 ObjValue** ObjectiveValues::obj

obj is a pointer to an array of [ObjValue](#) objects that give an index and objective function value for each objective function.

Definition at line 1336 of file OSResult.h.

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

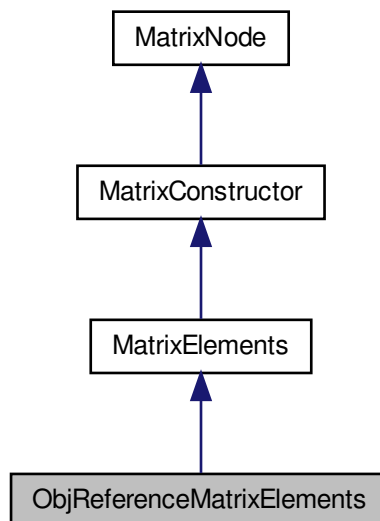
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.114 ObjReferenceMatrixElements Class Reference

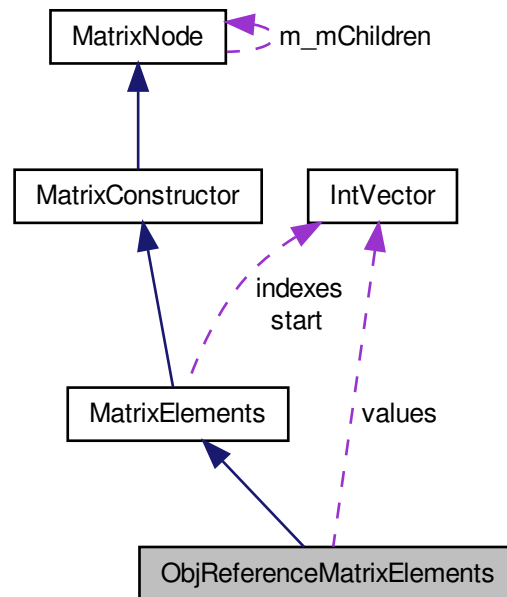
a data structure to represent objective reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of an objective (i.e., less than zero)

```
#include <OSMatrix.h>
```

Inheritance diagram for ObjReferenceMatrixElements:



Collaboration diagram for ObjReferenceMatrixElements:



Public Member Functions

- [ObjReferenceMatrixElements](#) ()
- [~ObjReferenceMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNode](#)Type ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrix](#)Type ()
- virtual std::string [getNode](#)Name ()
- virtual std::string [getMatrixNode](#)InXML ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [ObjReferenceMatrixElements](#) * [cloneMatrixNode](#) ()
- bool [isEqual](#) ([ObjReferenceMatrixElements](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([ObjReferenceMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [IntVector](#) * `values`

The objective references (indexes of core objectives) of the elements.

6.114.1 Detailed Description

a data structure to represent objective reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of an objective (i.e., less than zero)

Definition at line 1092 of file OSMatrix.h.

6.114.2 Constructor & Destructor Documentation

6.114.2.1 `ObjReferenceMatrixElements::ObjReferenceMatrixElements ()`

6.114.2.2 `ObjReferenceMatrixElements::~~ObjReferenceMatrixElements ()`

6.114.3 Member Function Documentation

6.114.3.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE ObjReferenceMatrixElements::getNodeType ()` [virtual]

Returns

the value of `nType`

Reimplemented from [MatrixNode](#).

6.114.3.2 `virtual ENUM_MATRIX_TYPE ObjReferenceMatrixElements::getMatrixType ()` [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.114.3.3 `virtual std::string ObjReferenceMatrixElements::getNodeName ()` [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.114.3.4 `virtual std::string ObjReferenceMatrixElements::getMatrixNodeInXML ()` [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a `<matrix>` element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.114.3.5 `virtual bool ObjReferenceMatrixElements::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols)`
`[virtual]`

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.114.3.6 `virtual ObjReferenceMatrixElements* ObjReferenceMatrixElements::cloneMatrixNode ()` `[virtual]`

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.114.3.7 `bool ObjReferenceMatrixElements::isEqual (ObjReferenceMatrixElements * that)`

A function to check for the equality of two objects.

6.114.3.8 `bool ObjReferenceMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.114.3.9 `bool ObjReferenceMatrixElements::deepCopyFrom (ObjReferenceMatrixElements * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.114.4 Member Data Documentation

6.114.4.1 `IntVector* ObjReferenceMatrixElements::values`

The objective references (indexes of core objectives) of the elements.

Definition at line 1096 of file `OSMatrix.h`.

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

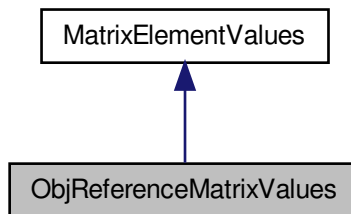
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h`

6.115 ObjReferenceMatrixValues Class Reference

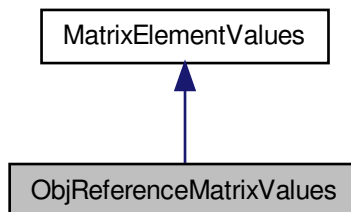
to represent the nonzeros in an `objReferenceMatrix` element

```
#include <OSMatrix.h>
```

Inheritance diagram for `ObjReferenceMatrixValues`:



Collaboration diagram for `ObjReferenceMatrixValues`:



Public Member Functions

- [ObjReferenceMatrixValues](#) ()
- [~ObjReferenceMatrixValues](#) ()
- bool [isEqual](#) ([ObjReferenceMatrixValues](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([ObjReferenceMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int * [el](#)

6.115.1 Detailed Description

to represent the nonzeros in an objReferenceMatrix element

Definition at line 676 of file OSMatrix.h.

6.115.2 Constructor & Destructor Documentation

6.115.2.1 [ObjReferenceMatrixValues::ObjReferenceMatrixValues](#) ()

6.115.2.2 [ObjReferenceMatrixValues::~~ObjReferenceMatrixValues](#) ()

6.115.3 Member Function Documentation

6.115.3.1 bool [ObjReferenceMatrixValues::isEqual](#) ([ObjReferenceMatrixValues](#) * that)

A function to check for the equality of two objects.

6.115.3.2 bool [ObjReferenceMatrixValues::setRandom](#) (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.115.3.3 bool [ObjReferenceMatrixValues::deepCopyFrom](#) ([ObjReferenceMatrixValues](#) * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.115.4 Member Data Documentation

6.115.4.1 int* ObjReferenceMatrixValues::el

Definition at line 679 of file OSMatrix.h.

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

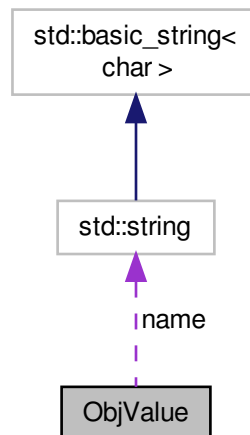
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h

6.116 ObjValue Class Reference

The [ObjValue](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for ObjValue:



Public Member Functions

- [ObjValue](#) ()
Default constructor.
- [~ObjValue](#) ()
Class destructor.
- bool [IsEqual](#) ([ObjValue](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int `idx`
idx is the index on an objective function
- std::string `name`
optional name
- double `value`
the value of the objective indexed by idx

6.116.1 Detailed Description

The `ObjValue` Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the value of an objective function

Definition at line 1273 of file OSResult.h.

6.116.2 Constructor & Destructor Documentation

6.116.2.1 `ObjValue::ObjValue ()`

Default constructor.

6.116.2.2 `ObjValue::~~ObjValue ()`

Class destructor.

6.116.3 Member Function Documentation

6.116.3.1 `bool ObjValue::isEqual (ObjValue * that)`

A function to check for the equality of two objects.

6.116.3.2 `bool ObjValue::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.116.4 Member Data Documentation

6.116.4.1 int ObjValue::idx

idx is the index on an objective function

Definition at line 1278 of file OSResult.h.

6.116.4.2 std::string ObjValue::name

optional name

Definition at line 1281 of file OSResult.h.

6.116.4.3 double ObjValue::value

the value of the objective indexed by idx

Definition at line 1284 of file OSResult.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.117 OptimizationOption Class Reference

the [OptimizationOption](#) class.

```
#include <OSOption.h>
```

- `OptimizationOption ()`
Default constructor.
- `~OptimizationOption ()`
Class destructor.
- `bool isEqual (OptimizationOption *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (OptimizationOption *that)`
A function to make a deep copy of an instance of this class.

- int **numberOfVariables**
the number of variables
- int **numberOfObjectives**
the number of objectives
- int **numberOfConstraints**
the number of constraints
- **VariableOption** * **variables**
the options for the variables
- **ObjectiveOption** * **objectives**

the options for the objectives

- [ConstraintOption](#) * *constraints*

the options for the constraints

- [SolverOptions](#) * *solverOptions*

other solver options

6.117.1 Detailed Description

the [OptimizationOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3495 of file OSOption.h.

6.117.2 Constructor & Destructor Documentation

6.117.2.1 OptimizationOption::OptimizationOption ()

Default constructor.

6.117.2.2 OptimizationOption::~~OptimizationOption ()

Class destructor.

6.117.3 Member Function Documentation

6.117.3.1 bool OptimizationOption::isEqual (OptimizationOption * *that*)

A function to check for the equality of two objects.

6.117.3.2 bool OptimizationOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.117.3.3 bool OptimizationOption::deepCopyFrom (OptimizationOption * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.117.4 Member Data Documentation**6.117.4.1 int OptimizationOption::numberOfVariables**

the number of variables

Definition at line 3500 of file OSOption.h.

6.117.4.2 int OptimizationOption::numberOfObjectives

the number of objectives

Definition at line 3503 of file OSOption.h.

6.117.4.3 int OptimizationOption::numberOfConstraints

the number of constraints

Definition at line 3506 of file OSOption.h.

6.117.4.4 VariableOption* OptimizationOption::variables

the options for the variables

Definition at line 3509 of file OSOption.h.

6.117.4.5 ObjectiveOption* OptimizationOption::objectives

the options for the objectives

Definition at line 3512 of file OSOption.h.

6.117.4.6 ConstraintOption* OptimizationOption::constraints

the options for the constraints

Definition at line 3515 of file OSOption.h.

6.117.4.7 SolverOptions* OptimizationOption::solverOptions

other solver options

Definition at line 3518 of file OSOption.h.

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

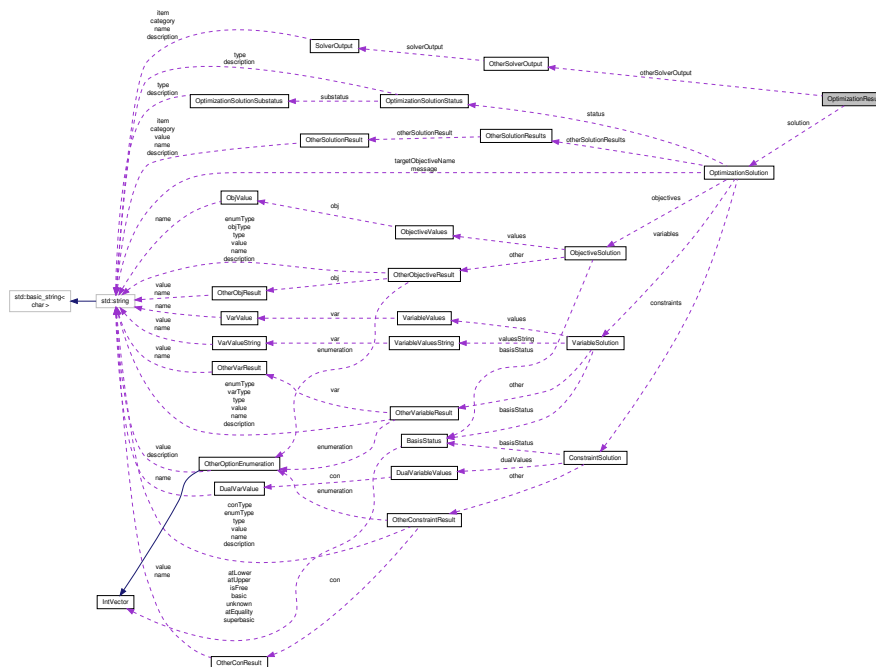
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.118 OptimizationResult Class Reference

The [OptimizationResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OptimizationResult:



Public Member Functions

- [OptimizationResult](#) ()
Default constructor.
- [~OptimizationResult](#) ()
Class destructor.
- bool [isEqual](#) ([OptimizationResult](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [numberOfSolutions](#)
numberOfSolutions is the number of objective functions reported.
- int [numberOfVariables](#)
numberOfVariables is the number of variables reported in the solution.
- int [numberOfObjectives](#)
numberOfObjectives is the number of objective functions reported in the solution.

- int [numberOfConstraints](#)
numberOfConstraints is the number of constraint functions reported in the solution.
- [OptimizationSolution](#) ** [solution](#)
solution is an array of pointers to [OptimizationSolution](#) objects
- [OtherSolverOutput](#) * [otherSolverOutput](#)
otherSolverOutput is a pointer to an [OtherSolverOutput](#) object

6.118.1 Detailed Description

The [OptimizationResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for holding information that might be associated with an optimization solution.

Definition at line 2237 of file OSResult.h.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 OptimizationResult::OptimizationResult ()

Default constructor.

6.118.2.2 OptimizationResult::~~OptimizationResult ()

Class destructor.

6.118.3 Member Function Documentation

6.118.3.1 bool OptimizationResult::isEqual ([OptimizationResult](#) * *that*)

A function to check for the equality of two objects.

6.118.3.2 bool OptimizationResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.118.4 Member Data Documentation

6.118.4.1 int OptimizationResult::numberOfSolutions

numberOfSolutions is the number of objective functions reported.

Definition at line 2244 of file OSResult.h.

6.118.4.2 int OptimizationResult::numberOfVariables

numberOfVariables is the number of variables reported in the solution.

Definition at line 2249 of file OSResult.h.

6.118.4.3 int OptimizationResult::numberOfObjectives

numberOfObjectives is the number of objective functions reported in the solution.

Definition at line 2254 of file OSResult.h.

6.118.4.4 int OptimizationResult::numberOfConstraints

numberOfConstraints is the number of constraint functions reported in the solution.

Definition at line 2259 of file OSResult.h.

6.118.4.5 OptimizationSolution** OptimizationResult::solution

solution is an array of pointers to [OptimizationSolution](#) objects

Definition at line 2264 of file OSResult.h.

6.118.4.6 OtherSolverOutput* OptimizationResult::otherSolverOutput

otherSolverOutput is a pointer to an [OtherSolverOutput](#) object

Definition at line 2269 of file OSResult.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.119 OptimizationSolution Class Reference

The [OptimizationSolution](#) Class.

```
#include <OSResult.h>
```

[illegible]

- `OptimizationSolution ()`
Default constructor.
- `~OptimizationSolution ()`
Class destructor.
- `bool isEqual (OptimizationSolution *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

- int `targetObjectiveIdx`
the index of the objective function for which we are reporting solution information
- std::string `targetObjectiveName`
an optional name of the objective function for which we are reporting solution information
- bool `weightedObjectives`
a marker to track whether the objectives are weighted
- `OptimizationSolutionStatus * status`
status is a pointer to an `OptimizationSolutionStatus` object associated with this optimization solution
- std::string `message`

a message associated with this solution

- [VariableSolution](#) * [variables](#)

variables holds the solution information for the variables

- [ObjectiveSolution](#) * [objectives](#)

objectives holds the solution information for the objectives

- [ConstraintSolution](#) * [constraints](#)

constraints holds the solution information for the constraints

- [OtherSolutionResults](#) * [otherSolutionResults](#)

otherSolutionResults is a pointer to an [OtherSolutionResults](#) object that is associated with this optimization solution

6.119.1 Detailed Description

The [OptimizationSolution](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting the various components of an optimization solution.

Definition at line 2032 of file OSResult.h.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 OptimizationSolution::OptimizationSolution ()

Default constructor.

6.119.2.2 OptimizationSolution::~~OptimizationSolution ()

Class destructor.

6.119.3 Member Function Documentation

6.119.3.1 bool OptimizationSolution::isEqual (OptimizationSolution * *that*)

A function to check for the equality of two objects.

6.119.3.2 `bool OptimizationSolution::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.119.4 Member Data Documentation**6.119.4.1** `int OptimizationSolution::targetObjectiveIdx`

the index of the objective function for which we are reporting solution information

Definition at line 2039 of file OSResult.h.

6.119.4.2 `std::string OptimizationSolution::targetObjectiveName`

an optional name of the objective function for which we are reporting solution information

Definition at line 2044 of file OSResult.h.

6.119.4.3 `bool OptimizationSolution::weightedObjectives`

a marker to track whether the objectives are weighted

Definition at line 2047 of file OSResult.h.

6.119.4.4 `OptimizationSolutionStatus* OptimizationSolution::status`

status is a pointer to an [OptimizationSolutionStatus](#) object associated with this optimization solution

Definition at line 2052 of file OSResult.h.

6.119.4.5 `std::string OptimizationSolution::message`

a message associated with this solution

Definition at line 2055 of file OSResult.h.

6.119.4.6 `VariableSolution* OptimizationSolution::variables`

variables holds the solution information for the variables

Definition at line 2060 of file OSResult.h.

6.119.4.7 `ObjectiveSolution* OptimizationSolution::objectives`

objectives holds the solution information for the objectives

Definition at line 2065 of file OSResult.h.

6.119.4.8 `ConstraintSolution* OptimizationSolution::constraints`

constraints holds the solution information for the constraints

Definition at line 2070 of file OSResult.h.

6.119.4.9 OtherSolutionResults* OptimizationSolution::otherSolutionResults

otherSolutionResults is a pointer to an [OtherSolutionResults](#) object that is associated with this optimization solution
Definition at line 2075 of file OSResult.h.

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

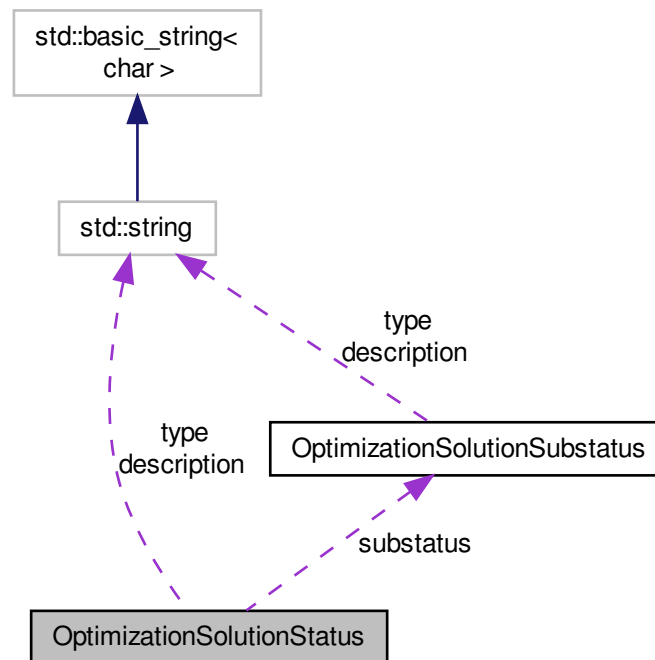
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.120 OptimizationSolutionStatus Class Reference

The [OptimizationSolutionStatus](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OptimizationSolutionStatus:



Public Member Functions

- [OptimizationSolutionStatus](#) ()
Default constructor.
- [~OptimizationSolutionStatus](#) ()
Class destructor.
- bool [IsEqual](#) ([OptimizationSolutionStatus](#) *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant)

A function to make a random instance of this class.

Public Attributes

- int [numberOfSubstatuses](#)
the number of substatus objects
- std::string [type](#)
the type of solution status
- std::string [description](#)
a description of the solution status type
- [OptimizationSolutionSubstatus](#) ** [substatus](#)
a pointer to an array of substatus objects

6.120.1 Detailed Description

The [OptimizationSolutionStatus](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for holding various attributes of a solution status

Definition at line 790 of file OSResult.h.

6.120.2 Constructor & Destructor Documentation

6.120.2.1 OptimizationSolutionStatus::OptimizationSolutionStatus ()

Default constructor.

6.120.2.2 OptimizationSolutionStatus::~~OptimizationSolutionStatus ()

Class destructor.

6.120.3 Member Function Documentation

6.120.3.1 bool OptimizationSolutionStatus::isEqual (OptimizationSolutionStatus * *that*)

A function to check for the equality of two objects.

6.120.3.2 bool OptimizationSolutionStatus::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.120.4 Member Data Documentation

6.120.4.1 int OptimizationSolutionStatus::numberOfSubstatuses

the number of substatus objects

Definition at line 795 of file OSResult.h.

6.120.4.2 std::string OptimizationSolutionStatus::type

the type of solution status

Definition at line 798 of file OSResult.h.

6.120.4.3 std::string OptimizationSolutionStatus::description

a description of the solution status type

Definition at line 801 of file OSResult.h.

6.120.4.4 OptimizationSolutionSubstatus** OptimizationSolutionStatus::substatus

a pointer to an array of substatus objects

Definition at line 804 of file OSResult.h.

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

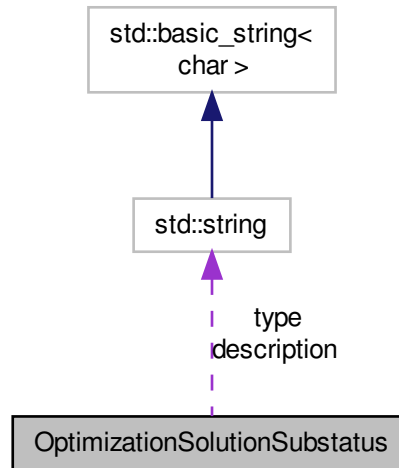
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.121 OptimizationSolutionSubstatus Class Reference

The [OptimizationSolutionSubstatus](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OptimizationSolutionSubstatus:



Public Member Functions

- [OptimizationSolutionSubstatus](#) ()
Default constructor.
- [~OptimizationSolutionSubstatus](#) ()
Class destructor.
- bool [IsEqual](#) ([OptimizationSolutionSubstatus](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string [type](#)
the type of the solution substatus
- std::string [description](#)
a description of the solution substatus

6.121.1 Detailed Description

The [OptimizationSolutionSubstatus](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for holding various attributes of a solution status

Definition at line 741 of file OSResult.h.

6.121.2 Constructor & Destructor Documentation**6.121.2.1 OptimizationSolutionSubstatus::OptimizationSolutionSubstatus ()**

Default constructor.

6.121.2.2 OptimizationSolutionSubstatus::~~OptimizationSolutionSubstatus ()

Class destructor.

6.121.3 Member Function Documentation**6.121.3.1 bool OptimizationSolutionSubstatus::isEqual (OptimizationSolutionSubstatus * *that*)**

A function to check for the equality of two objects.

6.121.3.2 bool OptimizationSolutionSubstatus::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.121.4 Member Data Documentation**6.121.4.1 std::string OptimizationSolutionSubstatus::type**

the type of the solution substatus

Definition at line 746 of file OSResult.h.

6.121.4.2 std::string OptimizationSolutionSubstatus::description

a description of the solution substatus

Definition at line 749 of file OSResult.h.

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

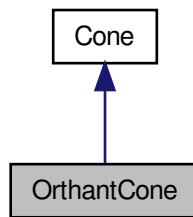
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.122 OrthantCone Class Reference

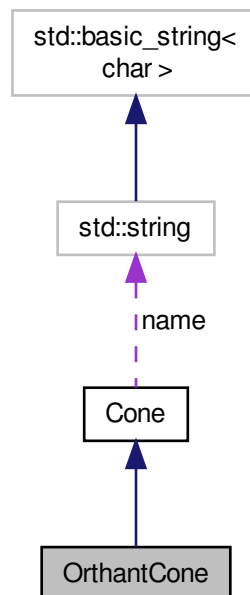
The [OrthantCone](#) Class.

```
#include <OSInstance.h>
```

Inheritance diagram for OrthantCone:



Collaboration diagram for OrthantCone:



Public Member Functions

- [OrthantCone](#) ()
default constructor.
- [~OrthantCone](#) ()
default destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write an [OrthantCone](#) object in XML format.
- bool [isEqual](#) ([OrthantCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([OrthantCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- double * [ub](#)
For each dimension of the cone, give the upper and lower bounds The upper bound can be only zero or +infy, the lower bound can be only zero or -infy.
- double * [lb](#)

6.122.1 Detailed Description

The [OrthantCone](#) Class.

Remarks

The in-memory representation of the OSiL element <orthantCone>

Definition at line 726 of file OSInstance.h.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 [OrthantCone::OrthantCone](#) ()

default constructor.

6.122.2.2 [OrthantCone::~~OrthantCone](#) ()

default destructor.

6.122.3 Member Function Documentation

6.122.3.1 virtual std::string [OrthantCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.122.3.2 virtual std::string OrthantCone::getConeInXML () [virtual]

Write an [OrthantCone](#) object in XML format.

This is used by [OSILWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.122.3.3 bool OrthantCone::isEqual (OrthantCone * that)

A function to check for the equality of two objects.

6.122.3.4 bool OrthantCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.122.3.5 bool OrthantCone::deepCopyFrom (OrthantCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.122.4 Member Data Documentation

6.122.4.1 double* OrthantCone::ub

For each dimension of the cone, give the upper and lower bounds The upper bound can be only zero or +infy, the lower bound can be only zero or -infy,.

Definition at line 733 of file OSInstance.h.

6.122.4.2 double* OrthantCone::lb

Definition at line 734 of file OSInstance.h.

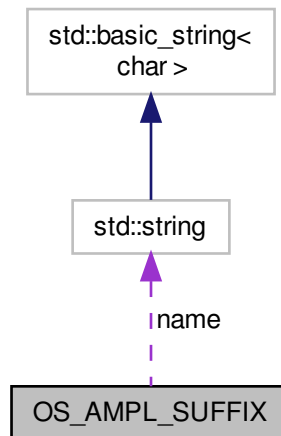
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.123 OS_AMPL_SUFFIX Struct Reference

```
#include <OSn12OS.h>
```

Collaboration diagram for OS_AMPL_SUFFIX:



Public Attributes

- `std::string` [name](#)
- `OS_AMPL_SUFFIX_TYPE` [type](#)
- `OS_AMPL_SUFFIX_SCOPE` [scope](#)
- `OS_AMPL_SUFFIX_DIRECTION` [direction](#)

6.123.1 Detailed Description

Definition at line 70 of file OSn12OS.h.

6.123.2 Member Data Documentation

6.123.2.1 `std::string` OS_AMPL_SUFFIX::name

Definition at line 72 of file OSn12OS.h.

6.123.2.2 `OS_AMPL_SUFFIX_TYPE` OS_AMPL_SUFFIX::type

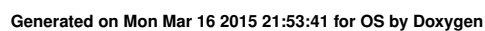
Definition at line 73 of file OSn12OS.h.

6.123.2.3 `OS_AMPL_SUFFIX_SCOPE` OS_AMPL_SUFFIX::scope

Definition at line 74 of file OSn12OS.h.

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

- Collaboration diagram for OSCommandLine:



Public Member Functions

- [OSCommandLine](#) ()
constructor method
- [~OSCommandLine](#) ()
destructor method
- void [reset_options](#) ()
a function to reset the command line to default values useful especially in the interactive shell
- std::string [list_options](#) ()
a function to print the current command line option values
- void [convertSolverNameToLowerCase](#) ()
to avoid ambiguity it might be necessary to convert the solver name to lower case ...
- void [convertSolverNameToUpperCase](#) ()
...

Public Attributes

- [OSInstance](#) * [osinstance](#)
osinstance is a representation of the instance in [OSInstance](#) format
- [OSOption](#) * [osoption](#)
osoption is a representation of the solver options in [OSOption](#) format
- std::string [serviceLocation](#)
serviceLocation is the URL of the remote solver when a local solver is not used
- std::string [serviceMethod](#)
the service method the OSSolverService should execute, i.e.
- std::string [solverName](#)
the name of the solver to be invoked locally, e.g -solver lpopt
- std::string [configFile](#)
configFile is the name of the file that holds the configuration options if the OSSolverService reads its options from a file rather than command line inputs
- std::string [osilFile](#)
osilFile is the name of the file that holds the model instance in OSiL format
- std::string [osil](#)
osil is the content of the osilFile
- std::string [osilOutputFile](#)
osilOutputFile is the name of the file to which the instance can be written in OSiL format.
- std::string [osolFile](#)
osolFile is the name of the file that holds the solver options in OSoL format
- std::string [osol](#)
osol is the content of the osolFile
- std::string [osolOutputFile](#)
osolOutputFile is the name of the file to which the solver options can be written in OSoL format.
- std::string [osrlFile](#)
osrlFile is the name of the file where the solver should write the result (in OSrL format)
- std::string [insListFile](#)
name of the file containing the instance in LINDO instruction list format
- std::string [insList](#)

- insList is the content of the insListFile – this is not implemented*
- `std::string osplInputFile`
*name of an input file with xml in OS process language format, used for example to knock on a server, for example -osplInput
../data/osplFiles/demo.ospl*
- `std::string osplInput`
osplInput is the content of the osplInputFile
- `std::string osplOutputFile`
name of an output file where the solver should write the result of a knock or kill service request
- `std::string mpsFile`
the name of the file that holds an instance in MPS format
- `std::string mps`
the string that holds an instance in MPS format
- `std::string nlFile`
the name of the file that holds an instance in AMPL nl format
- `std::string nl`
the string that holds an instance in AMPL nl format
- `std::string datFile`
the name of the file that holds an instance in GAMS dat format
- `std::string dat`
the string that holds an instance in GAMS dat format
- `std::string gamsControlFile`
the name of the file that holds the GAMS control parameters
- `std::string browser`
this parameter is a path to the browser on the local machine.
- `int printLevel`
*this parameter controls the amount of output to print the higher the number, the more output is generated details about
print levels can be found in [OSOutput.h](#)*
- `std::string logFile`
*this optional parameter contains the path to a logfile that can be used as an alternate output stream in addition to the
normal output to stdout*
- `int filePrintLevel`
*this parameter controls the amount of output to send to the log file (if used) the higher the number, the more output is
generated details about print levels can be found in [OSOutput.h](#)*
- `std::string jobID`
the JobID
- `bool invokeHelp`
if this parameter is true we print the contents of the file help.txt and return
- `bool listOptions`
if this parameter is true we echo the values of the options found on the command line
- `bool writeVersion`
if this parameter is true we print the current version of the OS project
- `bool printModel`
if this parameter is true we print the current instance as read from an osil, nl or mps file
- `std::string printRowNumberAsString`
*this parameter contains a string representation (!) of the row number if only a single row (constraint or objective) of the
current instance is to be printed String representations are easier to parse in [OSParseosss.l](#) and are easier to recognize
as being present or absent*
- `bool quit`
if this parameter is true we quit/exit

6.124.1 Detailed Description

This class is used to store command line options for the OSSolverService executable and to provide methods to manipulate them.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

the OSSolverService requires numerous options and these options are stored in the [OSCommandLine](#) class

Definition at line 36 of file OSCommandLine.h.

6.124.2 Constructor & Destructor Documentation

6.124.2.1 OSCommandLine::OSCommandLine ()

constructor method

6.124.2.2 OSCommandLine::~~OSCommandLine ()

destructor method

6.124.3 Member Function Documentation

6.124.3.1 void OSCommandLine::reset_options ()

a function to reset the command line to default values useful especially in the interactive shell

6.124.3.2 std::string OSCommandLine::list_options ()

a function to print the current command line option values

6.124.3.3 void OSCommandLine::convertSolverNameToLowerCase ()

to avoid ambiguity it might be necessary to convert the solver name to lower case ...

6.124.3.4 void OSCommandLine::convertSolverNameToUpperCase ()

...

or to upper case

6.124.4 Member Data Documentation

6.124.4.1 OSInstance* OSCommandLine::osinstance

osinstance is a representation of the instance in [OSInstance](#) format

Definition at line 42 of file OSCommandLine.h.

6.124.4.2 `OSOption*` `OSCommandLine::osoption`

`osoption` is a representation of the solver options in [OSOption](#) format

Definition at line 47 of file `OSCommandLine.h`.

6.124.4.3 `std::string` `OSCommandLine::serviceLocation`

`serviceLocation` is the URL of the remote solver when a local solver is not used

Definition at line 52 of file `OSCommandLine.h`.

6.124.4.4 `std::string` `OSCommandLine::serviceMethod`

the service method the `OSSolverService` should execute, i.e.

solve, send, getJobID, kill, knock, or retrieve

Definition at line 57 of file `OSCommandLine.h`.

6.124.4.5 `std::string` `OSCommandLine::solverName`

the name of the solver to be invoked locally, e.g -solver lpopt

Definition at line 62 of file `OSCommandLine.h`.

6.124.4.6 `std::string` `OSCommandLine::configFile`

`configFile` is the name of the file that holds the configuration options if the `OSSolverService` reads its options from a file rather than command line inputs

Definition at line 68 of file `OSCommandLine.h`.

6.124.4.7 `std::string` `OSCommandLine::osilFile`

`osilFile` is the name of the file that holds the model instance in OSiL format

Definition at line 73 of file `OSCommandLine.h`.

6.124.4.8 `std::string` `OSCommandLine::osil`

`osil` is the content of the `osilFile`

Definition at line 77 of file `OSCommandLine.h`.

6.124.4.9 `std::string` `OSCommandLine::osilOutputFile`

`osilOutputFile` is the name of the file to which the instance can be written in OSiL format.

This is especially useful for converting the instance from other representation formats such as AMPL nl format or MPS format. If this parameter is empty, the instance will not be saved.

Definition at line 84 of file `OSCommandLine.h`.

6.124.4.10 `std::string` `OSCommandLine::osolFile`

`osolFile` is the name of the file that holds the solver options in OSoL format

Definition at line 89 of file `OSCommandLine.h`.

6.124.4.11 std::string OSCommandLine::osol

osol is the content of the osolFile

Definition at line 93 of file OSCommandLine.h.

6.124.4.12 std::string OSCommandLine::osolOutputFile

osolOutputFile is the name of the file to which the solver options can be written in OSoL format.

This is especially useful when an instance represented in another representation format such as AMPL nl format or MPS format contains array-valued options such as initial values or basis information. If this parameter is empty, the solver options will not be saved.

Definition at line 101 of file OSCommandLine.h.

6.124.4.13 std::string OSCommandLine::osrFile

osrFile is the name of the file where the solver should write the result (in OSrL format)

Definition at line 106 of file OSCommandLine.h.

6.124.4.14 std::string OSCommandLine::insListFile

name of the file containing the instance in LINDO instruction list format

Definition at line 111 of file OSCommandLine.h.

6.124.4.15 std::string OSCommandLine::insList

insList is the content of the insListFile – this is not implemented

Definition at line 115 of file OSCommandLine.h.

6.124.4.16 std::string OSCommandLine::osplInputFile

name of an input file with xml in OS process language format, used for example to knock on a server, for example
-osplInput ../data/osplFiles/demo.ospl

Definition at line 121 of file OSCommandLine.h.

6.124.4.17 std::string OSCommandLine::osplInput

osplInput is the content of the osplInputFile

Definition at line 125 of file OSCommandLine.h.

6.124.4.18 std::string OSCommandLine::osplOutputFile

name of an output file where the solver should write the result of a knock or kill service request

Definition at line 131 of file OSCommandLine.h.

6.124.4.19 std::string OSCommandLine::mpsFile

the name of the file that holds an instance in MPS format

Definition at line 134 of file OSCommandLine.h.

6.124.4.20 std::string OSCommandLine::mps

the string that holds an instance in MPS format

Definition at line 137 of file OSCommandLine.h.

6.124.4.21 `std::string OSCommandLine::nlFile`

the name of the file that holds an instance in AMPL nl format

Definition at line 140 of file OSCommandLine.h.

6.124.4.22 `std::string OSCommandLine::nl`

the string that holds an instance in AMPL nl format

Definition at line 143 of file OSCommandLine.h.

6.124.4.23 `std::string OSCommandLine::datFile`

the name of the file that holds an instance in GAMS dat format

Definition at line 146 of file OSCommandLine.h.

6.124.4.24 `std::string OSCommandLine::dat`

the string that holds an instance in GAMS dat format

Definition at line 149 of file OSCommandLine.h.

6.124.4.25 `std::string OSCommandLine::gamsControlFile`

the name of the file that holds the GAMS control parameters

Definition at line 152 of file OSCommandLine.h.

6.124.4.26 `std::string OSCommandLine::browser`

this parameter is a path to the browser on the local machine.

If this optional parameter is specified then the solver result in OSrL format is transformed using XSLT into HTML and displayed in the browser, e.g. `-browser /Applications/Firefox.app/Contents/MacOS/firefox`

Definition at line 160 of file OSCommandLine.h.

6.124.4.27 `int OSCommandLine::printLevel`

this parameter controls the amount of output to print the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

Definition at line 166 of file OSCommandLine.h.

6.124.4.28 `std::string OSCommandLine::logFile`

this optional parameter contains the path to a logfile that can be used as an alternate output stream in addition to the normal output to stdout

Definition at line 172 of file OSCommandLine.h.

6.124.4.29 `int OSCommandLine::filePrintLevel`

this parameter controls the amount of output to send to the log file (if used) the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

Definition at line 179 of file OSCommandLine.h.

6.124.4.30 `std::string OSCommandLine::jobID`

the JobID

Definition at line 182 of file OSCommandLine.h.

6.124.4.31 `bool OSCommandLine::invokeHelp`

if this parameter is true we print the contents of the file help.txt and return

Definition at line 187 of file OSCommandLine.h.

6.124.4.32 `bool OSCommandLine::listOptions`

if this parameter is true we echo the values of the options found on the command line

Definition at line 192 of file OSCommandLine.h.

6.124.4.33 `bool OSCommandLine::writeVersion`

if this parameter is true we print the current version of the OS project

Definition at line 197 of file OSCommandLine.h.

6.124.4.34 `bool OSCommandLine::printModel`

if this parameter is true we print the current instance as read from an osil, nl or mps file

Definition at line 202 of file OSCommandLine.h.

6.124.4.35 `std::string OSCommandLine::printRowNumberAsString`

this parameter contains a string representation (!) of the row number if only a single row (constraint or objective) of the current instance is to be printed String representations are easier to parse in OSParseosss.l and are easier to recognize as being present or absent

Definition at line 210 of file OSCommandLine.h.

6.124.4.36 `bool OSCommandLine::quit`

if this parameter is true we quit/exit

- only used in the interactive shell

Definition at line 215 of file OSCommandLine.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLine.h](#)

6.125 OSCommandLineReader Class Reference

The [OSCommandLineReader](#) Class.

```
#include <OSCommandLineReader.h>
```

Public Member Functions

- [OSCommandLineReader](#) ()

- OSCommandLineReader* class constructor.
- [~OSCommandLineReader](#) ()
OSCommandLineReader class destructor.
- [OSCommandLine](#) * [readCommandLine](#) (const std::string &osss) throw (ErrorClass)
Get an [OSCommandLine](#) object from a command line string.
- [OSCommandLine](#) * [parseString](#) (const std::string &osss) throw (ErrorClass)
Parse a string and store it into an [OSCommandLine](#) object.

6.125.1 Detailed Description

The [OSCommandLineReader](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

A class for parsing a command line string and creating an [OSCommandLine](#) object from the string. This method can be used in OSSolverService, OSAmplClient, as well as the interactive shell

Definition at line 39 of file OSCommandLineReader.h.

6.125.2 Constructor & Destructor Documentation

6.125.2.1 OSCommandLineReader::OSCommandLineReader ()

[OSCommandLineReader](#) class constructor.

Parameters

osss	is the command line to be parsed
----------------------	----------------------------------

6.125.2.2 OSCommandLineReader::~~OSCommandLineReader ()

[OSCommandLineReader](#) class destructor.

6.125.3 Member Function Documentation

6.125.3.1 OSCommandLine* OSCommandLineReader::readCommandLine (const std::string & osss) throw (ErrorClass)

Get an [OSCommandLine](#) object from a command line string.

Parameters

osss	a command line string.
----------------------	------------------------

Returns

the [OSCommandLine](#) object corresponding to the command line string.

Remarks

Calls method `parseString` once and if a `configFile` item is found calls method `parseString` two more times (with the config file contents and again with the original command line)

6.125.3.2 OSExpressionTree* OSExpressionTreeReader::parseString (const std::string & osss) throw (ErrorClass)

Parse a string and store it into an [OSExpressionTree](#) object.

Parameters

<code>osss</code>	a command line string.
-------------------	------------------------

Returns

the [OSExpressionTree](#) object corresponding to the command line string.

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

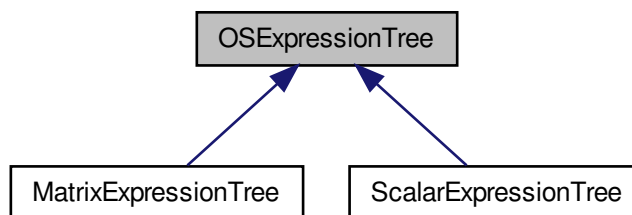
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTreeReader.h](#)

6.126 OSExpressionTree Class Reference

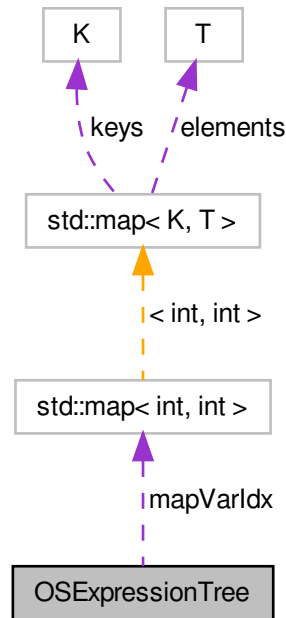
Used to hold the instance in memory.

```
#include <OSExpressionTree.h>
```

Inheritance diagram for `OSExpressionTree`:



Collaboration diagram for OSExpressionTree:



Public Member Functions

- [OSExpressionTree \(\)](#)
default constructor.
- [~OSExpressionTree \(\)](#)
default destructor.
- `bool` [isEqual](#) ([OSExpressionTree](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `std::map< int, int > *` [mapVarIdx](#)
`m_mapVarIdx` is a map used to generate the infix expression for AD the key is `idx`, a variable number; the value of the map is the location of the corresponding entry in the sparse Jacobian
- `bool` [m_bIndexMapGenerated](#)
Retrieve a map of the indices of the variables that are in the expression tree.
- `bool` [bADMustReTape](#)
is true if an AD Expression Tree has an expression that can change depending on the value of the input, e.g.
- `bool` [bDestroyNINodes](#)
`m_bDestroyNINodes` is true if the destructor deletes the nodes in the Expression tree

6.126.1 Detailed Description

Used to hold the instance in memory.

Remarks

This is a generic class. Specific classes [ScalarExpressionTree](#) (for expressions that evaluate to scalar values) and [MatrixExpressionTrees](#) (for expressions that evaluate to matrices) are derived from this class.

Definition at line 37 of file OSExpressionTree.h.

6.126.2 Constructor & Destructor Documentation

6.126.2.1 OSExpressionTree::OSExpressionTree ()

default constructor.

6.126.2.2 OSExpressionTree::~~OSExpressionTree ()

default destructor.

6.126.3 Member Function Documentation

6.126.3.1 bool OSExpressionTree::isEqual (OSExpressionTree * *that*)

A function to check for the equality of two objects.

6.126.4 Member Data Documentation

6.126.4.1 std::map<int, int>* OSExpressionTree::mapVarIdx

m_mapVarIdx is a map used to generate the infix expression for AD the key is idx, a variable number; the value of the map is the location of the corresponding entry in the sparse Jacobian

Definition at line 55 of file OSExpressionTree.h.

6.126.4.2 bool OSExpressionTree::m_bIndexMapGenerated

Retrieve a map of the indices of the variables that are in the expression tree.

Returns

a map of the variables in the current expression tree. m_bIndexMapGenerated is set to true if getVariableIndices-Map() has been called

Definition at line 70 of file OSExpressionTree.h.

6.126.4.3 bool OSExpressionTree::bADMustReTape

is true if an AD Expression Tree has an expression that can change depending on the value of the input, e.g.

an if statement – false by default

Definition at line 76 of file OSExpressionTree.h.

6.126.4.4 bool OSExpressionTree::bDestroyNINodes

m_bDestroyNINodes is true if the destructor deletes the nodes in the Expression tree

Definition at line 81 of file OSExpressionTree.h.

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

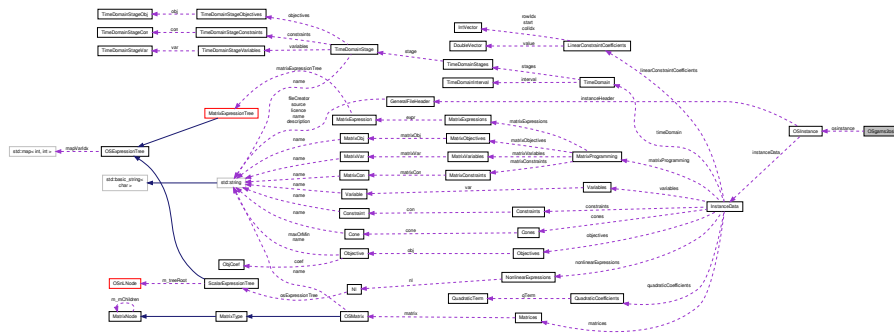
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTree.h

6.127 OSgams2osil Class Reference

Creating a [OSInstance](#) from a GAMS model given as GAMS Modeling Object (GMO).

```
#include <OSgams2osil.hpp>
```

Collaboration diagram for OSgams2osil:



Public Member Functions

- [OSgams2osil](#) (struct gmoRec *gmo_=NULL)
- [OSgams2osil](#) (std::string gamsControlFile)
- [~OSgams2osil](#) ()
- bool [initGMO](#) (const char *datfile)
- bool [createOSInstance](#) ()
Creates an [OSInstance](#) from the GAMS smag instance representation.
- [OSInstance *](#) [takeOverOSInstance](#) ()
Gives [OSInstance](#) and ownership to calling function.
- [OSInstance *](#) [getOSInstance](#) ()
Gives [OSInstances](#) but keeps ownership.

Public Attributes

- [OSInstance *](#) [osinstance](#)

6.127.1 Detailed Description

Creating a [OSInstance](#) from a GAMS model given as GAMS Modeling Object (GMO).

Definition at line 22 of file OSgams2osil.hpp.

6.127.2 Constructor & Destructor Documentation

6.127.2.1 OSgams2osil::OSgams2osil (struct gmoRec * *gmo_* = NULL)6.127.2.2 OSgams2osil::OSgams2osil (std::string *gamsControlFile*)

6.127.2.3 OSgams2osil::~OSgams2osil ()

6.127.3 Member Function Documentation

6.127.3.1 bool OSgams2osil::initGMO (const char * *datfile*)

6.127.3.2 bool OSgams2osil::createOSInstance ()

Creates an [OSInstance](#) from the GAMS smag instance representation.

Returns

whether the instance is created successfully.

6.127.3.3 OSInstance* OSgams2osil::takeOverOSInstance ()

Gives [OSInstance](#) and ownership to calling function.

This object forgets about the created instance.

6.127.3.4 OSInstance* OSgams2osil::getOSInstance () [inline]

Gives OSInstances but keeps ownership.

Destruction will destruct [OSInstance](#).

Definition at line 55 of file OSgams2osil.hpp.

6.127.4 Member Data Documentation

6.127.4.1 OSInstance* OSgams2osil::osinstance

Definition at line 32 of file OSgams2osil.hpp.

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

- [/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSgams2osil.hpp](#)

6.128 OSGeneral Class Reference

```
#include <OSGeneral.h>
```

6.128.1 Detailed Description

Definition at line 969 of file OSGeneral.h.

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

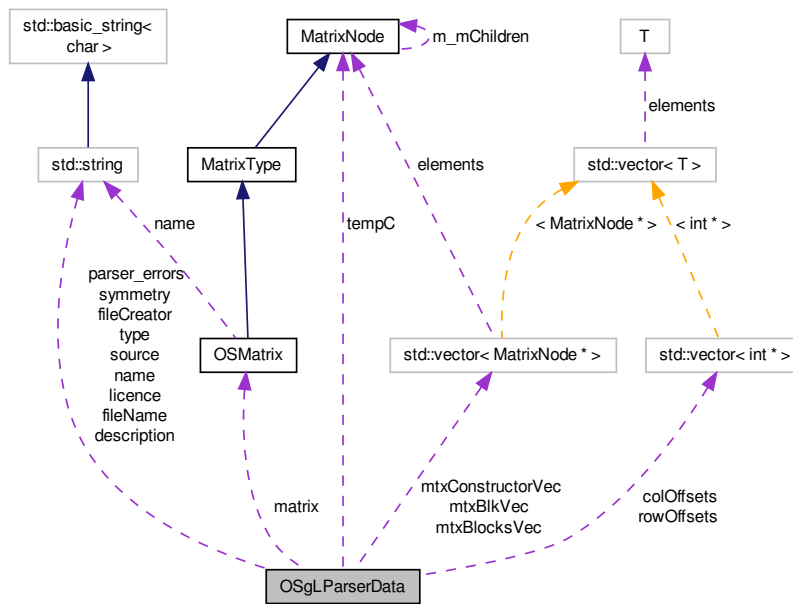
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.129 OSgLParseData Class Reference

The [OSgLParseData](#) Class.

```
#include <OSgLParseData.h>
```

Collaboration diagram for OSgLParseData:



Public Member Functions

- [OSgLParseData](#) ()
the *OSgLParseData* class constructor
- [~OSgLParseData](#) ()

Public Attributes

- bool [osglMultPresent](#)
data structure to process an *IntVector* and hold the data temporarily
- bool [osglIncrPresent](#)
- bool [osglNumberOfEIIPresent](#)
- int [osglNumberOfEI](#)
- int * [osglIntArray](#)
- int [osglMult](#)
- int [osglIncr](#)
- int [osglSize](#)
- int [osglCounter](#)
- int [osglTempint](#)
- double * [osglDblArray](#)

- int * [osglValArray](#)
- std::string [fileName](#)
 - data structure to process a [GeneralFileHeader](#) and hold the data temporarily*
- std::string [source](#)
- std::string [description](#)
- std::string [fileCreator](#)
- std::string [licence](#)
- bool [fileNamePresent](#)
- bool [sourcePresent](#)
- bool [descriptionPresent](#)
- bool [fileCreatorPresent](#)
- bool [licencePresent](#)
- void * [scanner](#)
 - scanner is used to store data in a reentrant lexer we use this to pass an [OSoLParseData](#) object to the parser*
- char * [errorText](#)
 - if the parser finds invalid text it is held here and we delete if the file was not valid*
- std::string [parser_errors](#)
 - used to accumulate error message so the parser does not die on the first error encountered*
- bool [ignoreDataAfterErrors](#)
 - two booleans to govern the behavior after an error has been encountered*
- bool [suppressFurtherErrorMessage](#)
- [OSMatrix](#) ** [matrix](#)
 - We need to hold an array of <matrix> elements temporarily.*
- int [numberOfMatrices](#)
 - We also need to keep track locally of the number of matrices.*
- int [matrixCounter](#)
- [MatrixNode](#) * [tempC](#)
 - This matrix constructor is needed in order to properly push the constructor vector.*
- std::vector< [MatrixNode](#) * > [mtxConstructorVec](#)
 - Several vectors to process the matrix nodes into the right order.*
- std::vector< [MatrixNode](#) * > [mtxBlocksVec](#)
- std::vector< [MatrixNode](#) * > [mtxBlkVec](#)
- std::vector< int * > [rowOffsets](#)
 - Vectors to hold rowOffset and colOffset arrays in a place where they are easily accessible while the <block> children are processed.*
- std::vector< int * > [colOffsets](#)
- std::string [symmetry](#)
 - other data structures to temporarily hold a matrix and its subordinate elements*
- bool [symmetryPresent](#)
- std::string [name](#)
- std::string [type](#)
- int [idx](#)
- bool [namePresent](#)
- bool [typePresent](#)
- bool [idxPresent](#)
- int [numberOfBlocks](#)
- int [numberOfColumns](#)
- int [numberOfRows](#)
- int [baseMatrixIdx](#)

- int [targetMatrixFirstRow](#)
- int [targetMatrixFirstCol](#)
- int [baseMatrixStartRow](#)
- int [baseMatrixStartCol](#)
- int [baseMatrixEndRow](#)
- int [baseMatrixEndCol](#)
- bool [baseTranspose](#)
- double [scalarMultiplier](#)
- bool [baseMatrixIdxPresent](#)
- bool [targetMatrixFirstRowPresent](#)
- bool [targetMatrixFirstColPresent](#)
- bool [baseMatrixStartRowPresent](#)
- bool [baseMatrixStartColPresent](#)
- bool [baseMatrixEndRowPresent](#)
- bool [baseMatrixEndColPresent](#)
- bool [baseTransposePresent](#)
- bool [scalarMultiplierPresent](#)
- bool [rowMajorPresent](#)
- bool [rowMajor](#)
- int [blockRowIdx](#)
- bool [blockRowIdxPresent](#)
- int [blockColIdx](#)
- bool [blockColIdxPresent](#)
- bool [osglConstantPresent](#)
- bool [osglCoefPresent](#)
- double [osglCoef](#)
- bool [numberOfBlocksPresent](#)
- bool [numberOfColumnsPresent](#)
- bool [numberOfRowsPresent](#)
- bool [numberOfValuesPresent](#)
- int [numberOfValues](#)
- bool [numberOfVarIdxPresent](#)
- int [numberOfVarIdx](#)
- bool [numberOfEIIPresent](#)
- int [numberOfEI](#)
- int [osglNumberOfNonzeros](#)
- int [osglNonzeroCounter](#)
- int * [matrixBlockNumberOfRows](#)
- int * [matrixBlockNumberOfCols](#)
- [ENUM_NL_EXPR_SHAPE](#) shape
- bool [shapePresent](#)
- [ENUM_CONREFERENCE_VALUETYPE](#) valueType
- bool [valueTypePresent](#)

6.129.1 Detailed Description

The [OSgLParseData](#) Class.

Remarks

the [OSgLParseData](#) class is used to temporarily hold data found in parsing the OSgL data structures. we do this so we can write reusable code.

Definition at line 33 of file [OSgLParseData.h](#).

6.129.2 Constructor & Destructor Documentation

6.129.2.1 OSGLParserData::OSGLParserData ()

the [OSGLParserData](#) class constructor

6.129.2.2 OSGLParserData::~OSGLParserData ()

6.129.3 Member Data Documentation

6.129.3.1 bool OSGLParserData::osglMultPresent

data structure to process an [IntVector](#) and hold the data temporarily

Definition at line 38 of file OSGLParserData.h.

6.129.3.2 bool OSGLParserData::osglIncrPresent

Definition at line 39 of file OSGLParserData.h.

6.129.3.3 bool OSGLParserData::osglNumberOfEIPresent

Definition at line 40 of file OSGLParserData.h.

6.129.3.4 int OSGLParserData::osglNumberOfEI

Definition at line 41 of file OSGLParserData.h.

6.129.3.5 int* OSGLParserData::osglIntArray

Definition at line 42 of file OSGLParserData.h.

6.129.3.6 int OSGLParserData::osglMult

Definition at line 43 of file OSGLParserData.h.

6.129.3.7 int OSGLParserData::osglIncr

Definition at line 44 of file OSGLParserData.h.

6.129.3.8 int OSGLParserData::osglSize

Definition at line 45 of file OSGLParserData.h.

6.129.3.9 int OSGLParserData::osglCounter

Definition at line 46 of file OSGLParserData.h.

6.129.3.10 int OSGLParserData::osglTempint

Definition at line 47 of file OSGLParserData.h.

6.129.3.11 double* OSGLParserData::osglDbIArray

Definition at line 49 of file OSGLParserData.h.

6.129.3.12 int* OSGLParserData::osglValArray

Definition at line 50 of file OSGLParserData.h.

6.129.3.13 std::string OSGLParserData::fileName

data structure to process a [GeneralFileHeader](#) and hold the data temporarily

Definition at line 53 of file OSGLParserData.h.

6.129.3.14 std::string OSGLParserData::source

Definition at line 54 of file OSGLParserData.h.

6.129.3.15 std::string OSGLParserData::description

Definition at line 55 of file OSGLParserData.h.

6.129.3.16 std::string OSGLParserData::fileCreator

Definition at line 56 of file OSGLParserData.h.

6.129.3.17 std::string OSGLParserData::licence

Definition at line 57 of file OSGLParserData.h.

6.129.3.18 bool OSGLParserData::fileNamePresent

Definition at line 58 of file OSGLParserData.h.

6.129.3.19 bool OSGLParserData::sourcePresent

Definition at line 59 of file OSGLParserData.h.

6.129.3.20 bool OSGLParserData::descriptionPresent

Definition at line 60 of file OSGLParserData.h.

6.129.3.21 bool OSGLParserData::fileCreatorPresent

Definition at line 61 of file OSGLParserData.h.

6.129.3.22 bool OSGLParserData::licencePresent

Definition at line 62 of file OSGLParserData.h.

6.129.3.23 void* OSGLParserData::scanner

scanner is used to store data in a reentrant lexer we use this to pass an [OSoLParserData](#) object to the parser

Definition at line 74 of file OSGLParserData.h.

6.129.3.24 char* OSGLParserData::errorText

if the parser finds invalid text it is held here and we delete if the file was not valid

Definition at line 79 of file OSGLParserData.h.

6.129.3.25 `std::string OSGLParserData::parser_errors`

used to accumulate error message so the parser does not die on the first error encountered

Definition at line 84 of file OSGLParserData.h.

6.129.3.26 `bool OSGLParserData::ignoreDataAfterErrors`

two booleans to govern the behavior after an error has been encountered

Definition at line 87 of file OSGLParserData.h.

6.129.3.27 `bool OSGLParserData::suppressFurtherErrorMessages`

Definition at line 88 of file OSGLParserData.h.

6.129.3.28 `OSMatrix** OSGLParserData::matrix`

We need to hold an array of <matrix> elements temporarily.

Definition at line 91 of file OSGLParserData.h.

6.129.3.29 `int OSGLParserData::numberOfMatrices`

We also need to keep track locally of the number of matrices.

Definition at line 94 of file OSGLParserData.h.

6.129.3.30 `int OSGLParserData::matrixCounter`

Definition at line 95 of file OSGLParserData.h.

6.129.3.31 `MatrixNode* OSGLParserData::tempC`

This matrix constructor is needed in order to properly push the constructor vector.

Definition at line 98 of file OSGLParserData.h.

6.129.3.32 `std::vector<MatrixNode*> OSGLParserData::mtxConstructorVec`

Several vectors to process the matrix nodes into the right order.

Definition at line 101 of file OSGLParserData.h.

6.129.3.33 `std::vector<MatrixNode*> OSGLParserData::mtxBlocksVec`

Definition at line 102 of file OSGLParserData.h.

6.129.3.34 `std::vector<MatrixNode*> OSGLParserData::mtxBlkVec`

Definition at line 103 of file OSGLParserData.h.

6.129.3.35 `std::vector<int*> OSGLParserData::rowOffsets`

Vectors to hold rowOffset and colOffset arrays in a place where they are easily accessible while the <block> children are processed.

Definition at line 109 of file OSGLParserData.h.

6.129.3.36 `std::vector<int*> OSGLParserData::colOffsets`

Definition at line 110 of file OSGLParserData.h.

6.129.3.37 `std::string OSGLParserData::symmetry`

other data structures to temporarily hold a matrix and its subordinate elements

Definition at line 113 of file OSGLParserData.h.

6.129.3.38 `bool OSGLParserData::symmetryPresent`

Definition at line 114 of file OSGLParserData.h.

6.129.3.39 `std::string OSGLParserData::name`

Definition at line 115 of file OSGLParserData.h.

6.129.3.40 `std::string OSGLParserData::type`

Definition at line 116 of file OSGLParserData.h.

6.129.3.41 `int OSGLParserData::idx`

Definition at line 117 of file OSGLParserData.h.

6.129.3.42 `bool OSGLParserData::namePresent`

Definition at line 118 of file OSGLParserData.h.

6.129.3.43 `bool OSGLParserData::typePresent`

Definition at line 119 of file OSGLParserData.h.

6.129.3.44 `bool OSGLParserData::idxPresent`

Definition at line 120 of file OSGLParserData.h.

6.129.3.45 `int OSGLParserData::numberOfBlocks`

Definition at line 121 of file OSGLParserData.h.

6.129.3.46 `int OSGLParserData::numberOfColumns`

Definition at line 122 of file OSGLParserData.h.

6.129.3.47 `int OSGLParserData::numberOfRows`

Definition at line 123 of file OSGLParserData.h.

6.129.3.48 `int OSGLParserData::baseMatrixIdx`

Definition at line 124 of file OSGLParserData.h.

6.129.3.49 `int OSGLParserData::targetMatrixFirstRow`

Definition at line 125 of file OSGLParserData.h.

6.129.3.50 `int OSGLParserData::targetMatrixFirstCol`

Definition at line 126 of file OSGLParserData.h.

6.129.3.51 `int OSGLParserData::baseMatrixStartRow`

Definition at line 127 of file OSGLParserData.h.

6.129.3.52 `int OSGLParserData::baseMatrixStartCol`

Definition at line 128 of file OSGLParserData.h.

6.129.3.53 `int OSGLParserData::baseMatrixEndRow`

Definition at line 129 of file OSGLParserData.h.

6.129.3.54 `int OSGLParserData::baseMatrixEndCol`

Definition at line 130 of file OSGLParserData.h.

6.129.3.55 `bool OSGLParserData::baseTranspose`

Definition at line 131 of file OSGLParserData.h.

6.129.3.56 `double OSGLParserData::scalarMultiplier`

Definition at line 132 of file OSGLParserData.h.

6.129.3.57 `bool OSGLParserData::baseMatrixIdxPresent`

Definition at line 133 of file OSGLParserData.h.

6.129.3.58 `bool OSGLParserData::targetMatrixFirstRowPresent`

Definition at line 134 of file OSGLParserData.h.

6.129.3.59 `bool OSGLParserData::targetMatrixFirstColPresent`

Definition at line 135 of file OSGLParserData.h.

6.129.3.60 `bool OSGLParserData::baseMatrixStartRowPresent`

Definition at line 136 of file OSGLParserData.h.

6.129.3.61 `bool OSGLParserData::baseMatrixStartColPresent`

Definition at line 137 of file OSGLParserData.h.

6.129.3.62 `bool OSGLParserData::baseMatrixEndRowPresent`

Definition at line 138 of file OSGLParserData.h.

6.129.3.63 `bool OSGLParserData::baseMatrixEndColPresent`

Definition at line 139 of file OSGLParserData.h.

6.129.3.64 `bool OSGLParserData::baseTransposePresent`

Definition at line 140 of file OSGLParserData.h.

6.129.3.65 `bool OSGLParserData::scalarMultiplierPresent`

Definition at line 141 of file OSGLParserData.h.

6.129.3.66 `bool OSGLParserData::rowMajorPresent`

Definition at line 142 of file OSGLParserData.h.

6.129.3.67 `bool OSGLParserData::rowMajor`

Definition at line 143 of file OSGLParserData.h.

6.129.3.68 `int OSGLParserData::blockRowIdx`

Definition at line 144 of file OSGLParserData.h.

6.129.3.69 `bool OSGLParserData::blockRowIdxPresent`

Definition at line 145 of file OSGLParserData.h.

6.129.3.70 `int OSGLParserData::blockColIdx`

Definition at line 146 of file OSGLParserData.h.

6.129.3.71 `bool OSGLParserData::blockColIdxPresent`

Definition at line 147 of file OSGLParserData.h.

6.129.3.72 `bool OSGLParserData::osglConstantPresent`

Definition at line 148 of file OSGLParserData.h.

6.129.3.73 `bool OSGLParserData::osglCoefPresent`

Definition at line 149 of file OSGLParserData.h.

6.129.3.74 `double OSGLParserData::osglCoef`

Definition at line 150 of file OSGLParserData.h.

6.129.3.75 `bool OSGLParserData::numberOfBlocksPresent`

Definition at line 151 of file OSGLParserData.h.

6.129.3.76 `bool OSGLParserData::numberOfColumnsPresent`

Definition at line 152 of file OSGLParserData.h.

6.129.3.77 `bool OSGLParserData::numberOfRowsPresent`

Definition at line 153 of file OSGLParserData.h.

6.129.3.78 `bool OSGLParserData::numberOfValuesPresent`

Definition at line 154 of file OSGLParserData.h.

6.129.3.79 `int OSGLParserData::numberOfValues`

Definition at line 155 of file OSGLParserData.h.

6.129.3.80 `bool OSGLParserData::numberOfVarIdxPresent`

Definition at line 156 of file OSGLParserData.h.

6.129.3.81 `int OSGLParserData::numberOfVarIdx`

Definition at line 157 of file OSGLParserData.h.

6.129.3.82 `bool OSGLParserData::numberOfEIIPresent`

Definition at line 158 of file OSGLParserData.h.

6.129.3.83 `int OSGLParserData::numberOfEI`

Definition at line 159 of file OSGLParserData.h.

6.129.3.84 `int OSGLParserData::osglNumberOfNonzeros`

Definition at line 160 of file OSGLParserData.h.

6.129.3.85 `int OSGLParserData::osglNonzeroCounter`

Definition at line 161 of file OSGLParserData.h.

6.129.3.86 `int* OSGLParserData::matrixBlockNumberOfRows`

Definition at line 163 of file OSGLParserData.h.

6.129.3.87 `int* OSGLParserData::matrixBlockNumberOfCols`

Definition at line 164 of file OSGLParserData.h.

6.129.3.88 `ENUM_NL_EXPR_SHAPE OSGLParserData::shape`

Definition at line 166 of file OSGLParserData.h.

6.129.3.89 `bool OSGLParserData::shapePresent`

Definition at line 167 of file OSGLParserData.h.

6.129.3.90 `ENUM_CONREFERENCE_VALUETYPE OSGLParserData::valueType`

Definition at line 168 of file OSGLParserData.h.

6.129.3.91 `bool OSGLParserData::valueTypePresent`

Definition at line 169 of file OSGLParserData.h.

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

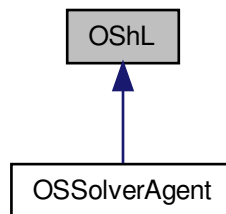
- </home/ted/COIN/trunk/OS/src/OSParsers/OSgLParserData.h>

6.130 OShL Class Reference

An interface that specified virtual methods to be implemented by agents.

```
#include "OShL.h"
```

Inheritance diagram for OShL:



Public Member Functions

- [OShL \(\)](#)
Default constructor.
- virtual [~OShL \(\)=0](#)
Class destructor.
- virtual std::string [solve](#) (std::string osil, std::string osol)=0
submit an instance with its options for a synchronous solution
- virtual std::string [getJobID](#) (std::string osol)=0
get a jobID for use in the send method
- virtual bool [send](#) (std::string osil, std::string osol)=0
submit an instance with its options for an asynchronous solution
- virtual std::string [kill](#) (std::string osol)=0
kill an instance that is running
- virtual std::string [retrieve](#) (std::string osol)=0
retrieve an instance result that ran in asynchronous mode
- virtual std::string [knock](#) (std::string ospl, std::string osol)=0
knock to get information on the current status of a job

6.130.1 Detailed Description

An interface that specified virtual methods to be implemented by agents.

Remarks

This is a virtual class that lists all of the methods a client (or scheduler/solver) should implement
 Definition at line 32 of file OShL.h.

6.130.2 Constructor & Destructor Documentation

6.130.2.1 OShL::OShL ()

Default constructor.

6.130.2.2 virtual OShL::~~OShL () [pure virtual]

Class destructor.

6.130.3 Member Function Documentation

6.130.3.1 virtual std::string OShL::solve (std::string *osil*, std::string *osol*) [pure virtual]

submit an instance with its options for a synchronous solution

Parameters

<i>osil</i>	is the string with the instance in OSiL format
<i>osol</i>	is the string with the options in OSoL format

Returns

a string which is the result in OSrL format.

Implemented in [OSSolverAgent](#).

6.130.3.2 virtual std::string OShL::getJobID (std::string *osol*) [pure virtual]

get a jobID for use in the send method

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is the jobID

Implemented in [OSSolverAgent](#).

6.130.3.3 virtual bool OShL::send (std::string *osil*, std::string *osol*) [pure virtual]

submit an instance with its options for an asynchronous solution

Parameters

<i>osil</i>	is the string with the instance in OSiL format
<i>osol</i>	is the string with the options in OSoL format

Returns

a bool which is true if the job is successfully submitted

Implemented in [OSSolverAgent](#).

6.130.3.4 `virtual std::string OShL::kill (std::string osol) [pure virtual]`

kill an instance that is running

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is in OSpL format

Implemented in [OSSolverAgent](#).

6.130.3.5 `virtual std::string OShL::retrieve (std::string osol) [pure virtual]`

retrieve an instance result that ran in asynchronous mode

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is in the result of the optimization is OSrL fomrat

Implemented in [OSSolverAgent](#).

6.130.3.6 `virtual std::string OShL::knock (std::string ospl, std::string osol) [pure virtual]`

knock to get information on the current status of a job

Parameters

<i>ospl</i>	is the string with the process information in OSpL format
<i>osol</i>	is the string with the options in OSoL format

Returns

a string which is the knock result in OSpL format.

Implemented in [OSSolverAgent](#).

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

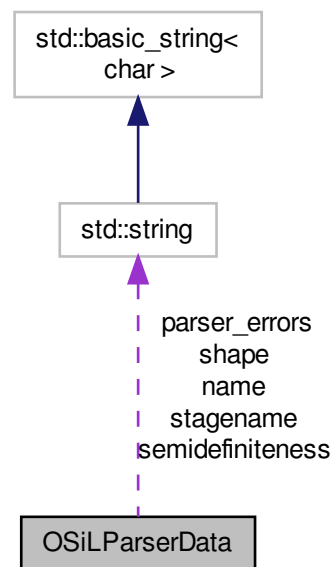
- [/home/ted/COIN/trunk/OS/src/OSAgent/OShL.h](#)

6.131 OSiLParserData Class Reference

The [OSiLParserData](#) Class, used to store parser data.

```
#include <OSiLParserData.h>
```

Collaboration diagram for OSiLParserData:



Public Member Functions

- [OSiLParserData](#) ()
the [OSiLParserData](#) class constructor
- [~OSiLParserData](#) ()
the [OSiLParserData](#) class destructor

Public Attributes

- void * [scanner](#)
scanner is used to store data in a reentrant lexer we use this to pass an [OSiLParserData](#) object to the parser
- int [osillinenno](#)
if there is a parser error, osillinenno holds the line number in the OSiL file where the error occurred.
- int [qtermcount](#)
These variables are used for processing the <quadraticCoefficients> element.

- bool [qtermidxOneattON](#)
qtermidxOneattON is true if we have found the first index of the quadratic term
- bool [qtermidxTwoattON](#)
qtermidxTwoattON is true if we have found the second index of the quadratic term
- bool [qtermidxattON](#)
qtermidxattON is true if we have found the row index of the of a quadratic term
- bool [qtermidattON](#)
qtermidattON is true if we have found the id of the quadratic term
- bool [qtermcoefattON](#)
qtermcoefattON is true if we have found the coefficient of the quadratic term
- bool [timeDomainStages](#)
These variables are used to parse the <timeDomain> element.
- bool [timeDomainInterval](#)
- int [stagecount](#)
store the number of stages
- bool [stagenameON](#)
stagenameON is true if the current stage was given a name
- std::string [stagename](#)
store the name of the current stage
- bool [stageVariablesON](#)
for each stage we need to track whether the <variables>, <constraints>, <objectives> elements are present...
- bool [stageConstraintsON](#)
- bool [stageObjectivesON](#)
- bool [stageVariablesOrdered](#)
...we need to track whether the variables, constraints, objectives are given in temporal order...
- bool [stageConstraintsOrdered](#)
- bool [stageObjectivesOrdered](#)
- int [stageVariableStartIdx](#)
...we need to track the first variable, constraint, objective...
- int [stageConstraintStartIdx](#)
- int [stageObjectiveStartIdx](#)
- int [stagevarcount](#)
...and we need to track the number of variables we have seen
- int [stageconcount](#)
- int [stageobjcount](#)
- int [nvarcovered](#)
these two integers track how many variables and constraints have been assigned to a stage; this is used for consistency checks.
- int [nconcovered](#)
- int * [m_miVarStageInfo](#)
m_miVarStageInfo is an array that for each variable gives the stage to which it belongs.
- int * [m_miConStageInfo](#)
m_miConStageInfo is an array that for each constraint gives the stage to which it belongs.
- int * [m_miObjStageInfo](#)
m_miObjStageInfo is an array that for each objective gives the stage to which it belongs.
- bool [intervalhorizonON](#)
intervalhorizonON is true if we have found a horizon for the time interval
- double [intervalhorizon](#)

intervalhorizon holds the value of the end of the planning horizon

- bool `intervalstartON`

intervalstartON is true if we have found a start time for the time interval

- double `intervalstart`

intervalstart holds the value for the start of the planning horizon

- bool `numberOfMatricesPresent`

some elements to hold matrices and cones

- int `numberOfMatrices`
- bool `numberOfConesPresent`
- int `numberOfCones`
- int `coneCounter`
- bool `numberOfRowsPresent`
- int `numberOfRows`
- bool `numberOfColumnsPresent`
- int `numberOfColumns`
- int `numberOfEI`
- int `numberOf`
- int `elCounter`
- bool `namePresent`
- std::string `name`
- bool `referenceMatrixIdxPresent`
- int `referenceMatrixIdx`
- bool `normScaleFactorPresent`
- double `normScaleFactor`
- bool `distortionMatrixPresent`
- int `distortionMatrix`
- bool `axisDirectionPresent`
- int `axisDirection`
- bool `firstAxisDirectionPresent`
- int `firstAxisDirection`
- bool `secondAxisDirectionPresent`
- int `secondAxisDirection`
- bool `semidefinitenessPresent`
- std::string `semidefiniteness`
- int `numberOfMatrixVar`
- int `numberOfMatrixObj`
- int `numberOfMatrixCon`
- int `numberOfMatrixTerms`
- int `numberOfMatrixExpr`
- bool `numberOfMatrixTermsPresent`
- bool `matrixIdxPresent`
- bool `lbMatrixIdxPresent`
- bool `lbConelIdxPresent`
- bool `ubMatrixIdxPresent`
- bool `ubConelIdxPresent`
- bool `orderConelIdxPresent`
- bool `templateMatrixIdxPresent`
- bool `constantMatrixIdxPresent`
- bool `varReferenceMatrixIdxPresent`
- bool `objReferenceMatrixIdxPresent`

- bool [conReferenceMatrixIdxPresent](#)
- bool [varTypePresent](#)
- int [matrixIdx](#)
- int [lbMatrixIdx](#)
- int [lbConIdx](#)
- int [ubMatrixIdx](#)
- int [ubConIdx](#)
- int [orderConIdx](#)
- int [templateMatrixIdx](#)
- int [constantMatrixIdx](#)
- int [varReferenceMatrixIdx](#)
- int [objReferenceMatrixIdx](#)
- int [conReferenceMatrixIdx](#)
- char [varType](#)
- bool [shapePresent](#)
- std::string [shape](#)
- bool [matrixTermInObj](#)
- int [kounter](#)
- int [kount2](#)
- double [tempVal](#)
- bool [ignoreDataAfterErrors](#)
if the parser finds invalid text it is held here and we delete if the file was not valid
- bool [suppressFurtherErrorMessage](#)
- char * [errorText](#)
- std::string [parser_errors](#)
used to accumulate error message so the parser does not die on the first error encountered

6.131.1 Detailed Description

The [OSiLParserData](#) Class, used to store parser data.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

The [OSiLParserData](#) class is used to hold the nonlinear part of the problem when an OSiL instance is parsed. We do this so we can have a reentrant parser. We do not have to store the linear part because we do not use flex/bison to parse the linear part of the problem.

Definition at line 34 of file [OSiLParserData.h](#).

6.131.2 Constructor & Destructor Documentation

6.131.2.1 OSiLParserData::OSiLParserData ()

the [OSiLParserData](#) class constructor

6.131.2.2 OSiLParserData::~~OSiLParserData ()

the [OSiLParserData](#) class destructor

6.131.3 Member Data Documentation

6.131.3.1 void* OSiLParserData::scanner

scanner is used to store data in a reentrant lexer we use this to pass an [OSiLParserData](#) object to the parser
Definition at line 45 of file OSiLParserData.h.

6.131.3.2 int OSiLParserData::osillineno

if there is a parser error, osillineno holds the line number in the OSiL file where the error occurred.
Definition at line 50 of file OSiLParserData.h.

6.131.3.3 int OSiLParserData::qtermcount

These variables are used for processing the <quadraticCoefficients> element.
store the number of quadratic terms
Definition at line 55 of file OSiLParserData.h.

6.131.3.4 bool OSiLParserData::qtermidxOneattON

qtermidxOneattON is true if we have found the first index of the quadratic term
Definition at line 59 of file OSiLParserData.h.

6.131.3.5 bool OSiLParserData::qtermidxTwoattON

qtermidxTwoattON is true if we have found the second index of the quadratic term
Definition at line 63 of file OSiLParserData.h.

6.131.3.6 bool OSiLParserData::qtermidxattON

qtermidxattON is true if we have found the row index of the of a quadratic term
Definition at line 67 of file OSiLParserData.h.

6.131.3.7 bool OSiLParserData::qtermidattON

qtermidattON is true if we have found the id of the quadratic term
Definition at line 70 of file OSiLParserData.h.

6.131.3.8 bool OSiLParserData::qtermcoefattON

qtermcoefattON is true if we have found the coefficient of the quadratic term

Definition at line 75 of file OSiLParserData.h.

6.131.3.9 bool OSiLParserData::timeDomainStages

These variables are used to parse the <timeDomain> element.

store the type of <timeDomain> (extend as needed)

Definition at line 82 of file OSiLParserData.h.

6.131.3.10 bool OSiLParserData::timeDomainInterval

Definition at line 83 of file OSiLParserData.h.

6.131.3.11 int OSiLParserData::stagecount

store the number of stages

Definition at line 86 of file OSiLParserData.h.

6.131.3.12 bool OSiLParserData::stagenameON

stagenameON is true if the current stage was given a name

Definition at line 89 of file OSiLParserData.h.

6.131.3.13 std::string OSiLParserData::stagename

store the name of the current stage

Definition at line 92 of file OSiLParserData.h.

6.131.3.14 bool OSiLParserData::stageVariablesON

for each stage we need to track whether the <variables>, <constraints>, <objectives> elements are present...

Definition at line 96 of file OSiLParserData.h.

6.131.3.15 bool OSiLParserData::stageConstraintsON

Definition at line 97 of file OSiLParserData.h.

6.131.3.16 bool OSiLParserData::stageObjectivesON

Definition at line 98 of file OSiLParserData.h.

6.131.3.17 bool OSiLParserData::stageVariablesOrdered

...we need to track whether the variables, constraints, objectives are given in temporal order...

Definition at line 102 of file OSiLParserData.h.

6.131.3.18 bool OSiLParserData::stageConstraintsOrdered

Definition at line 103 of file OSiLParserData.h.

6.131.3.19 bool OSiLParserData::stageObjectivesOrdered

Definition at line 104 of file OSiLParserData.h.

6.131.3.20 int OSiLParserData::stageVariableStartIdx

...we need to track the first variable, constraint, objective...

Definition at line 107 of file OSiLParserData.h.

6.131.3.21 int OSiLParserData::stageConstraintStartIdx

Definition at line 108 of file OSiLParserData.h.

6.131.3.22 int OSiLParserData::stageObjectiveStartIdx

Definition at line 109 of file OSiLParserData.h.

6.131.3.23 int OSiLParserData::stagevarcount

...and we need to track the number of variables we have seen

Definition at line 112 of file OSiLParserData.h.

6.131.3.24 int OSiLParserData::stageconcount

Definition at line 113 of file OSiLParserData.h.

6.131.3.25 int OSiLParserData::stageobjcount

Definition at line 114 of file OSiLParserData.h.

6.131.3.26 int OSiLParserData::nvarcovered

these two integers track how many variables and constraints have been assigned to a stage; this is used for consistency checks.

Definition at line 118 of file OSiLParserData.h.

6.131.3.27 int OSiLParserData::nconcovered

Definition at line 119 of file OSiLParserData.h.

6.131.3.28 int* OSiLParserData::m_miVarStageInfo

m_miVarStageInfo is an array that for each variable gives the stage to which it belongs.

Definition at line 124 of file OSiLParserData.h.

6.131.3.29 int* OSiLParserData::m_miConStageInfo

m_miConStageInfo is an array that for each constraint gives the stage to which it belongs.

Definition at line 129 of file OSiLParserData.h.

6.131.3.30 int* OSiLParserData::m_miObjStageInfo

m_miObjStageInfo is an array that for each objective gives the stage to which it belongs.

Definition at line 134 of file OSiLParserData.h.

6.131.3.31 bool OSiLParserData::intervalhorizonON

intervalhorizonON is true if we have found a horizon for the time interval

Definition at line 138 of file OSiLParserData.h.

6.131.3.32 double OSiLParserData::intervalhorizon

intervalhorizon holds the value of the end of the planning horizon

Definition at line 141 of file OSiLParserData.h.

6.131.3.33 bool OSiLParserData::intervalstartON

intervalstartON is true if we have found a start time for the time interval

Definition at line 145 of file OSiLParserData.h.

6.131.3.34 double OSiLParserData::intervalstart

intervalstart holds the value for the start of the planning horizon

Definition at line 148 of file OSiLParserData.h.

6.131.3.35 bool OSiLParserData::numberOfMatricesPresent

some elements to hold matrices and cones

Definition at line 151 of file OSiLParserData.h.

6.131.3.36 int OSiLParserData::numberOfMatrices

Definition at line 152 of file OSiLParserData.h.

6.131.3.37 bool OSiLParserData::numberOfConesPresent

Definition at line 153 of file OSiLParserData.h.

6.131.3.38 int OSiLParserData::numberOfCones

Definition at line 154 of file OSiLParserData.h.

6.131.3.39 int OSiLParserData::coneCounter

Definition at line 155 of file OSiLParserData.h.

6.131.3.40 bool OSiLParserData::numberOfRowsPresent

Definition at line 156 of file OSiLParserData.h.

6.131.3.41 int OSiLParserData::numberOfRows

Definition at line 157 of file OSiLParserData.h.

6.131.3.42 bool OSiLParserData::numberOfColumnsPresent

Definition at line 158 of file OSiLParserData.h.

6.131.3.43 int OSiLParserData::numberOfColumns

Definition at line 159 of file OSiLParserData.h.

6.131.3.44 `int OSiLParserData::numberOfEI`

Definition at line 161 of file OSiLParserData.h.

6.131.3.45 `int OSiLParserData::numberOf`

Definition at line 162 of file OSiLParserData.h.

6.131.3.46 `int OSiLParserData::elCounter`

Definition at line 163 of file OSiLParserData.h.

6.131.3.47 `bool OSiLParserData::namePresent`

Definition at line 165 of file OSiLParserData.h.

6.131.3.48 `std::string OSiLParserData::name`

Definition at line 166 of file OSiLParserData.h.

6.131.3.49 `bool OSiLParserData::referenceMatrixIdxPresent`

Definition at line 168 of file OSiLParserData.h.

6.131.3.50 `int OSiLParserData::referenceMatrixIdx`

Definition at line 169 of file OSiLParserData.h.

6.131.3.51 `bool OSiLParserData::normScaleFactorPresent`

Definition at line 171 of file OSiLParserData.h.

6.131.3.52 `double OSiLParserData::normScaleFactor`

Definition at line 172 of file OSiLParserData.h.

6.131.3.53 `bool OSiLParserData::distortionMatrixPresent`

Definition at line 173 of file OSiLParserData.h.

6.131.3.54 `int OSiLParserData::distortionMatrix`

Definition at line 174 of file OSiLParserData.h.

6.131.3.55 `bool OSiLParserData::axisDirectionPresent`

Definition at line 175 of file OSiLParserData.h.

6.131.3.56 `int OSiLParserData::axisDirection`

Definition at line 176 of file OSiLParserData.h.

6.131.3.57 `bool OSiLParserData::firstAxisDirectionPresent`

Definition at line 177 of file OSiLParserData.h.

6.131.3.58 `int OSiLParserData::firstAxisDirection`

Definition at line 178 of file OSiLParserData.h.

6.131.3.59 `bool OSiLParserData::secondAxisDirectionPresent`

Definition at line 179 of file OSiLParserData.h.

6.131.3.60 `int OSiLParserData::secondAxisDirection`

Definition at line 180 of file OSiLParserData.h.

6.131.3.61 `bool OSiLParserData::semidefinitenessPresent`

Definition at line 181 of file OSiLParserData.h.

6.131.3.62 `std::string OSiLParserData::semidefiniteness`

Definition at line 182 of file OSiLParserData.h.

6.131.3.63 `int OSiLParserData::numberOfMatrixVar`

Definition at line 185 of file OSiLParserData.h.

6.131.3.64 `int OSiLParserData::numberOfMatrixObj`

Definition at line 186 of file OSiLParserData.h.

6.131.3.65 `int OSiLParserData::numberOfMatrixCon`

Definition at line 187 of file OSiLParserData.h.

6.131.3.66 `int OSiLParserData::numberOfMatrixTerms`

Definition at line 188 of file OSiLParserData.h.

6.131.3.67 `int OSiLParserData::numberOfMatrixExpr`

Definition at line 189 of file OSiLParserData.h.

6.131.3.68 `bool OSiLParserData::numberOfMatrixTermsPresent`

Definition at line 191 of file OSiLParserData.h.

6.131.3.69 `bool OSiLParserData::matrixIdxPresent`

Definition at line 193 of file OSiLParserData.h.

6.131.3.70 `bool OSiLParserData::lbMatrixIdxPresent`

Definition at line 194 of file OSiLParserData.h.

6.131.3.71 `bool OSiLParserData::lbConelIdxPresent`

Definition at line 195 of file OSiLParserData.h.

6.131.3.72 bool OSiLParserData::ubMatrixIdxPresent

Definition at line 196 of file OSiLParserData.h.

6.131.3.73 bool OSiLParserData::ubConelIdxPresent

Definition at line 197 of file OSiLParserData.h.

6.131.3.74 bool OSiLParserData::orderConelIdxPresent

Definition at line 198 of file OSiLParserData.h.

6.131.3.75 bool OSiLParserData::templateMatrixIdxPresent

Definition at line 199 of file OSiLParserData.h.

6.131.3.76 bool OSiLParserData::constantMatrixIdxPresent

Definition at line 200 of file OSiLParserData.h.

6.131.3.77 bool OSiLParserData::varReferenceMatrixIdxPresent

Definition at line 201 of file OSiLParserData.h.

6.131.3.78 bool OSiLParserData::objReferenceMatrixIdxPresent

Definition at line 202 of file OSiLParserData.h.

6.131.3.79 bool OSiLParserData::conReferenceMatrixIdxPresent

Definition at line 203 of file OSiLParserData.h.

6.131.3.80 bool OSiLParserData::varTypePresent

Definition at line 204 of file OSiLParserData.h.

6.131.3.81 int OSiLParserData::matrixIdx

Definition at line 206 of file OSiLParserData.h.

6.131.3.82 int OSiLParserData::lbMatrixIdx

Definition at line 207 of file OSiLParserData.h.

6.131.3.83 int OSiLParserData::lbConelIdx

Definition at line 208 of file OSiLParserData.h.

6.131.3.84 int OSiLParserData::ubMatrixIdx

Definition at line 209 of file OSiLParserData.h.

6.131.3.85 int OSiLParserData::ubConelIdx

Definition at line 210 of file OSiLParserData.h.

6.131.3.86 `int OSiLParserData::orderConeldx`

Definition at line 211 of file OSiLParserData.h.

6.131.3.87 `int OSiLParserData::templateMatrixIdx`

Definition at line 212 of file OSiLParserData.h.

6.131.3.88 `int OSiLParserData::constantMatrixIdx`

Definition at line 213 of file OSiLParserData.h.

6.131.3.89 `int OSiLParserData::varReferenceMatrixIdx`

Definition at line 214 of file OSiLParserData.h.

6.131.3.90 `int OSiLParserData::objReferenceMatrixIdx`

Definition at line 215 of file OSiLParserData.h.

6.131.3.91 `int OSiLParserData::conReferenceMatrixIdx`

Definition at line 216 of file OSiLParserData.h.

6.131.3.92 `char OSiLParserData::varType`

Definition at line 217 of file OSiLParserData.h.

6.131.3.93 `bool OSiLParserData::shapePresent`

Definition at line 219 of file OSiLParserData.h.

6.131.3.94 `std::string OSiLParserData::shape`

Definition at line 220 of file OSiLParserData.h.

6.131.3.95 `bool OSiLParserData::matrixTermInObj`

Definition at line 222 of file OSiLParserData.h.

6.131.3.96 `int OSiLParserData::kounter`

Definition at line 224 of file OSiLParserData.h.

6.131.3.97 `int OSiLParserData::kount2`

Definition at line 225 of file OSiLParserData.h.

6.131.3.98 `double OSiLParserData::tempVal`

Definition at line 226 of file OSiLParserData.h.

6.131.3.99 `bool OSiLParserData::ignoreDataAfterErrors`

if the parser finds invalid text it is held here and we delete if the file was not valid

Definition at line 231 of file OSiLParserData.h.

6.131.3.100 bool OSiLParserData::suppressFurtherErrorMessages

Definition at line 232 of file OSiLParserData.h.

6.131.3.101 char* OSiLParserData::errorText

Definition at line 233 of file OSiLParserData.h.

6.131.3.102 std::string OSiLParserData::parser_errors

used to accumulate error message so the parser does not die on the first error encountered

Definition at line 238 of file OSiLParserData.h.

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

- /home/ted/COIN/trunk/OS/src/OSParsers/[OSiLParserData.h](#)

6.132 OSiLReader Class Reference

Used to read an OSiL string.

```
#include <OSiLReader.h>
```

Public Member Functions

- [OSiLReader](#) ()
Default constructor.
- [~OSiLReader](#) ()
Class destructor.
- [OSInstance](#) * [readOSiL](#) (const std::string &osil) throw (ErrorClass)
parse the OSiL model instance.

6.132.1 Detailed Description

Used to read an OSiL string.

Remarks

This class wraps around the OSiL parser and sends the parser an OSiL string and is returned an [OSInstance](#) object.

Definition at line 37 of file OSiLReader.h.

6.132.2 Constructor & Destructor Documentation

6.132.2.1 OSiLReader::OSiLReader ()

Default constructor.

6.132.2.2 OSiLReader::~~OSiLReader ()

Class destructor.

6.132.3 Member Function Documentation

6.132.3.1 OSInstance* OSiLReader::readOSiL (const std::string & osil) throw (ErrorClass)

parse the OSiL model instance.

Parameters

<i>osil</i>	a string that holds the problem instance.
-------------	---

Returns

the instance as an [OSInstance](#) object.

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

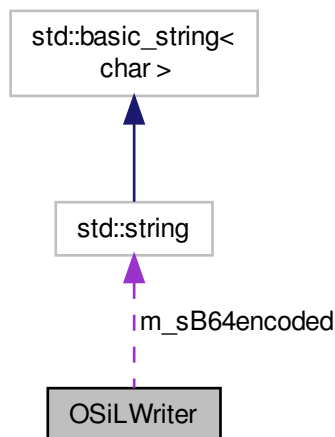
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSiLReader.h](#)

6.133 OSiLWriter Class Reference

Take an [OSInstance](#) object and write a string that validates against the OSiL schema.

```
#include "OSiLWriter.h"
```

Collaboration diagram for OSiLWriter:



Public Member Functions

- [OSiLWriter \(\)](#)
Default constructor.
- [~OSiLWriter \(\)](#)

Class destructor.

- `std::string writeOSiL (const OSInstance *theosinstance)`
create an osil string from an [OSInstance](#) object

Public Attributes

- `bool m_bWriteBase64`
m_bWriteBase64 is set to true if we encode the linear constraint coefficients in base64 binary
- `bool m_bWhiteSpace`
m_bWhiteSpace is set to true if we write white space in the file
- `std::string m_sB64encoded`
m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

6.133.1 Detailed Description

Take an [OSInstance](#) object and write a string that validates against the OSiL schema.

Definition at line 29 of file OSiLWriter.h.

6.133.2 Constructor & Destructor Documentation

6.133.2.1 OSiLWriter::OSiLWriter ()

Default constructor.

6.133.2.2 OSiLWriter::~~OSiLWriter ()

Class destructor.

6.133.3 Member Function Documentation

6.133.3.1 `std::string OSiLWriter::writeOSiL (const OSInstance * theosinstance)`

create an osil string from an [OSInstance](#) object

Parameters

<i>theosinstance</i>	is a pointer to an OSInstance object
----------------------	--

Returns

a string with the [OSInstance](#) data that validates against the OSiL schema.

6.133.4 Member Data Documentation

6.133.4.1 `bool OSiLWriter::m_bWriteBase64`

`m_bWriteBase64` is set to true if we encode the linear constraint coefficients in base64 binary

Definition at line 64 of file OSiLWriter.h.

6.133.4.2 bool OSILWriter::m_bWhiteSpace

m_bWhiteSpace is set to true if we write white space in the file

Definition at line 68 of file OSILWriter.h.

6.133.4.3 std::string OSILWriter::m_sB64encoded

m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

Definition at line 73 of file OSILWriter.h.

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

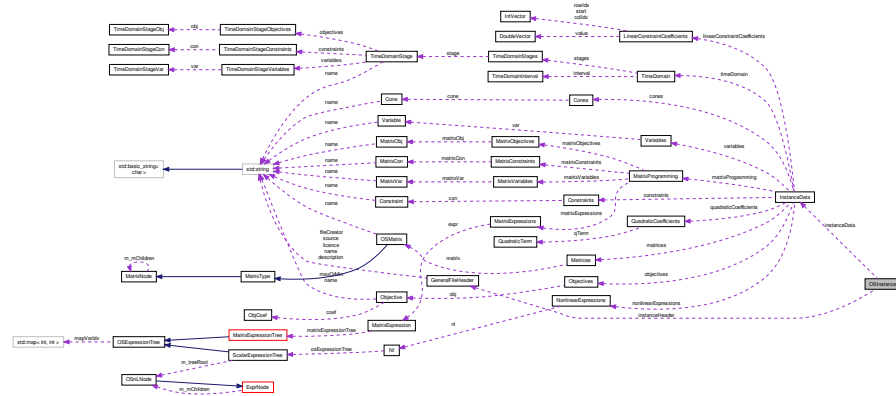
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSILWriter.h

6.134 OSInstance Class Reference

The in-memory representation of an OSiL instance.

```
#include "OSInstance.h"
```

Collaboration diagram for OSInstance:



Public Member Functions

- **OSInstance** ()
The *OSInstance* class constructor.
- **~OSInstance** ()
The *OSInstance* class destructor.
- bool **IsEqual** (*OSInstance* *that)
A function to check for the equality of two objects.
- std::string **getInstanceName** ()
Get instance name.
- std::string **getInstanceSource** ()
Get instance source.
- std::string **getInstanceDescription** ()

- Get instance description.*

 - `std::string getInstanceCreator ()`

Get instance fileCreator.

 - `std::string getInstanceLicence ()`

Get instance licence.

 - `int getVariableNumber ()`

Get number of variables.

 - `std::string * getVariableNames ()`

Get variable names.

 - `char * getVariableTypes ()`

Get variable initial values.

 - `int getNumberOfIntegerVariables ()`

getNumberOfIntegerVariables

 - `int getNumberOfBinaryVariables ()`

getNumberOfBinaryVariables

 - `int getNumberOfSemiContinuousVariables ()`

getNumberOfSemiContinuousVariables

 - `int getNumberOfSemIntegerVariables ()`

getNumberOfSemIntegerVariables

 - `int getNumberOfStringVariables ()`

getNumberOfStringVariables

 - `double * getVariableLowerBounds ()`

Get variable lower bounds.

 - `double * getVariableUpperBounds ()`

Get variable upper bounds.

 - `int getObjectiveNumber ()`

Get number of objectives.

 - `std::string * getObjectiveNames ()`

Get objective names.

 - `std::string * getObjectiveMaxOrMins ()`

Get objective maxOrMins.

 - `int * getObjectiveCoefficientNumbers ()`

Get objective coefficient number.

 - `double * getObjectiveConstants ()`

Get objective constants.

 - `double * getObjectiveWeights ()`

Get objective weights.

 - `SparseVector ** getObjectiveCoefficients ()`

Get objective coefficients.

 - `double ** getDenseObjectiveCoefficients ()`

getDenseObjectiveCoefficients.

 - `int getConstraintNumber ()`

Get number of constraints.

 - `std::string * getConstraintNames ()`

Get constraint names.

 - `double * getConstraintLowerBounds ()`

Get constraint lower bounds.

- double * [getConstraintUpperBounds](#) ()
Get constraint upper bounds.
- double * [getConstraintConstants](#) ()
Get constraint constants.
- char * [getConstraintTypes](#) ()
Get constraint types.
- int [getLinearConstraintCoefficientNumber](#) ()
Get number of specified (usually nonzero) linear constraint coefficient values.
- bool [getLinearConstraintCoefficientMajor](#) ()
Get whether the constraint coefficients is in column major (true) or row major (false).
- [SparseMatrix](#) * [getLinearConstraintCoefficientsInColumnMajor](#) ()
Get linear constraint coefficients in column major.
- [SparseMatrix](#) * [getLinearConstraintCoefficientsInRowMajor](#) ()
Get linear constraint coefficients in row major.
- int [getNumberOfQuadraticTerms](#) ()
Get the number of specified (usually nonzero) qTerms in the quadratic coefficients.
- [QuadraticTerms](#) * [getQuadraticTerms](#) ()
Get all the quadratic terms in the instance.
- int * [getQuadraticRowIndexes](#) ()
Get the indexes of rows which have a quadratic term.
- int [getNumberOfQuadraticRowIndexes](#) ()
Get the number of rows which have a quadratic term.
- int [getNumberOfNonlinearExpressions](#) ()
Get number of nonlinear expressions.
- [NI](#) ** [getNonlinearExpressions](#) ()
Get the pointers to the roots of all expression trees.
- [ScalarExpressionTree](#) * [getNonlinearExpressionTree](#) (int rowIdx)
Get the expression tree for a given row index.
- [ScalarExpressionTree](#) * [getNonlinearExpressionTreeMod](#) (int rowIdx)
Get the expression tree for a given row index for the modified expression trees (quadratic terms added)
- std::vector< [ExprNode](#) * > [getNonlinearExpressionTreeInPostfix](#) (int rowIdx)
Get the postfix tokens for a given row index.
- std::vector< [ExprNode](#) * > [getNonlinearExpressionTreeModInPostfix](#) (int rowIdx)
Get the postfix tokens for a given row index for the modified Expression Tree (quadratic terms added).
- std::vector< [ExprNode](#) * > [getNonlinearExpressionTreeInPrefix](#) (int rowIdx)
Get the prefix tokens for a given row index.
- std::string [getNonlinearExpressionTreeInInfix](#) (int rowIdx)
Get the infix representation for a given row (or objective function) index.
- std::vector< [ExprNode](#) * > [getNonlinearExpressionTreeModInPrefix](#) (int rowIdx)
Get the prefix tokens for a given row index for the modified Expression Tree (quadratic terms added).
- int [getNumberOfNonlinearObjectives](#) ()
- int [getNumberOfNonlinearConstraints](#) ()
- std::map< int,
 [ScalarExpressionTree](#) * > [getAllNonlinearExpressionTrees](#) ()
- std::map< int,
 [ScalarExpressionTree](#) * > [getAllNonlinearExpressionTreesMod](#) ()
- int * [getNonlinearExpressionTreeIndexes](#) ()

Get all the nonlinear expression tree indexes, i.e., indexes of rows (objectives or constraints) that contain nonlinear expressions.

- int [getNumberOfNonlinearExpressionTreeIndexes](#) ()

Get the number of unique nonlinear expression tree indexes.

- int * [getNonlinearExpressionTreeModIndexes](#) ()

Get all the nonlinear expression tree indexes, i.e., indexes of rows (objectives or constraints) that contain nonlinear expressions after modifying the expression tree to contain quadratic terms.

- int [getNumberOfNonlinearExpressionTreeModIndexes](#) ()

Get the number of unique nonlinear expression tree indexes after modifying the expression tree to contain quadratic terms.

- int [getMatrixNumber](#) ()

Get the number of matrices.

- [ENUM_MATRIX_TYPE](#) [getMatrixType](#) (int n)

Get the matrix type.

- [ENUM_MATRIX_SYMMETRY](#) [getMatrixSymmetry](#) (int n)

Get the matrix symmetry.

- int [getNumberOfColumnsForMatrix](#) (int n)

Get the number of blocks in the matrix.

- int [getNumberOfRowsForMatrix](#) (int n)

Get the number of rows in the matrix.

- std::string [getMatrixName](#) (int n)

Get the number of (nonzero) values in the matrix.

- bool [matrixHasBase](#) (int n)

Several tools to parse the constructor list of a matrix.

- bool [matrixHasElements](#) (int n)

- bool [matrixHasTransformations](#) (int n)

- bool [matrixHasBlocks](#) (int n)

- int [getNumberOfElementConstructors](#) (int n)

- int [getNumberOfTransformationConstructors](#) (int n)

- int [getNumberOfBlocksConstructors](#) (int n)

- [OSMatrix](#) * [getMatrix](#) (int n)

Get the list of constructors of the matrix.

- [GeneralSparseMatrix](#) * [getMatrixCoefficientsInColumnMajor](#) (int n)

Get the (nonzero) elements of the matrix in column major form.

- [GeneralSparseMatrix](#) * [getMatrixCoefficientsInRowMajor](#) (int n)

Get the (nonzero) elements of the matrix in row major form.

- [GeneralSparseMatrix](#) * [getMatrixBlockInColumnMajorForm](#) (int n, int columnIdx, int rowIdx)

Get the (nonzero) elements of the matrix in symmetric block form.

- int [getNumberOfMatrixVariables](#) ()

Get the number of matrix variables.

- int [getNumberOfMatrixObjectives](#) ()

Get the number of matrix objectives.

- int [getNumberOfMatrixConstraints](#) ()

Get the number of matrix constraints.

- int [getNumberOfMatrixExpressions](#) ()

Get the number of matrix-valued expressions.

- [MatrixExpression](#) ** [getMatrixExpressions](#) ()

Get the pointers to the roots of all matrix expression trees.

- [MatrixExpressionTree](#) * [getMatrixExpressionTree](#) (int rowIdx)

- Get the matrix expression tree for a given row index.*

 - `std::vector< ExprNode * > getMatrixExpressionTreeInPostfix (int rowIdx)`
- Get the postfix tokens for a given row index.*

 - `std::vector< ExprNode * > getMatrixExpressionTreeModInPostfix (int rowIdx)`
- Get the postfix tokens for a given row index for the modified Expression Tree (quadratic terms added).*

 - `std::vector< ExprNode * > getMatrixExpressionTreeInPrefix (int rowIdx)`
- Get the prefix tokens for a given row index.*

 - `std::string getMatrixExpressionTreeInInfix (int rowIdx)`
- Get the infix representation for a given row (or objective function) index.*

 - `std::map< int, MatrixExpressionTree * > getAllMatrixExpressionTrees ()`
- Get the modified infix representation for a given row (or objective function) index.*

 - `std::map< int, MatrixExpressionTree * > getAllMatrixExpressionTreesMod ()`
- Get all the matrix expression tree indexes, i.e.*

 - `int * getMatrixExpressionTreeIndexes ()`
- Get the number of unique matrix expression tree indexes.*

 - `int getNumberOfMatrixExpressionTreeIndexes ()`
- Get the format of the time domain ("stages"/"interval")*

 - `std::string getTimeDomainFormat ()`
- Get the number of stages that make up the time domain.*

 - `int getTimeDomainStageNumber ()`
- Get the names of the stages (NULL or empty string ("") if a stage has not been given a name.*

 - `std::string * getTimeDomainStageNames ()`
- Get the number of variables contained in each time stage.*

 - `int * getTimeDomainStageNumberOfVariables ()`
- Get the number of constraints contained in each time stage.*

 - `int * getTimeDomainStageNumberOfConstraints ()`
- Get the number of objectives contained in each time stage.*

 - `int * getTimeDomainStageNumberOfObjectives ()`
- Get the list of variables in each stage.*

 - `int ** getTimeDomainStageVarList ()`
- Get the list of constraints in each stage.*

 - `int ** getTimeDomainStageConList ()`
- Get the list of objectives in each stage.*

 - `int ** getTimeDomainStageObjList ()`
- Get the start for the time domain interval.*

 - `double getTimeDomainIntervalStart ()`
- Get the horizon for the time domain interval.*

 - `double getTimeDomainIntervalHorizon ()`
- set the instance name.*

 - `bool setInstanceName (std::string name)`
- set the instance source.*

 - `bool setInstanceSource (std::string source)`
- set the instance description.*

 - `bool setInstanceDescription (std::string description)`
- set the instance creator.*

 - `bool setInstanceCreator (std::string fileCreator)`

- bool [setInstanceLicence](#) (std::string licence)
set the instance licence.
- bool [setVariableNumber](#) (int number)
set the number of variables.
- bool [addVariable](#) (int index, std::string name, double lowerBound, double upperBound, char type)
add a variable.
- bool [setVariables](#) (int number, std::string *names, double *lowerBounds, double *upperBounds, char *types)
set all the variable related elements.
- bool [setObjectiveNumber](#) (int number)
set the number of objectives.
- bool [addObjective](#) (int index, std::string name, std::string maxOrMin, double constant, double weight, [SparseVector](#) *objectiveCoefficients)
add an objective.
- bool [setObjectives](#) (int number, std::string *names, std::string *maxOrMins, double *constants, double *weights, [SparseVector](#) **objectiveCoefficients)
set all the objectives related elements.
- bool [setConstraintNumber](#) (int number)
set the number of constraints.
- bool [addConstraint](#) (int index, std::string name, double lowerBound, double upperBound, double constant)
add a constraint.
- bool [setConstraints](#) (int number, std::string *names, double *lowerBounds, double *upperBounds, double *constants)
set all the constraint related elements.
- bool [setLinearConstraintCoefficients](#) (int numberOfValues, bool isColumnMajor, double *values, int valuesBegin, int valuesEnd, int *indexes, int indexesBegin, int indexesEnd, int *starts, int startsBegin, int startsEnd)
set linear constraint coefficients
- bool [copyLinearConstraintCoefficients](#) (int numberOfValues, bool isColumnMajor, double *values, int valuesBegin, int valuesEnd, int *indexes, int indexesBegin, int indexesEnd, int *starts, int startsBegin, int startsEnd)
copy linear constraint coefficients: perform a deep copy of the sparse matrix
- bool [setNumberOfQuadraticTerms](#) (int nq)
set the number of quadratic terms
- bool [setQuadraticCoefficients](#) (int number, int *rowIndexes, int *varOneIndexes, int *varTwoIndexes, double *coefficients, int begin, int end)
set quadratic coefficients into the QuadraticCoefficients->qTerm data structure
- bool [setQuadraticTermsInNonlinearExpressions](#) (int number, int *rowIndexes, int *varOneIndexes, int *varTwoIndexes, double *coefficients)
set quadratic terms in nonlinearExpressions
- bool [setNonlinearExpressions](#) (int nexpr, [NI](#) **root)
set nonlinear expressions
- bool [setMatrixNumber](#) (int number)
set the number of matrices
- bool [addMatrix](#) (int index, std::string name, int numberOfRows, int numberOfColumns, [ENUM_MATRIX_SYMMETRY](#) symmetry, [ENUM_MATRIX_TYPE](#) matrixType, unsigned int inumberOfChildren, [MatrixNode](#) **m_mChildren)
add a matrix.
- bool [setConeNumber](#) (int number)
set the number of cones
- bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int numberOfOtherIndexes=0, int *otherIndexes=NULL)

- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int numberOfComponents, int *components, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int referenceIdx, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, std::string semidefiniteness, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int distortionMatrixIdx, double normFactor, int axisDirection, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int distortionMatrixIdx, double normFactor, int firstAxisDirection, int secondAxisDirection, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - bool [addCone](#) (int index, int numberOfRows, int numberOfColumns, [ENUM_CONE_TYPE](#) coneType, std::string name, int maxDegree, int numberOfUB, double *ub, int numberOfLB, double *lb, int numberOfOtherIndexes=0, int *otherIndexes=NULL)
- add a cone.*

 - std::string [printModel](#) ()

Print the infix representation of the problem.
- std::string [printModel](#) (int rowIdx)

Print the infix representation of the row (which could be an an objective function row) indexed by rowIdx.
- bool [initializeNonLinearStructures](#) ()

Initialize the data structures for the nonlinear API.
- double [calculateFunctionValue](#) (int idx, double *x, bool new_x)

Calculate the function value for function (constraint or objective) indexed by idx.
- double * [calculateAllConstraintFunctionValues](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)

Calculate all of the constraint function values.
- double * [calculateAllConstraintFunctionValues](#) (double *x, bool new_x)

Calculate all of the constraint function values, we are overloading this function and this version of the method will not use any AD and will evaluate function values from the OS Expression Tree.
- double * [calculateAllObjectiveFunctionValues](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)

Calculate all of the objective function values.
- double * [calculateAllObjectiveFunctionValues](#) (double *x, bool new_x)

Calculate all of the objective function values, we are overloading this function and this version of the method will not use any AD and will evaluate function values from the OS Expression Tree.
- [SparseJacobianMatrix](#) * [calculateAllConstraintFunctionGradients](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)

Calculate the gradient of all constraint functions.

- [SparseVector](#) * [calculateConstraintFunctionGradient](#) (double *x, double *objLambda, double *conLambda, int idx, bool new_x, int highestOrder)
Calculate the gradient of the constraint function indexed by idx.
- [SparseVector](#) * [calculateConstraintFunctionGradient](#) (double *x, int idx, bool new_x)
Calculate the gradient of the constraint function indexed by idx this function is overloaded.
- double ** [calculateAllObjectiveFunctionGradients](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)
Calculate the gradient of all objective functions.
- double * [calculateObjectiveFunctionGradient](#) (double *x, double *objLambda, double *conLambda, int objIdx, bool new_x, int highestOrder)
Calculate the gradient of the objective function indexed by objIdx.
- double * [calculateObjectiveFunctionGradient](#) (double *x, int objIdx, bool new_x)
Calculate the gradient of the objective function indexed by objIdx this function is overloaded.
- [SparseHessianMatrix](#) * [calculateLagrangianHessian](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)
Calculate the Hessian of the Lagrangian Expression Tree This method will build the CppAD expression tree for only the first iteration Use this method on if the value of x does not affect the operations sequence.
- [SparseHessianMatrix](#) * [calculateHessian](#) (double *x, int idx, bool new_x)
Calculate the Hessian of a constraint or objective function.
- bool [getSparseJacobianFromColumnMajor](#) ()
- bool [getSparseJacobianFromRowMajor](#) ()
- [ScalarExpressionTree](#) * [getLagrangianExpTree](#) ()
- std::map< int, int > [getAllNonlinearVariablesIndexMap](#) ()
- [SparseHessianMatrix](#) * [getLagrangianHessianSparsityPattern](#) ()
- bool [addQTermsToExressionTree](#) ()
- bool [addQTermsToExpressionTree](#) ()
This method adds quadratic terms into the array of expression trees.
- [SparseJacobianMatrix](#) * [getJacobianSparsityPattern](#) ()
- void [duplicateExpressionTreesMap](#) ()
duplicate the map of expression trees.
- bool [createOSADFun](#) (std::vector< double > vdX)
Create the a CppAD Function object: this is a function where the domain is the set of variables for the problem and the range is the objective function plus constraints.
- std::vector< double > [forwardAD](#) (int p, std::vector< double > vdX)
Perform an AD forward sweep.
- std::vector< double > [reverseAD](#) (int p, std::vector< double > vdlambda)
Perform an AD reverse sweep.
- int [getADSparsityHessian](#) ()
end revised AD code
- bool [getIterateResults](#) (double *x, double *objLambda, double *conLambda, bool new_x, int highestOrder)
end revised AD code
- bool [getZeroOrderResults](#) (double *x, double *objLambda, double *conLambda)
Calculate function values.
- bool [getFirstOrderResults](#) (double *x, double *objLambda, double *conLambda)
Calculate first derivatives.
- bool [getSecondOrderResults](#) (double *x, double *objLambda, double *conLambda)
Calculate second derivatives.
- bool [initForAlgDiff](#) ()

This should be called by nonlinear solvers using callback functions.

- bool [initObjGradients](#) ()

This should be called by [initForAlgDiff\(\)](#)

- bool [setTimeDomain](#) (std::string format)

This sets the format of the time domain ("stages"/"interval"/"none")

- bool [setTimeDomainStages](#) (int number, std::string *names)

This sets the number (and optionally names) of the time stages.

- bool [setTimeDomainStageVariablesOrdered](#) (int numberOfStages, int *numberOfVariables, int *startIdx)

This sets the variables associated with each time domain stage in temporal order.

- bool [setTimeDomainStageVariablesUnordered](#) (int numberOfStages, int *numberOfVariables, int **varIndex)

This sets the variables associated with each time domain stage in arbitrary order.

- bool [setTimeDomainStageConstraintsOrdered](#) (int numberOfStages, int *numberOfConstraints, int *startIdx)

This sets the constraints associated with each time domain stage in temporal order.

- bool [setTimeDomainStageConstraintsUnordered](#) (int numberOfStages, int *numberOfConstraints, int **con-Index)

This sets the constraints associated with each time domain stage in arbitrary order.

- bool [setTimeDomainStageObjectivesOrdered](#) (int numberOfStages, int *numberOfObjectives, int *startIdx)

This sets the objectives associated with each time domain stage in temporal order.

- bool [setTimeDomainStageObjectivesUnordered](#) (int numberOfStages, int *numberOfObjectives, int **varIndex)

This sets the objectives associated with each time domain stage in arbitrary order.

- bool [setTimeDomainInterval](#) (double start, double horizon)

This sets the start and end of the time interval.

Public Attributes

- [GeneralFileHeader](#) * [instanceHeader](#)

the instanceHeader is implemented as a general file header object to allow sharing of classes between schemas

- [InstanceData](#) * [instanceData](#)

A pointer to an [InstanceData](#) object.

- bool [bVariablesModified](#)

bVariablesModified is true if the variables data has been modified.

- bool [bObjectivesModified](#)

bObjectivesModified is true if the objective function data has been modified.

- bool [bConstraintsModified](#)

bConstraintsModified is true if the constraints data has been modified.

- bool [bAMatrixModified](#)

bAMatrixModified is true if the A matrix data has been modified.

- bool [bUseExpTreeForFunEval](#)

bUseExpTreeForFunEval is set to true if you wish to use the OS Expression Tree for function evaluations instead of AD – false by default.

6.134.1 Detailed Description

The in-memory representation of an OSiL instance.

Remarks

1. Elements become objects of class type (the ComplexType is the class)
2. The attributes, children of the element, and text correspond to members of the class. (Note text does not have a name and becomes .value)
3. Model groups such as choice and sequence and all correspond to arrays
 1. anything specific to XML such as base64, multi, incr do not go into classes
 2. The root [OSnLNode](#) of each <nl> element is called ExpressionTree
 3. Root is not called osil it is called osinstance

The [OSInstance](#) class is composed of two objects: the header object instanceHeader and the data object instanceData
Definition at line 2264 of file OSInstance.h.

6.134.2 Constructor & Destructor Documentation

6.134.2.1 OSInstance::OSInstance ()

The [OSInstance](#) class constructor.

6.134.2.2 OSInstance::~~OSInstance ()

The [OSInstance](#) class destructor.

6.134.3 Member Function Documentation

6.134.3.1 bool OSInstance::IsEqual (OSInstance * that)

A function to check for the equality of two objects.

6.134.3.2 std::string OSInstance::getInstanceName ()

Get instance name.

Returns

instance name. Null or empty std::string ("") if there is no instance name.

6.134.3.3 std::string OSInstance::getInstanceSource ()

Get instance source.

Returns

instance source. Null or empty std::string ("") if there is no instance source.

6.134.3.4 `std::string OSInstance::getInstanceDescription ()`

Get instance description.

Returns

instance description. Null or empty `std::string ("")` if there is no instance description.

6.134.3.5 `std::string OSInstance::getInstanceCreator ()`

Get instance fileCreator.

Returns

instance fileCreator. Null or empty `std::string ("")` if there is no instance file creator.

6.134.3.6 `std::string OSInstance::getInstanceLicence ()`

Get instance licence.

Returns

instance licence. Null or empty `std::string ("")` if there is no instance licence.

6.134.3.7 `int OSInstance::getVariableNumber ()`

Get number of variables.

Returns

number of variables.

6.134.3.8 `std::string* OSInstance::getVariableNames ()`

Get variable names.

Returns

a `std::string` array of variable names, null if no variable names.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent.
------------------	--

6.134.3.9 `char* OSInstance::getVariableTypes ()`

Get variable initial values.

Returns

a double array of variable initial values, null if no initial variable values.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent. – now deprecated Get variable initial std::string values.
------------------	--

Returns

a std::string array of variable initial values, null if no initial variable std::string values.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent. – now deprecated Get variable types. <ul style="list-style-type: none"> • C for Continuous • B for Binary • I for Integer • S for String
------------------	---

Returns

a char array of variable types.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent.
------------------	--

6.134.3.10 int OSInstance::getNumberOfIntegerVariables ()

getNumberOfIntegerVariables

Returns

an integer which is the number of I variables.

6.134.3.11 int OSInstance::getNumberOfBinaryVariables ()

getNumberOfBinaryVariables

Returns

an integer which is the number of B variables.

6.134.3.12 int OSInstance::getNumberOfSemiContinuousVariables ()

getNumberOfSemiContinuousVariables

Returns

an integer which is the number of D variables.

6.134.3.13 int OSInstance::getNumberOfSemiIntegerVariables ()

getNumberOfSemiIntegerVariables

Returns

an integer which is the number of J variables.

6.134.3.14 `int OSInstance::getNumberOfStringVariables ()`

getNumberOfStringVariables

Returns

an integer which is the number of S variables.

6.134.3.15 `double* OSInstance::getVariableLowerBounds ()`

Get variable lower bounds.

Returns

a double array of variable lower bounds.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent.
------------------	--

6.134.3.16 `double* OSInstance::getVariableUpperBounds ()`

Get variable upper bounds.

Returns

a double array of variable upper bounds.

Exceptions

<i>Exception</i>	if the elements in variables are logically inconsistent.
------------------	--

6.134.3.17 `int OSInstance::getObjectiveNumber ()`

Get number of objectives.

Returns

number of objectives.

6.134.3.18 `std::string* OSInstance::getObjectiveNames ()`

Get objective names.

Returns

a std::string array of objective names. Null if no objective names.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.19 std::string* OSInstance::getObjectiveMaxOrMins ()

Get objective maxOrMins.

One maxOrMin for each objective.

Returns

a std::string array of objective maxOrMins ("max" or "min"), null if no objectives.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.20 int* OSInstance::getObjectiveCoefficientNumbers ()

Get objective coefficient number.

One number for each objective.

Returns

an integer array of size of which is equal to number of objectives, each element of the array is the number of nonzero coefficients in that objective function, null if no objectives.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.21 double* OSInstance::getObjectiveConstants ()

Get objective constants.

One constant for each objective.

Returns

a double array of objective constants, null if no objectives.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.22 double* OSInstance::getObjectiveWeights ()

Get objective weights.

One weight for each objective.

Returns

a double array of objective weights, null if no objectives.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.23 SparseVector OSInstance::getObjectiveCoefficients ()**

Get objective coefficients.

One set of objective coefficients for each objective.

See Also

org.optimizationservices.oscommon.datastructure.SparseVector

Returns

an array of objective coefficients, null if no objectives. Each member of the array is of type ObjectiveCoefficients. The ObjectiveCoefficients class contains two arrays: variableIndexes is an integer array and values is a double array of coefficient values.

Exceptions

<i>Exception</i>	if the elements in objectives are logically inconsistent.
------------------	---

6.134.3.24 double OSInstance::getDenseObjectiveCoefficients ()**

getDenseObjectiveCoefficients.

Returns

an vector of pointers, each pointer points to a dense vector of ObjectiveCoefficients.

6.134.3.25 int OSInstance::getConstraintNumber ()

Get number of constraints.

Returns

number of constraints.

6.134.3.26 std::string* OSInstance::getConstraintNames ()

Get constraint names.

Returns

a std::string array of constraint names, null if no constraint names.

Exceptions

<i>Exception</i>	if the elements in constraints are logically inconsistent.
------------------	--

6.134.3.27 `double* OSInstance::getConstraintLowerBounds ()`

Get constraint lower bounds.

Returns

a double array of constraint lower bounds, null if no constraints.

Exceptions

<i>Exception</i>	if the elements in constraints are logically inconsistent.
------------------	--

6.134.3.28 `double* OSInstance::getConstraintUpperBounds ()`

Get constraint upper bounds.

Returns

a double array of constraint upper bounds, null if no constraints.

Exceptions

<i>Exception</i>	if the elements in constraints are logically inconsistent.
------------------	--

6.134.3.29 `double* OSInstance::getConstraintConstants ()`

Get constraint constants.

Returns

a double array of constraint constants, null if no constraints.

Exceptions

<i>Exception</i>	if the elements in constraints are logically inconsistent.
------------------	--

6.134.3.30 `char* OSInstance::getConstraintTypes ()`

Get constraint types.

The constraint types are not part of the OSiL schema, but they are used in solver interfaces such as OSLindoSolver.cpp.

- R for range constraint $lb \leq \text{constraint} \leq ub$
- L for less than constraint $-INF \leq \text{con} \leq ub$ or $\text{con} \leq ub$
- G for greater than constraint $lb \leq \text{con} \leq INF$ or $\text{con} \geq lb$
- E for equal to constraint $lb \leq \text{con} \leq ub$ where $lb = ub$ or $\text{con} = lb$ (or $\text{con} = ub$)
- U for unconstrained constraint $-INF \leq \text{con} \leq INF$

Returns

a char array of constraint types, null if no constraints.

Exceptions

<i>Exception</i>	if the elements in constraints are logically inconsistent.
------------------	--

6.134.3.31 int OSInstance::getLinearConstraintCoefficientNumber ()

Get number of specified (usually nonzero) linear constraint coefficient values.

Returns

number of specified (usually nonzero) linear constraint coefficient values.

6.134.3.32 bool OSInstance::getLinearConstraintCoefficientMajor ()

Get whether the constraint coefficients is in column major (true) or row major (false).

Returns

whether the constraint coefficients is in column major (true) or row major (false).

Exceptions

<i>Exception</i>	if the elements in linear constraint coefficients are logically inconsistent.
------------------	---

6.134.3.33 SparseMatrix* OSInstance::getLinearConstraintCoefficientsInColumnMajor ()

Get linear constraint coefficients in column major.

Returns

a sparse matrix representation of linear constraint coefficients in column major, null if no linear constraint coefficients.

Exceptions

<i>Exception</i>	if the elements in linear constraint coefficients are logically inconsistent.
------------------	---

See Also

org.optimizationservices.oscommon.datastructure.SparseMatrix

6.134.3.34 SparseMatrix* OSInstance::getLinearConstraintCoefficientsInRowMajor ()

Get linear constraint coefficients in row major.

Returns

a sparse matrix representation of linear constraint coefficients in row major, null if no linear constraint coefficients.

Exceptions

<i>Exception</i>	if the elements in linear constraint coefficients are logically inconsistent.
------------------	---

See Also

org.optimizationservices.oscommon.datastructure.SparseMatrix

6.134.3.35 int OSInstance::getNumberOfQuadraticTerms ()

Get the number of specified (usually nonzero) qTerms in the quadratic coefficients.

Returns

qTerm number.

6.134.3.36 QuadraticTerms* OSInstance::getQuadraticTerms ()

Get all the quadratic terms in the instance.

Returns

the [QuadraticTerms](#) data structure for all quadratic terms in the instance, null if no quadratic terms. The [QuadraticTerms](#) contains four arrays: rowIndexes, varOneIndexes, varTwoIndexes, coefficients.

Exceptions

<i>Exception</i>	if the elements in quadratic coefficients are logically inconsistent.
------------------	---

See Also

org.optimizationservices.oscommon.datastructure.QuadraticTerms

6.134.3.37 int* OSInstance::getQuadraticRowIndexes ()

Get the indexes of rows which have a quadratic term.

Returns

an integer pointer to the row indexes of rows with quadratic terms, objectives functions have index < 0 NULL if there are no quadratic terms.

6.134.3.38 int OSInstance::getNumberOfQuadraticRowIndexes ()

Get the number of rows which have a quadratic term.

Returns

an integer which is the number of distinct rows (including obj) with quadratic terms,

6.134.3.39 int OSInstance::getNumberOfNonlinearExpressions ()

Get number of nonlinear expressions.

Returns

the number of nonlinear expressions.

6.134.3.40 NI OSInstance::getNonlinearExpressions ()**

Get the pointers to the roots of all expression trees.

Returns

an array of pointers to [NI](#) objects

6.134.3.41 ScalarExpressionTree* OSInstance::getNonlinearExpressionTree (int rowIdx)

Get the expression tree for a given row index.

Returns

an expression tree

6.134.3.42 ScalarExpressionTree* OSInstance::getNonlinearExpressionTreeMod (int rowIdx)

Get the expression tree for a given row index for the modified expression trees (quadratic terms added)

Returns

an expression tree

6.134.3.43 std::vector<ExprNode*> OSInstance::getNonlinearExpressionTreeInPostfix (int rowIdx)

Get the postfix tokens for a given row index.

Returns

a vector of pointers to ExprNodes in postfix, if rowIdx does not index a row with a nonlinear term throw an exception

Remarks

The root node of the expression tree is of type [OSnLNode](#)

6.134.3.44 std::vector<ExprNode*> OSInstance::getNonlinearExpressionTreeModInPostfix (int rowIdx)

Get the postfix tokens for a given row index for the modified Expression Tree (quadratic terms added).

Returns

a vector of pointers to ExprNodes in postfix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.45 std::vector<ExprNode*> OSInstance::getNonlinearExpressionTreeInPrefix (int rowIdx)

Get the prefix tokens for a given row index.

Returns

a vector of pointers to ExprNodes in prefix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.46 `std::string OSInstance::getNonlinearExpressionTreeInInfix (int rowIdx)`

Get the infix representation for a given row (or objective function) index.

Parameters

<i>rowIdx</i>	is the index of the row we want to express in infix.
---------------	--

Returns

a string representation of the tree, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.47 `std::vector<ExprNode*> OSInstance::getNonlinearExpressionTreeModInPrefix (int rowIdx)`

Get the prefix tokens for a given row index for the modified Expression Tree (quadratic terms added).

Returns

a vector of pointers to ExprNodes in prefix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.48 `int OSInstance::getNumberOfNonlinearObjectives ()`**Returns**

the number of [Objectives](#) with a nonlinear term

6.134.3.49 `int OSInstance::getNumberOfNonlinearConstraints ()`**Returns**

the number of [Constraints](#) with a nonlinear term

6.134.3.50 `std::map<int, ScalarExpressionTree* > OSInstance::getAllNonlinearExpressionTrees ()`**Returns**

a map: the key is the row index and the value is the corresponding expression tree

Remarks

If there are several expressions in a single row, this method combines them by adding OSnLPlus nodes

6.134.3.51 `std::map<int, ScalarExpressionTree* > OSInstance::getAllNonlinearExpressionTreesMod ()`**Returns**

a map: the key is the row index and the value is the corresponding expression tree

6.134.3.52 `int* OSInstance::getNonlinearExpressionTreeIndexes ()`

Get all the nonlinear expression tree indexes, i.e., indexes of rows (objectives or constraints) that contain nonlinear expressions.

Returns

a pointer to an integer array of nonlinear expression tree indexes.

6.134.3.53 int OSInstance::getNumberOfNonlinearExpressionTreeIndexes ()

Get the number of unique nonlinear expression tree indexes.

Returns

the number of unique nonlinear expression tree indexes.

6.134.3.54 int* OSInstance::getNonlinearExpressionTreeModIndexes ()

Get all the nonlinear expression tree indexes, i.e., indexes of rows (objectives or constraints) that contain nonlinear expressions after modifying the expression tree to contain quadratic terms.

Returns

a pointer to an integer array of nonlinear expression tree indexes (including quadratic terms).

6.134.3.55 int OSInstance::getNumberOfNonlinearExpressionTreeModIndexes ()

Get the number of unique nonlinear expression tree indexes after modifying the expression tree to contain quadratic terms.

Returns

the number of unique nonlinear expression tree indexes (including quadratic terms).

6.134.3.56 int OSInstance::getMatrixNumber ()

Get the number of matrices.

Returns

the number of matrices.

6.134.3.57 ENUM_MATRIX_TYPE OSInstance::getMatrixType (int *n*)

Get the matrix type.

Returns

the type of elements contained in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Remarks

only the most general element type is returned. (e.g., if matrix contains both constants and general expressions, matrix type is `ENUM_MATRIX_TYPE_general`
for possible types see [OSParameters.h](#)

6.134.3.58 ENUM_MATRIX_SYMMETRY OSInstance::getMatrixSymmetry (int *n*)

Get the matrix symmetry.

Returns

the type of symmetry found in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Remarks

for possible symmetry types see [OSParameters.h](#)

6.134.3.59 int OSInstance::getNumberOfColumnsForMatrix (int *n*)

Get the number of blocks in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the number of blocks. Get the number of columns in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the number of columns.

6.134.3.60 int OSInstance::getNumberOfRowsForMatrix (int *n*)

Get the number of rows in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the number of rows.

6.134.3.61 `std::string OSInstance::getMatrixName (int n)`

Get the number of (nonzero) values in the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the number of values. Get the name of the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the matrix name.

6.134.3.62 `bool OSInstance::matrixHasBase (int n)`

Several tools to parse the constructor list of a matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

6.134.3.63 `bool OSInstance::matrixHasElements (int n)`**6.134.3.64** `bool OSInstance::matrixHasTransformations (int n)`**6.134.3.65** `bool OSInstance::matrixHasBlocks (int n)`**6.134.3.66** `int OSInstance::getNumberOfElementConstructors (int n)`**6.134.3.67** `int OSInstance::getNumberOfTransformationConstructors (int n)`**6.134.3.68** `int OSInstance::getNumberOfBlocksConstructors (int n)`**6.134.3.69** `OSMatrix* OSInstance::getMatrix (int n)`

Get the list of constructors of the matrix.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the matrix constructors.

6.134.3.70 `GeneralSparseMatrix* OSInstance::getMatrixCoefficientsInColumnMajor (int n)`

Get the (nonzero) elements of the matrix in column major form.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the (nonzero) matrix elements.

6.134.3.71 GeneralSparseMatrix* OSInstance::getMatrixCoefficientsInRowMajor (int *n*)

Get the (nonzero) elements of the matrix in row major form.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the (nonzero) matrix elements.

6.134.3.72 GeneralSparseMatrix* OSInstance::getMatrixBlockInColumnMajorForm (int *n*, int *columnIdx*, int *rowIdx*)

Get the (nonzero) elements of the matrix in symmetric block form.

Parameters

<i>n</i>	is the index number associated with the matrix.
----------	---

Returns

the (nonzero) matrix elements. Get a block of the matrix (in symmetric column major form).

Parameters

<i>n</i>	is the index number associated with the matrix.
<i>columnIdx</i>	is the column index of the block's location
<i>rowIdx</i>	is the row index of the block's location

Returns

the (nonzero) matrix elements on column major form.

Remarks

if the block in this location is empty, return NULL.

6.134.3.73 int OSInstance::getNumberOfMatrixVariables ()

Get the number of matrix variables.

Returns

the number of matrix variables

6.134.3.74 `int OSInstance::getNumberOfMatrixObjectives ()`

Get the number of matrix objectives.

Returns

the number of matrix objectives

6.134.3.75 `int OSInstance::getNumberOfMatrixConstraints ()`

Get the number of matrix constraints.

Returns

the number of matrix constraints

6.134.3.76 `int OSInstance::getNumberOfMatrixExpressions ()`

Get the number of matrix-valued expressions.

Returns

the number of matrix-valued variables

6.134.3.77 `MatrixExpression** OSInstance::getMatrixExpressions ()`

Get the pointers to the roots of all matrix expression trees.

Returns

an array of pointers to [MatrixExpression](#) objects

6.134.3.78 `MatrixExpressionTree* OSInstance::getMatrixExpressionTree (int rowIdx)`

Get the matrix expression tree for a given row index.

Returns

a matrix expression tree

6.134.3.79 `std::vector<ExprNode*> OSInstance::getMatrixExpressionTreeInPostfix (int rowIdx)`

Get the postfix tokens for a given row index.

Returns

a vector of pointers to OSnLNodes in postfix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.80 `std::vector<ExprNode*> OSInstance::getMatrixExpressionTreeModInPostfix (int rowIdx)`

Get the postfix tokens for a given row index for the modified Expression Tree (quadratic terms added).

Returns

a vector of pointers to OSnLNodes in postfix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.81 `std::vector<ExprNode*> OSInstance::getMatrixExpressionTreeInPrefix (int rowIdx)`

Get the prefix tokens for a given row index.

Returns

a vector of pointers to OSnLNodes in prefix, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.82 `std::string OSInstance::getMatrixExpressionTreeInInfix (int rowIdx)`

Get the infix representation for a given row (or objective function) index.

Parameters

<i>rowIdx</i>	is the index of the row we want to express in infix.
---------------	--

Returns

a string representation of the tree, if rowIdx does not index a row with a nonlinear term throw an exception

6.134.3.83 `std::map<int, MatrixExpressionTree* > OSInstance::getAllMatrixExpressionTrees ()`

Returns

a map: the key is the row index and the value is the corresponding expression tree

6.134.3.84 `std::map<int, MatrixExpressionTree* > OSInstance::getAllMatrixExpressionTreesMod ()`

Returns

a map: the key is the row index and the value is the corresponding expression tree

6.134.3.85 `int* OSInstance::getMatrixExpressionTreeIndexes ()`

Get all the matrix expression tree indexes, i.e.

indexes of matrix objectives or matrix constraints that contain matrix expressions.

Returns

a pointer to an integer array of matrix expression tree indexes.

6.134.3.86 `int OSInstance::getNumberOfMatrixExpressionTreeIndexes ()`

Get the number of unique matrix expression tree indexes.

Returns

the number of unique matrix expression tree indexes.

6.134.3.87 `std::string OSInstance::getTimeDomainFormat ()`

Get the format of the time domain ("stages"/"interval")

Returns

the format of the time domain.

6.134.3.88 `int OSInstance::getTimeDomainStageNumber ()`

Get the number of stages that make up the time domain.

Returns

the number of time stages.

6.134.3.89 `std::string* OSInstance::getTimeDomainStageNames ()`

Get the names of the stages (NULL or empty string ("") if a stage has not been given a name.

Returns

the names of time stages.

6.134.3.90 `int* OSInstance::getTimeDomainStageNumberOfVariables ()`

Get the number of variables contained in each time stage.

Returns

a vector of size numberOfStages.

6.134.3.91 `int* OSInstance::getTimeDomainStageNumberOfConstraints ()`

Get the number of constraints contained in each time stage.

Returns

a vector of size numberOfStages.

6.134.3.92 `int* OSInstance::getTimeDomainStageNumberOfObjectives ()`

Get the number of objectives contained in each time stage.

Returns

a vector of size numberOfStages.

6.134.3.93 `int** OSInstance::getTimeDomainStageVarList ()`

Get the list of variables in each stage.

Returns

one array of integers for each stage.

6.134.3.94 `int** OSInstance::getTimeDomainStageConList ()`

Get the list of constraints in each stage.

Returns

one array of integers for each stage.

6.134.3.95 `int** OSInstance::getTimeDomainStageObjList ()`

Get the list of objectives in each stage.

Returns

one array of integers for each stage.

6.134.3.96 `double OSInstance::getTimeDomainIntervalStart ()`

Get the start for the time domain interval.

Returns

start end of the time interval.

6.134.3.97 `double OSInstance::getTimeDomainIntervalHorizon ()`

Get the horizon for the time domain interval.

Returns

the end of the time interval.

6.134.3.98 `bool OSInstance::setInstanceName (std::string name)`

set the instance name.

Parameters

<i>name</i>	holds the instance name.
-------------	--------------------------

Returns

whether the instance name was set successfully.

6.134.3.99 `bool OSInstance::setInstanceSource (std::string source)`

set the instance source.

Parameters

<i>source</i>	holds the instance source.
---------------	----------------------------

Returns

whether the instance source was set successfully.

6.134.3.100 `bool OSInstance::setInstanceDescription (std::string description)`

set the instance description.

Parameters

<i>description</i>	holds the instance description.
--------------------	---------------------------------

Returns

whether the instance description was set successfully.

6.134.3.101 `bool OSInstance::setInstanceCreator (std::string fileCreator)`

set the instance creator.

Parameters

<i>fileCreator</i>	holds the instance creator.
--------------------	-----------------------------

Returns

whether the instance creator was set successfully.

6.134.3.102 `bool OSInstance::setInstanceLicence (std::string licence)`

set the instance licence.

Parameters

<i>licence</i>	holds the instance licence.
----------------	-----------------------------

Returns

whether the instance licence was set successfully.

6.134.3.103 `bool OSInstance::setVariableNumber (int number)`

set the number of variables.

Parameters

<i>number</i>	holds the number of variables.
---------------	--------------------------------

Returns

whether the number was set successfully.

6.134.3.104 `bool OSInstance::addVariable (int index, std::string name, double lowerBound, double upperBound, char type)`

add a variable.

In order to use the add method, the setVariableNumber must first be called so that the number of variables is known ahead of time to allocate appropriate memory. If a variable with the given variable index already exists, the old variable will be replaced.

Parameters

<i>index</i>	holds the variable index. It is required.
<i>name</i>	holds the variable name; use null or empty std::string ("") if no variable name.
<i>lowerBound</i>	holds the variable lower bound; use -OSDBL_MAX if no lower bound.
<i>upperBound</i>	holds the variable upper bound; use OSDBL_MAX if no upper bound.
<i>type</i>	holds the variable type character: C for Continuous, B for Binary, I for Integer, S for String, D for semi-continuous, J for semi-integer (i.e., either 0 or integer $\geq n$).

Returns

whether the variable was added successfully.

6.134.3.105 `bool OSInstance::setVariables (int number, std::string * names, double * lowerBounds, double * upperBounds, char * types)`

set all the variable related elements.

All the previous variable-related elements will be deleted.

Parameters

<i>number</i>	holds the number of variables. It is required.
<i>names</i>	holds a std::string array of variable names; use null if no variable names.
<i>lowerBounds</i>	holds a double array of variable lower bounds; use null if all lower bounds are 0; use -OSDBL_MAX if no lower bound for a specific variable in the array.
<i>upperBounds</i>	holds a double array of variable upper bounds; use null if no upper bounds; use OSDBL_MAX if no upper bound for a specific variable in the array.
<i>types</i>	holds a char array of variable types; use null if all variables are continuous; for a specific variable in the array use C for Continuous, B for Binary, I for Integer, S for String, D for semi-continuous, J for semi-integer (i.e., either 0 or integer $\geq n$).
<i>inits</i>	holds a double array of variable initial values; use null if no initial values. – deprecated
<i>initsString</i>	holds a std::string array of variable initial values; use null if no initial std::string values. – deprecated

Returns

whether the variables were set successfully.

6.134.3.106 `bool OSInstance::setObjectiveNumber (int number)`

set the number of objectives.

Parameters

<i>number</i>	holds the number of objectives.
---------------	---------------------------------

Returns

whether the number of objectives was set successfully.

6.134.3.107 `bool OSInstance::addObjective (int index, std::string name, std::string maxOrMin, double constant, double weight, SparseVector * objectiveCoefficients)`

add an objective.

In order to use the add method, the setObjectiveNumber must first be called so that the objective number is known ahead of time to allocate appropriate memory. If a objective with the given objective index already exists, the old objective will be replaced. [Objective](#) index will start from -1, -2, -3, ... down, with -1 corresponding to the first objective.

Parameters

<i>index</i>	holds the objective index. Remember the first objective index is -1, second -2, ...
<i>name</i>	holds the objective name; use null or empty std::string ("") if no objective name.
<i>maxOrMin</i>	holds the objective sense or direction; it can only take two values: "max" or "min".
<i>constant</i>	holds the objective constant; use 0.0 if no objective constant.
<i>weight</i>	holds the objective weight; use 1.0 if no objective weight.
<i>objective-Coefficients</i>	holds the objective coefficients (null if no objective coefficients) in a sparse representation that holds two arrays: index array and a value array.

Returns

whether the objective was added successfully.

6.134.3.108 `bool OSInstance::setObjectives (int number, std::string * names, std::string * maxOrMins, double * constants, double * weights, SparseVector ** objectitiveCoefficients)`

set all the objectives related elements.

All the previous objective-related elements will be deleted.

Parameters

<i>number</i>	holds the number of objectives. It is required.
<i>names</i>	holds a std::string array of objective names; use null if no objective names.
<i>maxOrMins</i>	holds a std::string array of objective objective senses or directions: "max" or "min"; use null if all objectives are "min".
<i>constants</i>	holds a double array of objective constants; use null if all objective constants are 0.0.
<i>weights</i>	holds a double array of objective weights; use null if all objective weights are 1.0.
<i>objective-Coefficients</i>	holds an array of objective coefficients, (null if no objective has any coefficients) For each objective, the coefficients are stored in a sparse representation that holds two arrays: index array and a value array. If for a specific objective, there are no objective coefficients, use null for the corresponding array member.

Returns

whether the objectives were set successfully.

6.134.3.109 `bool OSInstance::setConstraintNumber (int number)`

set the number of constraints.

Parameters

<i>number</i>	holds the number of constraints.
---------------	----------------------------------

Returns

whether the number of constraints was set successfully.

6.134.3.110 **bool** OSInstance::addConstraint (int *index*, std::string *name*, double *lowerBound*, double *upperBound*, double *constant*)

add a constraint.

In order to use the add method, the setConstraintNumber must first be called so that the constraint number is known ahead of time to allocate appropriate memory. If a constraint with the given constraint index already exists, the old constraint will be replaced.

Parameters

<i>index</i>	holds the constraint index. It is required.
<i>name</i>	holds the constraint name; use null or empty std::string ("") if no constraint name.
<i>lowerBound</i>	holds the constraint lower bound; use -OSDBL_MAX if no lower bound.
<i>upperBound</i>	holds the constraint upper bound; use OSDBL_MAX if no upper bound.

Returns

whether the constraint was added successfully.

6.134.3.111 **bool** OSInstance::setConstraints (int *number*, std::string * *names*, double * *lowerBounds*, double * *upperBounds*, double * *constants*)

set all the constraint related elements.

All the previous constraint-related elements will be deleted.

Parameters

<i>number</i>	holds the number of constraints. It is required.
<i>names</i>	holds a std::string array of constraint names; use null if no constraint names.
<i>lowerBounds</i>	holds a double array of constraint lower bounds; use null if no lower bounds; use -OSDBL_MAX if no lower bound for a specific constraint in the array.
<i>upperBounds</i>	holds a double array of constraint upper bounds; use null if no upper bounds; use OSDBL_MAX if no upper bound for a specific constraint in the array.

Returns

whether the constraints were set successfully.

6.134.3.112 **bool** OSInstance::setLinearConstraintCoefficients (int *numberOfValues*, bool *isColumnMajor*, double * *values*, int *valuesBegin*, int *valuesEnd*, int * *indexes*, int *indexesBegin*, int *indexesEnd*, int * *starts*, int *startsBegin*, int *startsEnd*)

set linear constraint coefficients

Parameters

<i>numberOfValues</i>	holds the number of specified coefficient values (usually nonzero) in the coefficient matrix.
<i>isColumnMajor</i>	holds whether the coefficient matrix is stored in column major (true) or row major (false).
<i>values</i>	holds a double array coefficient values in the matrix.
<i>valuesBegin</i>	holds the begin index of the values array to copy from (usually 0).
<i>valuesEnd</i>	holds the end index of the values array to copy till (usually values.length - 1).
<i>indexes</i>	holds an integer array column/row indexes for each value in the values array.
<i>indexesBegin</i>	holds the begin index of the indexes array to copy from (usually 0).

<i>indexesEnd</i>	holds the end index of the indexes array to copy till (usually <code>indexes.length - 1</code>).
<i>starts</i>	holds an integer array start indexes in the matrix; the first value of starts should always be 0.
<i>startsBegin</i>	holds the begin index of the starts array to copy from (usually 0).
<i>startsEnd</i>	holds the end index of the starts array to copy till (usually <code>starts.length - 1</code>).

Returns

whether the linear constraint coefficients were set successfully.

6.134.3.113 `bool OSInstance::copyLinearConstraintCoefficients (int numberOfValues, bool isColumnMajor, double * values, int valuesBegin, int valuesEnd, int * indexes, int indexesBegin, int indexesEnd, int * starts, int startsBegin, int startsEnd)`

copy linear constraint coefficients: perform a deep copy of the sparse matrix

Parameters

<i>numberOfValues</i>	holds the number of specified coefficient values (usually nonzero) in the coefficient matrix.
<i>isColumnMajor</i>	holds whether the coefficient matrix is stored in column major (true) or row major (false).
<i>values</i>	holds a double array coefficient values in the matrix.
<i>valuesBegin</i>	holds the begin index of the values array to copy from (usually 0).
<i>valuesEnd</i>	holds the end index of the values array to copy till (usually <code>values.length - 1</code>).
<i>indexes</i>	holds an integer array column/row indexes for each value in the values array.
<i>indexesBegin</i>	holds the begin index of the indexes array to copy from (usually 0).
<i>indexesEnd</i>	holds the end index of the indexes array to copy till (usually <code>indexes.length - 1</code>).
<i>starts</i>	holds an integer array start indexes in the matrix; the first value of starts should always be 0.
<i>startsBegin</i>	holds the begin index of the starts array to copy from (usually 0).
<i>startsEnd</i>	holds the end index of the starts array to copy till (usually <code>starts.length - 1</code>).

Returns

whether the linear constraint coefficients were copied successfully.

6.134.3.114 `bool OSInstance::setNumberOfQuadraticTerms (int nq)`

set the number of quadratic terms

Parameters

<i>nq</i>	holds the number of quadratic terms.
-----------	--------------------------------------

Returns

whether the number of quadratic terms was set successfully.

6.134.3.115 `bool OSInstance::setQuadraticCoefficients (int number, int * rowIndexes, int * varOneIndexes, int * varTwoIndexes, double * coefficients, int begin, int end)`

set quadratic coefficients into the QuadraticCoefficients->qTerm data structure

Parameters

<i>number</i>	holds the number of quadratic terms.
<i>rowIndexes</i>	holds an integer array of row indexes of all the quadratic terms. A negative integer corresponds to an objective row, e.g. -1 for 1st objective and -2 for 2nd.
<i>varOneIndexes</i>	holds an integer array of the first variable indexes of all the quadratic terms.
<i>varTwoIndexes</i>	holds an integer array of the second variable indexes of all the quadratic terms.
<i>coefficients</i>	holds an array of double containing all the quadratic term coefficients.
<i>begin</i>	holds the begin index of all the arrays to copy from (usually = 0).
<i>end</i>	holds the end index of all the arrays to copy till (usually = array length -1).

Returns

whether the quadratic terms were set successfully.

6.134.3.116 `bool OSInstance::setQuadraticTermsInNonlinearExpressions (int number, int * rowIndexes, int * varOneIndexes, int * varTwoIndexes, double * coefficients)`

set quadratic terms in nonlinearExpressions

Parameters

<i>number</i>	holds the number of quadratic terms.
<i>rowIndexes</i>	holds an integer array of row indexes of all the quadratic terms. A negative integer corresponds to an objective row, e.g. -1 for 1st objective and -2 for 2nd.
<i>varOneIndexes</i>	holds an integer array of the first variable indexes of all the quadratic terms.
<i>varTwoIndexes</i>	holds an integer array of the second variable indexes of all the quadratic terms.
<i>coefficients</i>	holds a double array all the quadratic term coefficients.

Returns

whether the quadratic terms were set successfully.

6.134.3.117 `bool OSInstance::setNonlinearExpressions (int nexpr, NI ** root)`

set nonlinear expressions

Parameters

<i>nexpr</i>	holds the number of nonlinear expressions.
<i>root</i>	holds a pointer array to the root nodes of all the nonlinear expressions.

Returns

whether the nonlinear expressions were set successfully.

6.134.3.118 `bool OSInstance::setMatrixNumber (int number)`

set the number of matrices

Parameters

<i>number</i>	holds the number of matrices
---------------	------------------------------

Returns

whether the number of matrices was set successfully.

6.134.3.119 `bool OSInstance::addMatrix (int index, std::string name, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry, ENUM_MATRIX_TYPE matrixType, unsigned int inumberOfChildren, MatrixNode ** m_mChildren)`

add a matrix.

In order to use the add method, the setMatrixNumber must first be called so that the number of matrices is known ahead of time to allocate appropriate memory. If a matrix with the given matrix index already exists, the old matrix will be replaced.

Parameters

<i>index</i>	holds the matrix index. It is required.
<i>name</i>	holds the matrix name; use null or empty std::string ("") if no matrix name.
<i>numberOfRows</i>	holds the number of rows. It is required. Use 1 for column vectors.
<i>numberOfColumns</i>	holds the number of columns. It is required. Use 1 for row vectors.
<i>symmetry</i>	holds the type of symmetry used in the definition of the matrix. For more information see the enumeration ENUM_MATRIX_SYMMETRY in OSGeneral.h . If no symmetry, use ENUM_MATRIX_SYMMETRY_none.
<i>matrixType</i>	tracks the type of elements contained in this matrix. For more information see the enumeration ENUM_MATRIX_TYPE in OSGeneral.h . If unsure, use ENUM_MATRIX_TYPE_unknown.
<i>inumberOfChildren</i>	is the number of MatrixNode child elements, i.e., the number of matrix constructors in the m_mChildren array.
<i>m_mChildren</i>	is the array of matrix constructors used in the definition of this matrix.

Returns

whether the matrix was added successfully.

6.134.3.120 `bool OSInstance::setConeNumber (int number)`

set the number of cones

Parameters

<i>number</i>	holds the number of cones
---------------	---------------------------

Returns

whether the number of cones was set successfully.

6.134.3.121 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for cones that require basic information only.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be one of ENUM_CONE_TYPE_nonnegative, ENUM_CONE_TYPE_nonpositive, ENUM_CONE_TYPE_copositiveMatrices, ENUM_CONE_TYPE_completelyPositiveMatrices.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

```
6.134.3.122 bool OSInstance::addCone ( int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType,
std::string name, int numberOfComponents, int * components, int numberOfOtherIndexes = 0, int * otherIndexes =
NULL )
```

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for product and intersection cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be one of ENUM_CONE_TYPE_product, ENUM_CONE_TYPE_intersection.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>numberOfComponents</i>	holds the number of components of this cone.
<i>components</i>	holds the indexes of the components of this cone.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.123 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int referenceIdx, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for positive or negative cones that reference another cone or matrix.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be one of ENUM_CONE_TYPE_dual, ENUM_CONE_TYPE_polar, ENUM_CONE_TYPE_polyhedral.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>referenceIdx</i>	holds the index of a cone or matrix used in the definition of this cone.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.124 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, std::string semidefiniteness, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for positive or negative semidefinite cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.

<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be ENUM_CONE_TYPE_semidefinite.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>semidefiniteness</i>	distinguishes positive and negative semidefinite cones. It must be either "positive" or "negative".
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.125 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int distortionMatrixIdx, double normFactor, int axisDirection, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for quadratic cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be ENUM_CONE_TYPE_quadratic.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>distortionMatrixIdx</i>	holds the index of a distortion matrix. Use -1 if there is none.
<i>normFactor</i>	holds a scale factor for the norm. Use 1 if there is none.
<i>axisDirection</i>	holds the index of the axis direction. The most usual value is 0.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.126 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int distortionMatrixIdx, double normFactor, int firstAxisDirection, int secondAxisDirection, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for rotated quadratic cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be ENUM_CONE_TYPE_rotatedQuadratic.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>distortionMatrixIdx</i>	holds the index of a distortion matrix. Use -1 if there is none.
<i>normFactor</i>	holds a scale factor for the norm. Use 1 if there is none.
<i>firstAxisDirection</i>	holds the index of the first axis direction. The most usual value is 0.
<i>secondAxisDirection</i>	holds the index of the second axis direction. The most usual value is 1.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.127 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int distortionMatrixIdx, double normFactor, int axisDirection, double pNorm, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for normed cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be ENUM_CONE_TYPE_normed.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>distortionMatrixIdx</i>	holds the index of a distortion matrix. Use -1 if there is none.
<i>normFactor</i>	holds a scale factor for the norm. Use 1 if there is none.
<i>pNorm</i>	holds the norm descriptor. It must be greater than or equal to 1.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.128 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int maxDegree, int numberOfUB, double * ub, int numberOfLB, double * lb, int numberOfOtherIndexes = 0, int * otherIndexes = NULL)`

add a cone.

In order to use the add method, the setConeNumber must first be called so that the number of cones is known ahead of time to allocate appropriate memory. If a cone with the given cone index already exists, the old cone will be replaced.

Remarks

This method has different signatures to cater for different types of cones. This signature is used for cones of nonnegative polynomials and similar cones.

Parameters

<i>index</i>	holds the cone index. It is required.
<i>numberOfRows</i>	holds the number of rows. It is required.
<i>numberOfColumns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYPE further up in this file. This argument is required and must be ENUM_CONE_TYPE_nonnegative-Polynomials. ENUM_CONE_TYPE_sumOfSquaresPolynomials. ENUM_CONE_TYPE_moment.
<i>name</i>	holds the cone name; use null or empty std::string ("") if no cone name.
<i>maxDegree</i>	holds the maximum degree of the polynomials. Use 1, 2, 3, ..., INF.
<i>numberOfUB</i>	holds the number of (box-type) upper bound constraints. Use 0 if there are none.
<i>ub</i>	holds the upper bound values. Use null if there are no upper bounds.
<i>numberOfLB</i>	holds the number of (box-type) lower bound constraints. Use 0 if there are none.
<i>lb</i>	holds the lower bound values. Use null if there are no lower bounds.
<i>numberOfOtherIndexes</i>	holds the number of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to 0.
<i>otherIndexes</i>	holds the array of other indexes if the cone contains higher-dimensional tensors. This argument is optional and can be omitted. It defaults to null.

Returns

whether the cone was added successfully.

6.134.3.129 std::string OSInstance::printModel ()

Print the infix representation of the problem.

Returns

a string with the infix representation

6.134.3.130 std::string OSInstance::printModel (int rowIdx)

Print the infix representation of the row (which could be an an objective function row) indexed by rowIdx.

Parameters

<i>rowIdx</i>	is the index of the row we want to express in infix.
---------------	--

Returns

a string with the infix representation

6.134.3.131 bool OSInstance::initializeNonLinearStructures ()

Initialize the data structures for the nonlinear API.

Returns

true if we have initialized the nonlinear data strucutres.

6.134.3.132 double OSInstance::calculateFunctionValue (int idx, double * x, bool new_x)

Calculate the function value for function (constraint or objective) indexed by idx.

Parameters

<i>idx</i>	is the index on the constraint (0, 1, 2, 3, ...) or objective function (-1, -2, -3, ...).
<i>x</i>	is a pointer (double array) to the current variable values
<i>new_x</i>	is false if any evaluation method was previously called for the current x has been evaluated for the current iterate x use a value of false if not sure

Returns

the function value as a double.

6.134.3.133 double* OSInstance::calculateAllConstraintFunctionValues (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)

Calculate all of the constraint function values.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called for the current x for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a double array of constraint function values – the size of the array is equal to [getConstraintNumber\(\)](#).

6.134.3.134 `double* OSInstance::calculateAllConstraintFunctionValues (double * x, bool new_x)`

Calculate all of the constraint function values, we are overloading this function and this version of the method will not use any AD and will evaluate function values from the OS Expression Tree.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate

Returns

a double array of constraint function values – the size of the array is equal to [getConstraintNumber\(\)](#).

6.134.3.135 `double* OSInstance::calculateAllObjectiveFunctionValues (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)`

Calculate all of the objective function values.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a double array of objective function values – the size of the array is equal to [getObjectiveNumber\(\)](#).

6.134.3.136 `double* OSInstance::calculateAllObjectiveFunctionValues (double * x, bool new_x)`

Calculate all of the objective function values, we are overloading this function and this version of the method will not use any AD and will evaluate function values from the OS Expression Tree.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate

Returns

a double array of objective function values – the size of the array is equal to [getObjectiveNumber\(\)](#).

6.134.3.137 `SparseJacobianMatrix*` `OSInstance::calculateAllConstraintFunctionGradients (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)`

Calculate the gradient of all constraint functions.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a pointer a [SparseJacobianMatrix](#).

6.134.3.138 `SparseVector*` `OSInstance::calculateConstraintFunctionGradient (double * x, double * objLambda, double * conLambda, int idx, bool new_x, int highestOrder)`

Calculate the gradient of the constraint function indexed by *idx*.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints <i>idx</i> is the index of the constraint function gradient
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a pointer to a sparse vector of doubles.

6.134.3.139 `SparseVector*` `OSInstance::calculateConstraintFunctionGradient (double * x, int idx, bool new_x)`

Calculate the gradient of the constraint function indexed by *idx* this function is overloaded.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values <i>idx</i> is the index of the constraint function gradient
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a pointer to a sparse vector of doubles.

6.134.3.140 `double** OSInstance::calculateAllObjectiveFunctionGradients (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)`

Calculate the gradient of all objective functions.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

an array of pointer to dense objective function gradients.

6.134.3.141 `double* OSInstance::calculateObjectiveFunctionGradient (double * x, double * objLambda, double * conLambda, int objIdx, bool new_x, int highestOrder)`

Calculate the gradient of the objective function indexed by *objIdx*.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints <i>objIdx</i> is the index of the objective function being optimized
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a pointer to a dense vector of doubles.

6.134.3.142 `double* OSInstance::calculateObjectiveFunctionGradient (double * x, int objIdx, bool new_x)`

Calculate the gradient of the objective function indexed by *objIdx* this function is overloaded.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objIdx</i>	is the index of the objective function being optimized
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate

Returns

a pointer to a dense vector of doubles.

6.134.3.143 `SparseHessianMatrix* OSInstance::calculateLagrangianHessian (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)`

Calculate the Hessian of the Lagrangian Expression Tree This method will build the CppAD expression tree for only the first iteration Use this method on if the value of *x* does not affect the operations sequence.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>objLambda</i>	is the Lagrange multiplier on the objective function
<i>conLambda</i>	is pointer (double array) of Lagrange multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate
<i>highestOrder</i>	is the highest order of the derivative being calculated

Returns

a pointer a [SparseHessianMatrix](#). Each array member corresponds to one constraint gradient.

6.134.3.144 `SparseHessianMatrix* OSInstance::calculateHessian (double * x, int idx, bool new_x)`

Calculate the Hessian of a constraint or objective function.

Parameters

<i>x</i>	is a pointer (double array) to the current variable values
<i>new_x</i>	is false if any evaluation method was previously called for the current iterate idx is the index of the either a constraint or objective function Hessian

Returns

a pointer a [SparseVector](#). Each array member corresponds to one constraint gradient.

6.134.3.145 `bool OSInstance::getSparseJacobianFromColumnMajor ()`

Returns

true if successful in generating the constraints gradient.

6.134.3.146 `bool OSInstance::getSparseJacobianFromRowMajor ()`

Returns

true if successful in generating the constraints gradient.

6.134.3.147 `ScalarExpressionTree* OSInstance::getLagrangianExpTree ()`

Returns

a pointer to the ExpressionTree for the Lagrangian function of current instance we only take the Lagrangian of the rows with nonlinear terms

6.134.3.148 `std::map<int, int> OSInstance::getAllNonlinearVariablesIndexMap ()`

Returns

a pointer to a map of the indices of all of the variables that appear in the Lagrangian function

6.134.3.149 `SparseHessianMatrix* OSInstance::getLagrangianHessianSparsityPattern ()`

Returns

a pointer to a [SparseHessianMatrix](#) with the nonzero structure of the Lagrangian Expression Tree

6.134.3.150 `bool OSInstance::addQTermsToExpressionTree ()`**Returns**

true if successful in adding the qTerms to the ExpressionTree.

Remarks

due to the typo in the name of the method, this has been flagged as obsolescent and is being replaced by [addQ-TermsToExpressionTree\(\)](#) – see below

6.134.3.151 `bool OSInstance::addQTermsToExpressionTree ()`

This method adds quadratic terms into the array of expression trees.

There is at most one expression tree per row (see `getAllNonlinearExpressionTrees`)

Returns

true if successful in adding the qTerms to the ExpressionTree.

6.134.3.152 `SparseJacobianMatrix* OSInstance::getJacobianSparsityPattern ()`**Returns**

pointer to a [SparseJacobianMatrix](#).

6.134.3.153 `void OSInstance::duplicateExpressionTreesMap ()`

duplicate the map of expression trees.

6.134.3.154 `bool OSInstance::createOSADFun (std::vector< double > vdX)`

Create the a CppAD Function object: this is a function where the domain is the set of variables for the problem and the range is the objective function plus constraints.

Parameters

<code>vdX</code>	is a vector of doubles holding the current primal variable values the size of x should equal <code>instance-Data->variables->numberOfVariables</code>
------------------	---

Returns

if successfully created

6.134.3.155 `std::vector<double> OSInstance::forwardAD (int p, std::vector< double > vdX)`

Perform an AD forward sweep.

Parameters

<code>p</code>	is the highest order Taylor coefficient
<code>vdX</code>	is a vector of doubles of the current primal variable values the size of <code>vdX</code> <code>m_iNumberOfNonlinear-Variables</code>

Returns

a double vector equal to the dimension of the range space the result of the forward p sweep

6.134.3.156 `std::vector<double> OSInstance::reverseAD (int p, std::vector< double > vdlambda)`

Perform an AD reverse sweep.

Parameters

<i>p</i>	is the order of the sweep
<i>vdlambda</i>	is a vector of doubles of the current dual (lagrange) variable values the size of lambda should equal number of objective functions plus number of constraints

Returns

a double vector equal to the n*p

6.134.3.157 `int OSInstance::getADSparsityHessian ()`

end revised AD code

Call the AD routine to fill in m_vbLagHessNonz and determine the nonzeros.

Returns

the number of nonzeros in the Hessian

6.134.3.158 `bool OSInstance::getIterateResults (double * x, double * objLambda, double * conLambda, bool new_x, int highestOrder)`

end revised AD code

Get the information for each iteration. Get the functions values, Jacobian and Hessian of the Lagrangian

Parameters

<i>x</i>	is a pointer of doubles of primal values for the current iteration
<i>objLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the objective functions
<i>conLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the constraints
<i>new_x</i>	is false if any evaluation method was previously called
<i>highestOrder</i>	is the highest order derivative to be calculated

Returns

true if successful

6.134.3.159 `bool OSInstance::getZeroOrderResults (double * x, double * objLambda, double * conLambda)`

Calculate function values.

Parameters

<i>x</i>	is a pointer of doubles of primal values for the current iteration
<i>objLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the objective functions
<i>conLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the constraints

Returns

true if successful

6.134.3.160 `bool OSInstance::getFirstOrderResults (double * x, double * objLambda, double * conLambda)`

Calculate first derivatives.

Parameters

<i>x</i>	is a pointer of doubles of primal values for the current iteration
<i>objLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the objective functions
<i>conLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the constraints

Returns

true if successful

6.134.3.161 `bool OSInstance::getSecondOrderResults (double * x, double * objLambda, double * conLambda)`

Calculate second derivatives.

Parameters

<i>x</i>	is a pointer of doubles of primal values for the current iteration
<i>objLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the objective functions
<i>conLambda</i>	is a pointer of doubles of the current dual (Lagrange) multipliers on the constraints

Returns

true if successful

6.134.3.162 `bool OSInstance::initForAlgDiff ()`

This should be called by nonlinear solvers using callback functions.

initForAlgDiff will initialize the correct nonlinear structures in preparation for using the algorithmic differentiation routines.

Returns

true if successful

6.134.3.163 `bool OSInstance::initObjGradients ()`

This should be called by [initForAlgDiff\(\)](#)

initObjGradients will initialize the objective function gradients to be equal to the coefficients given in the <coef> section of the OSiL instance

Returns

true if successful

6.134.3.164 `bool OSInstance::setTimeDomain (std::string format)`

This sets the format of the time domain ("stages"/"interval"/"none")

6.134.3.165 `bool OSInstance::setTimeDomainStages (int number, std::string * names)`

This sets the number (and optionally names) of the time stages.

6.134.3.166 `bool OSInstance::setTimeDomainStageVariablesOrdered (int numberOfStages, int * numberOfVariables, int * startIdx)`

This sets the variables associated with each time domain stage in temporal order.

(I.e., for each stage *numberOfVariables* gives the number of variables associated with this stage and *startIdx* gives the first variable in this stage.)

6.134.3.167 `bool OSInstance::setTimeDomainStageVariablesUnordered (int numberOfStages, int * numberOfVariables, int ** varIndex)`

This sets the variables associated with each time domain stage in arbitrary order.

(I.e., for each stage *numberOfVariables* gives the number of variables associated with this stage and *varIndex[i]* gives the index of each variable in stage[i].)

6.134.3.168 `bool OSInstance::setTimeDomainStageConstraintsOrdered (int numberOfStages, int * numberOfConstraints, int * startIdx)`

This sets the constraints associated with each time domain stage in temporal order.

(I.e., for each stage *numberOfConstraints* gives the number of constraints associated with this stage and *startIdx* gives the first constraint in this stage.)

6.134.3.169 `bool OSInstance::setTimeDomainStageConstraintsUnordered (int numberOfStages, int * numberOfConstraints, int ** conIndex)`

This sets the constraints associated with each time domain stage in arbitrary order.

(I.e., for each stage *numberOfConstraints* gives the number of constraints associated with this stage and *conIndex[i]* gives the index of each constraint in stage[i].)

6.134.3.170 `bool OSInstance::setTimeDomainStageObjectivesOrdered (int numberOfStages, int * numberOfObjectives, int * startIdx)`

This sets the objectives associated with each time domain stage in temporal order.

(I.e., for each stage *numberOfObjectives* gives the number of objectives associated with this stage and *startIdx* gives the first objective in this stage.)

6.134.3.171 `bool OSInstance::setTimeDomainStageObjectivesUnordered (int numberOfStages, int * numberOfObjectives, int ** objIndex)`

This sets the objectives associated with each time domain stage in arbitrary order.

(I.e., for each stage *numberOfObjectives* gives the number of objectives associated with this stage and *objIndex[i]* gives the index of each objective in stage[i].)

6.134.3.172 `bool OSInstance::setTimeDomainInterval (double start, double horizon)`

This sets the start and end of the time interval.

6.134.4 Member Data Documentation

6.134.4.1 GeneralFileHeader* OSInstance::instanceHeader

the instanceHeader is implemented as a general file header object to allow sharing of classes between schemas
Definition at line 2277 of file OSInstance.h.

6.134.4.2 InstanceData* OSInstance::instanceData

A pointer to an [InstanceData](#) object.
Definition at line 2280 of file OSInstance.h.

6.134.4.3 bool OSInstance::bVariablesModified

bVariablesModified is true if the variables data has been modified.
Definition at line 2290 of file OSInstance.h.

6.134.4.4 bool OSInstance::bObjectivesModified

bObjectivesModified is true if the objective function data has been modified.
Definition at line 2295 of file OSInstance.h.

6.134.4.5 bool OSInstance::bConstraintsModified

bConstraintsModified is true if the constraints data has been modified.
Definition at line 2300 of file OSInstance.h.

6.134.4.6 bool OSInstance::bAMatrixModified

bAMatrixModified is true if the A matrix data has been modified.
Definition at line 2305 of file OSInstance.h.

6.134.4.7 bool OSInstance::bUseExpTreeForFunEval

bUseExpTreeForFunEval is set to true if you wish to use the OS Expression Tree for function evaluations instead of AD – false by default.
Definition at line 4906 of file OSInstance.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.135 OSMatlab Class Reference

The [OSMatlab](#) Class.

```
#include <OSMatlabSolver.h>
```

- `OSMatlab ()`
the *OSMatlab* class constructor
- `~OSMatlab ()`
the *OSMatlab* class destructor

- `std::string solve ()`
Solve the problem instance.
- `void createOSInstance ()`
Create an [OSInstance](#).

Public Attributes

- `SparseMatrix * sparseMat`
sparseMat is a pointer to an OS Sprase Matrix data structure
- `double * bl`
bl is a pointer to the lower bounds on the constraints
- `double * bu`
bu is a pointer to the upper bounds on the constraints
- `double * obj`
obj is a pointer to the objective function coefficients
- `double * vl`
vl is a pointer to the lower bounds on the varialbes
- `double * vu`
vu is a pointer to the upper bounds on the variables
- `int numVar`
numVar is the number of variables in the problem
- `int numCon`
numCon is the number of constraints in the problem
- `char * varType`
varType is a pointer to the variable type eg C, B, I
- `bool objType`
objType indicates whether or not we have a max (1) or a min (0)
- `int numQTerms`
numQTerms is the number of quadratic terms
- `int * qRows`
qRows is a pointer to the row index of each quadratic term
- `int * qIndex1`
qIndex1 is a pointer to the index of the first variable in each of the quadratic terms
- `int * qIndex2`
qIndex2 is a pointer to the index of the second variable in each of the quadratic terms
- `double * qVal`
qVal is a pointer to the coefficient value of each of the quadratic terms.
- `DefaultSolver * solverType`
solverType is the a pointer to the sovler that will be requested
- `std::string instanceName`
instanceName is the name of the problem instance
- `std::string sSolverName`
sSolverName is the name of the solver
- `std::string sAgentAddress`
is the address of the solver service
- `OSInstance * osinstance`
osinstance is a pointer to an [OSInstance](#) object that gets created from the MATLAB data structures
- `std::string osil`
is the osil instance that gets created from the MATLAB data structures

6.135.1 Detailed Description

The [OSMatlab](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

the [OSMatlab](#) class is used by the matlabSolver mex (Matlab EXecutable) which takes MATLAB data structures and creates an OSiL string.

for now we can only handle linear integer and quadratic programming problems, not general nonlinear problems

Definition at line 49 of file OSMatlabSolver.h.

6.135.2 Constructor & Destructor Documentation

6.135.2.1 OSMatlab::OSMatlab ()

the [OSMatlab](#) class constructor

6.135.2.2 OSMatlab::~OSMatlab ()

the [OSMatlab](#) class destructor

6.135.3 Member Function Documentation

6.135.3.1 std::string OSMatlab::solve ()

Solve the problem instance.

Returns

a string with the solution in OSrL format

6.135.3.2 void OSMatlab::createOSInstance ()

Create an [OSInstance](#).

6.135.4 Member Data Documentation

6.135.4.1 SparseMatrix* OSMatlab::sparseMat

sparseMat is a pointer to an OS Sprase Matrix data structure

Definition at line 63 of file OSMatlabSolver.h.

6.135.4.2 double* OSMatlab::bl

bl is a pointer to the lower bounds on the constraints

Definition at line 68 of file OSMatlabSolver.h.

6.135.4.3 double* OSMatlab::bu

bu is a pointer to the upper bounds on the constraints

Definition at line 73 of file OSMatlabSolver.h.

6.135.4.4 double* OSMatlab::obj

obj is a pointer to the objective function coefficients

Definition at line 78 of file OSMatlabSolver.h.

6.135.4.5 double* OSMatlab::vl

vl is a pointer to the lower bounds on the variables

Definition at line 83 of file OSMatlabSolver.h.

6.135.4.6 double* OSMatlab::vu

vu is a pointer to the upper bounds on the variables

Definition at line 88 of file OSMatlabSolver.h.

6.135.4.7 int OSMatlab::numVar

numVar is the number of variables in the problem

Definition at line 93 of file OSMatlabSolver.h.

6.135.4.8 int OSMatlab::numCon

numCon is the number of constraints in the problem

Definition at line 98 of file OSMatlabSolver.h.

6.135.4.9 char* OSMatlab::varType

varType is a pointer to the variable type eg C, B, I

Definition at line 103 of file OSMatlabSolver.h.

6.135.4.10 bool OSMatlab::objType

objType indicates whether or not we have a max (1) or a min (0)

Definition at line 108 of file OSMatlabSolver.h.

6.135.4.11 int OSMatlab::numQTerms

numQTerms is the number of quadratic terms

Definition at line 111 of file OSMatlabSolver.h.

6.135.4.12 int* OSMatlab::qRows

qRows is a pointer to the row index of each quadratic term

Definition at line 116 of file OSMatlabSolver.h.

6.135.4.13 int* OSMatlab::qIndex1

qIndex1 is a pointer to the index of the first variable in each of the quadratic terms

Definition at line 121 of file OSMatlabSolver.h.

6.135.4.14 int* OSMatlab::qIndex2

qIndex2 is a pointer to the index of the second variable in each of the quadratic terms

Definition at line 126 of file OSMatlabSolver.h.

6.135.4.15 double* OSMatlab::qVal

qVal is a pointer to the coefficient value of each of the quadratic terms.

Definition at line 131 of file OSMatlabSolver.h.

6.135.4.16 DefaultSolver* OSMatlab::solverType

solverType is the a pointer to the sovler that will be requested

Definition at line 136 of file OSMatlabSolver.h.

6.135.4.17 std::string OSMatlab::instanceName

instanceName is the name of the problem instance

Definition at line 139 of file OSMatlabSolver.h.

6.135.4.18 std::string OSMatlab::sSolverName

sSolverName is the name of the solver

Definition at line 142 of file OSMatlabSolver.h.

6.135.4.19 std::string OSMatlab::sAgentAddress

is the address of the solver service

Definition at line 145 of file OSMatlabSolver.h.

6.135.4.20 OSInstance* OSMatlab::osinstance

osinstance is a pointer to an [OSInstance](#) object that gets created from the MATLAB data structures

Definition at line 163 of file OSMatlabSolver.h.

6.135.4.21 std::string OSMatlab::osil

is the osil instance that gets created from the MATLAB data structures

Definition at line 168 of file OSMatlabSolver.h.

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

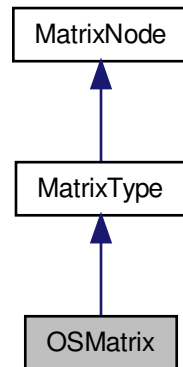
- [/home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSMatlabSolver.h](#)

6.136 OSMatrix Class Reference

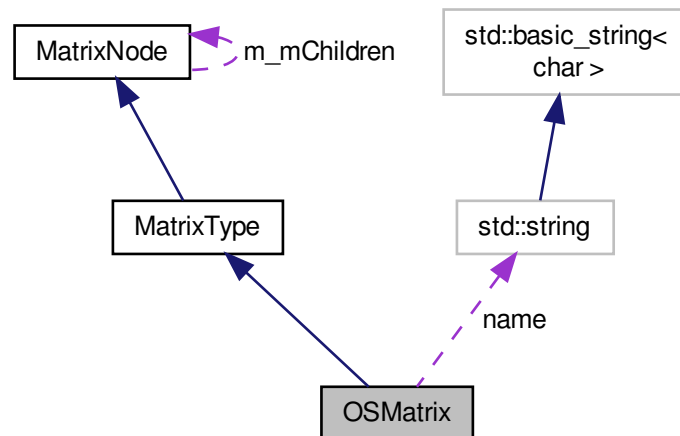
a data structure to represent a matrix object (derived from [MatrixType](#))

```
#include <OSMatrix.h>
```

Inheritance diagram for OSMatrix:



Collaboration diagram for OSMatrix:



Public Member Functions

- [OSMatrix](#) ()
- [~OSMatrix](#) ()
- [OSMatrix * createConstructorTreeFromPrefix](#) (std::vector< [MatrixNode](#) * > mtxConstructorVec)
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) getNodeType ()
- virtual std::string getNodeName ()
- virtual [ENUM_MATRIX_TYPE](#) getMatrixType ()
- int [getRowPartitionSize](#) ()
get the size of the row partition of a matrix
- int * [getRowPartition](#) ()
get the row partition of the matrix
- int [getColumnPartitionSize](#) ()
get the size of the column partition of a matrix
- int * [getColumnPartition](#) ()
get the column partition of the matrix
- bool [processBlocks](#) ()
process the dimensions of blocks found in the constructors of the matrix (Note that there could be several block structures, potentially conflicting)
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- bool [isBlockDiagonal](#) ()
A method to check whether a matrix is block-diagonal.
- bool [setMatrix](#) (std::string name, int numberOfRows, int numberOfColumns, [ENUM_MATRIX_SYMMETRY](#) symmetry, [ENUM_MATRIX_TYPE](#) matrixType, unsigned int innumberOfChildren, [MatrixNode](#) **m_mChildren)
add values to this matrix.
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual [OSMatrix * cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [isEqual](#) ([OSMatrix](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([OSMatrix](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
- std::string [name](#)

6.136.1 Detailed Description

a data structure to represent a matrix object (derived from [MatrixType](#))

Definition at line 1990 of file OSMatrix.h.

6.136.2 Constructor & Destructor Documentation

6.136.2.1 OSMatrix::OSMatrix ()

6.136.2.2 OSMatrix::~~OSMatrix ()

6.136.3 Member Function Documentation

6.136.3.1 OSMatrix* OSMatrix::createConstructorTreeFromPrefix (std::vector< MatrixNode * > mtxConstructorVec)

Take a vector of MatrixNodes in prefix format and create a matrix root node

Parameters

<i>mtxConstructorVec</i>	holds a vector of pointers to matrix constructors, mtxConstructorVec and blocks in prefix format
--------------------------	--

Returns

a pointer to an [OSMatrix](#) which is the root of a list of constructors.

6.136.3.2 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE OSMatrix::getNodeType () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.136.3.3 virtual std::string OSMatrix::getNodeName () [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

6.136.3.4 virtual ENUM_MATRIX_TYPE OSMatrix::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.136.3.5 int OSMatrix::getRowPartitionSize ()

get the size of the row partition of a matrix

Returns

an corresponding to the number of partition points of the rows of this matrix (which is one more than the number of blocks in one row)

6.136.3.6 int* OSMatrix::getRowPartition ()

get the row partition of the matrix

Returns

a vector of int corresponding to the partition points of the rows of this matrix

6.136.3.7 int OSMatrix::getColumnPartitionSize ()

get the size of the column partition of a matrix

Returns

an corresponding to the number of partition points of the columns of this matrix (which is one more than the number of blocks in one column)

6.136.3.8 int* OSMatrix::getColumnPartition ()

get the column partition of the matrix

Returns

a vector of int corresponding to the partition points of the columns of this matrix

6.136.3.9 bool OSMatrix::processBlocks ()

process the dimensions of blocks found in the constructors of the matrix (Note that there could be several block structures, potentially conflicting)

Remarks

This method is called by the previous four methods

6.136.3.10 virtual bool OSMatrix::alignsOnBlockBoundary (int *firstRow*, int *firstColumn*, int *nRows*, int *nCols*) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block

Reimplemented from [MatrixType](#).

6.136.3.11 bool OSMatrix::isBlockDiagonal ()

A method to check whether a matrix is block-diagonal.

6.136.3.12 bool OSMatrix::setMatrix (std::string name, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry, ENUM_MATRIX_TYPE matrixType, unsigned int inumberOfChildren, MatrixNode ** m_mChildren)

add values to this matrix.

Parameters

<i>name</i>	holds the matrix name; use null or empty std::string ("") if no matrix name.
<i>numberOfRows</i>	holds the number of rows. It is required. Use 1 for column vectors.
<i>numberOfColumns</i>	holds the number of columns. It is required. Use 1 for row vectors.
<i>symmetry</i>	holds the type of symmetry used in the definition of the matrix. For more information see the enumeration ENUM_MATRIX_SYMMETRY in OSGeneral.h . If no symmetry, use ENUM_MATRIX_SYMMETRY_none.
<i>matrixType</i>	tracks the type of elements contained in this matrix. For more information see the enumeration ENUM_MATRIX_TYPE in OSGeneral.h . If unsure, use ENUM_MATRIX_TYPE_unknown.
<i>inumberOfChildren</i>	is the number of MatrixNode child elements, i.e., the number of matrix constructors in the m_mChildren array.
<i>m_mChildren</i>	is the array of matrix constructors used in the definition of this matrix.

Returns

whether the matrix was added successfully.

6.136.3.13 virtual std::string OSMatrix::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.136.3.14 OSMatrix * OSMatrix::cloneMatrixNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [MatrixNode](#) of the proper type.

Implements [MatrixNode](#).

6.136.3.15 bool OSMatrix::isEqual (OSMatrix * that)

A function to check for the equality of two objects.

6.136.3.16 `bool OSMatrix::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixType](#).

6.136.3.17 `bool OSMatrix::deepCopyFrom (OSMatrix * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.136.4 Member Data Documentation

6.136.4.1 `int OSMatrix::idx`

Definition at line 1993 of file OSMatrix.h.

6.136.4.2 `std::string OSMatrix::name`

Definition at line 1994 of file OSMatrix.h.

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

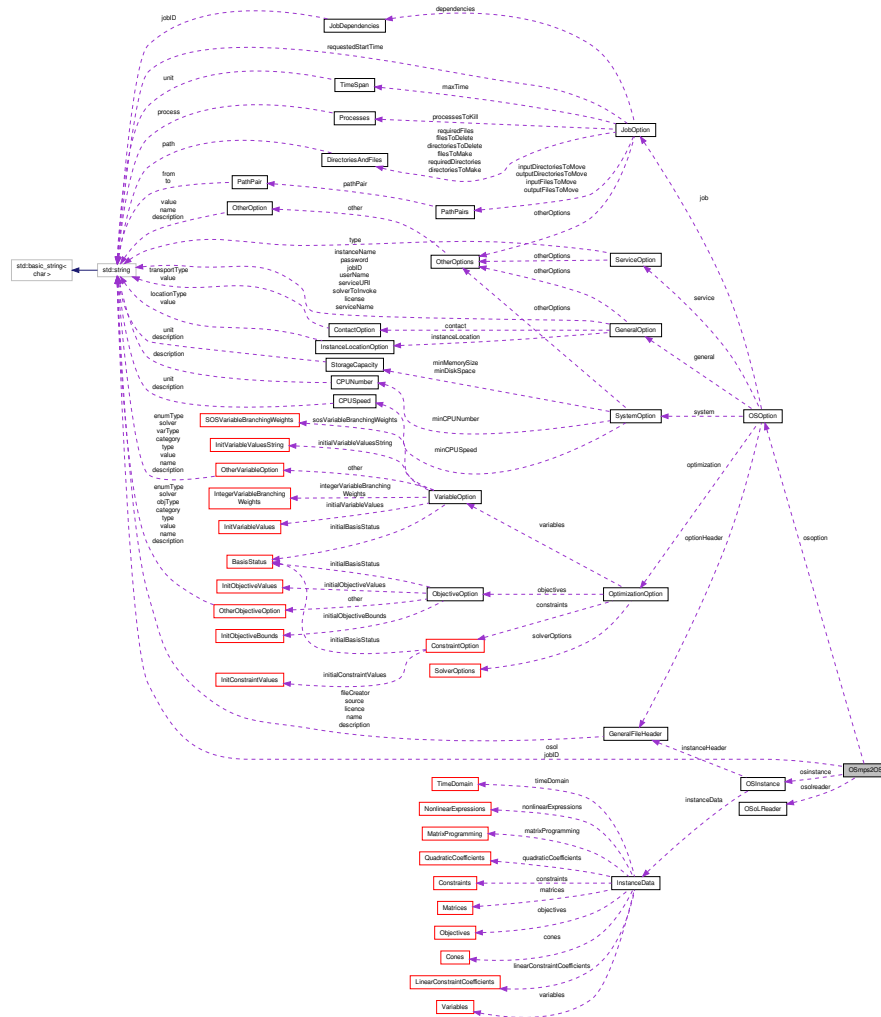
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSMatrix.h](#)

6.137 OSmps2OS Class Reference

The [OSmps2OS](#) Class.

```
#include <OSmps2OS.h>
```

Collaboration diagram for OSmps2OS:



Public Member Functions

- [OSmps2OS](#) (std::string mpsfilename)
the [OSmps2OS](#) class constructor
- [~OSmps2OS](#) ()
the [OSmps2os](#) class destructor
- void [setOsol](#) (std::string osol)
set the osol string
- void [setJobID](#) (std::string jobID)
set the job ID
- bool [createOSObjects](#) ()
create an [OSInstance](#) from the MPS instance representation and an [OSOption](#) in case of nonstandard sections such as SOS

Public Attributes

- [OSInstance](#) * [osinstance](#)
osinstance is a pointer to the [OSInstance](#) object that gets created from the instance represented in MPS format
- [OSOption](#) * [osoption](#)
osoption is a pointer to an [OSOption](#) object that gets created if the MPS file contains nonstandard sections such as SOS
- [OSoLReader](#) * [osolreader](#)
we may need to parse an OSoL file if the MPS file contains an SOS os BASIS section
- std::string [osol](#)
osol is a string containing the content of the OS option file (it may be empty if no option file was provided by the user).
- std::string [jobID](#)
jobID is a string containing a jobID that may have been supplied on the command line (it may be empty).

6.137.1 Detailed Description

The [OSmps2OS](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Remarks

the [OSmps2osil](#) class is used for reading an instance in MPS format and creating an [OSInstance](#) object in OSiL format and possibly an [OSOption](#) object in OSoL format (if the MPS file contains nonstandard sections such as SOS)

Definition at line 39 of file OSmps2OS.h.

6.137.2 Constructor & Destructor Documentation**6.137.2.1 OSmps2OS::OSmps2OS (std::string *mpsfilename*)**

the [OSmps2OS](#) class constructor

6.137.2.2 OSmps2OS::~~OSmps2OS ()

the OSmps2os class destructor

6.137.3 Member Function Documentation**6.137.3.1 void OSmps2OS::setOsol (std::string *osol*)**

set the osol string

6.137.3.2 void OSmps2OS::setJobID (std::string *jobID*)

set the job ID

6.137.3.3 bool OSmps2OS::createOSObjects ()

create an [OSInstance](#) from the MPS instance representation and an [OSOption](#) in case of nonstandard sections such as SOS

Returns

whether the objects are created successfully.

6.137.4 Member Data Documentation

6.137.4.1 OSInstance* OSmps2OS::osinstance

osinstance is a pointer to the [OSInstance](#) object that gets created from the instance represented in MPS format

Definition at line 65 of file OSmps2OS.h.

6.137.4.2 OSOption* OSmps2OS::osoption

osoption is a pointer to an [OSOption](#) object that gets created if the MPS file contains nonstandard sections such as SOS

Definition at line 70 of file OSmps2OS.h.

6.137.4.3 OSoLReader* OSmps2OS::osolreader

we may need to parse an OSoL file if the MPS file contains an SOS os BASIS section

Definition at line 76 of file OSmps2OS.h.

6.137.4.4 std::string OSmps2OS::osol

osol is a string containing the content of the OS option file (it may be empty if no option file was provided by the user).

If osoption is NULL, the option information is found in osol.

Definition at line 82 of file OSmps2OS.h.

6.137.4.5 std::string OSmps2OS::jobID

jobID is a string containing a jobID that may have been supplied on the command line (it may be empty).

If osoption is not NULL, the jobID has been duplicated to osoption.

Definition at line 88 of file OSmps2OS.h.

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

- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/[OSmps2OS.h](#)

6.138 OSmps2osil Class Reference

The [OSmps2osil](#) Class.

```
#include <OSmps2osil.h>
```

The diagram illustrates the dependency structure for the 'minimize' function. It shows a flow from various input parameters and data sources through intermediate processing nodes to the final 'minimize' function. Key inputs include 'minimizeObjective', 'minimizeConstraints', and 'minimizeVariables'. Intermediate nodes like 'minimizeObjectiveCoefficients', 'minimizeConstraintsCoefficients', and 'minimizeVariablesCoefficients' represent processed data. The graph also shows dependencies on 'minimizeCoefficients' and 'minimizeCoefficientsCoefficients'.

- `OSmps2osil` (`std::string mpsfilename`)
the *OSmps2osil* class constructor
- `~OSmps2osil` ()
the *OSmps2osil* class destructor
- `bool createOSInstance` ()
create an *OSInstance* from the MPS instance representation

- **OSInstance** * **osinstance**
osinstance is a pointer to the OSInstance object that gets created from the instance represented in NL format

The OSmps2osil Class.

Robert Fourer, Jun Ma, Kipp Martin

1.0, 03/14/2004

OS 1.0

the `OSmps2osil` class is used for reading an instance in MPS format and creating an `OSInstance` object in OSiL format

Generated on Mon Mar 16 2015 21:53:41 for OS by Doxygen

6.138.2 Constructor & Destructor Documentation

6.138.2.1 OSmps2osil::OSmps2osil (std::string *mpsfilename*)

the [OSmps2osil](#) class constructor

6.138.2.2 OSmps2osil::~~OSmps2osil ()

the [OSmps2osil](#) class destructor

6.138.3 Member Function Documentation

6.138.3.1 bool OSmps2osil::createOSInstance ()

create an [OSInstance](#) from the MPS instance representation

Returns

whether the instance is created successfully.

6.138.4 Member Data Documentation

6.138.4.1 OSInstance* OSmps2osil::osinstance

osinstance is a pointer to the [OSInstance](#) object that gets created from the instance represented in NL format

Definition at line 58 of file OSmps2osil.h.

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

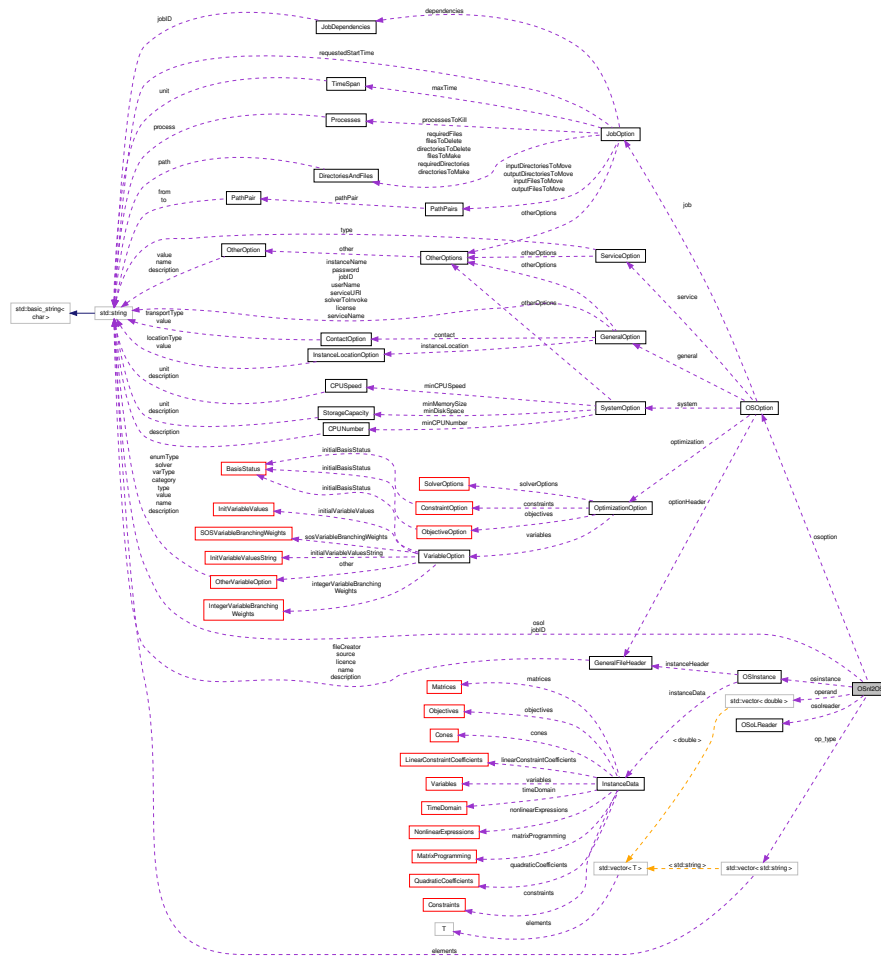
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/[OSmps2osil.h](#)

6.139 OSnl2OS Class Reference

The [OSnl2OS](#) Class.

```
#include <OSnl2OS.h>
```

Collaboration diagram for OSnl2OS:



Public Member Functions

- [OSnl2OS](#) ()
the [OSnl2OS](#) class constructor
- [OSnl2OS](#) (ASL *cw, ASL *rw, ASL *asl)
alternate constructor which does not allocate the ASL structs
- [~OSnl2OS](#) ()
the [OSnl2OS](#) class destructor
- ASL * [getASL](#) (std::string name)
return a pointer to an ASL object
- bool [readNI](#) (std::string stub)
read the nl file
- void [setOsol](#) (std::string osol)
set the osol string
- void [setJobID](#) (std::string jobId)
set the job ID

- bool [setASL](#) (ASL *asl, ASL *rw, ASL *cw)
set the pointers to the three ASL objects
- bool [createOSObjects](#) ()
create an [OSInstance](#) and [OSOption](#) representation from the AMPL nl content (Some of the information in the nl file — such as initial values, basis information, branching priorities, etc.
- void [setVar](#) ([OSInstance](#) *osinstance, int lower, int upper, char vartype)
store a number of variables into an [OSInstance](#) object
- void [setIBVar](#) ([OSInstance](#) *osinstance, int lower, int upper)
special version of the previous method because AMPL makes no distinction between integer and binary variables that occur in nonlinear expressions.
- [OSNLNode](#) * [walkTree](#) (expr *e)
parse an nl tree structure holding a nonlinear expression

Public Attributes

- [OSoLReader](#) * [osolreader](#)
we may need to parse an OSoL file if there is suffix information indicated in the AMPL nl content
- [OSInstance](#) * [osinstance](#)
osinstance is a pointer to the [OSInstance](#) object that gets created from the information in the nl file
- [OSOption](#) * [osoption](#)
osoption is a pointer to the [OSOption](#) object that gets created from the information in the nl file (and the OSoL string if present)
- std::string [osol](#)
osol is a string containing the content of the OS option file (it may be empty if no option file was provided).
- std::string [jobID](#)
jobID is a string containing a jobID that may have been supplied on the command line (it may be empty).
- std::vector< std::string > [op_type](#)
- std::vector< double > [operand](#)
- int [numkount](#)

6.139.1 Detailed Description

The [OSnl2OS](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

The [OSnl2OS](#) class is used for reading an AMPL nl file, extracting from it the instance along with any options indexed over variables, constraints, objectives. The latter may, if needed, be merged with the contents of an OSoL file. The processed information is stored into an [OSInstance](#) object and either an [OSOption](#) object (if not NULL) or an OSoL string.

Definition at line 78 of file OSnl2OS.h.

6.139.2 Constructor & Destructor Documentation

6.139.2.1 OSnl2OS::OSnl2OS ()

the [OSnl2OS](#) class constructor

6.139.2.2 OSnl2OS::OSnl2OS (ASL * *cw*, ASL * *rw*, ASL * *asl*)

alternate constructor which does not allocate the ASL structs

¶m *cw* a pointer to a (previously allocated) struct used for column-wise representation ¶m *rw* a pointer to a (previously allocated) struct used for row-wise representation ¶m *asl* an extra pointer that can be used to switch between *rw* and *cw*

6.139.2.3 OSnl2OS::~~OSnl2OS ()

the [OSnl2OS](#) class destructor

6.139.3 Member Function Documentation

6.139.3.1 ASL* OSnl2OS::getASL (std::string *name*)

return a pointer to an ASL object

Parameters

<i>name</i>	carries the name of the ASL object (there are three of them: <i>asl</i> , <i>rw</i> , <i>cw</i>)
-------------	---

Returns

the pointer to the object named

6.139.3.2 bool OSnl2OS::readNI (std::string *stub*)

read the *nl* file

Parameters

<i>stub</i>	is the (relevant part of the) file name
-------------	---

Returns

whether the read was successful

6.139.3.3 void OSnl2OS::setOsol (std::string *osol*)

set the *osol* string

6.139.3.4 void OSnl2OS::setJobID (std::string *jobID*)

set the job ID

6.139.3.5 bool OSnl2OS::setASL (ASL * *asl*, ASL * *rw*, ASL * *cw*)

set the pointers to the three ASL objects

Parameters

<i>asl</i>	carries a pointer to the object named "asl"
<i>rw</i>	carries a pointer to the object named "rw"
<i>cw</i>	carries a pointer to the object named "cw" (asl should point to the same location as either rw or cw)

Returns

whether the operation was successful

6.139.3.6 bool OSnl2OS::createOSObjects ()

create an [OSInstance](#) and [OSOption](#) representation from the AMPL nl content (Some of the information in the nl file — such as initial values, basis information, branching priorities, etc.

— cannot be stored into an [OSInstance](#) and must be stored in an [OSOption](#) object instead.)

Returns

whether the objects were created successfully.

6.139.3.7 void OSnl2OS::setVar (OSInstance * *osinstance*, int *lower*, int *upper*, char *vartype*)

store a number of variables into an [OSInstance](#) object

Parameters

<i>osinstance,:</i>	a pointer to the OSInstance object
<i>lower,:</i>	index of the first variable to be set in this call
<i>upper,:</i>	set all variables from lower...upper-1
<i>vartype,:</i>	the type of the variable (in AMPL this is 'C', 'B' or 'I')

6.139.3.8 void OSnl2OS::setIBVar (OSInstance * *osinstance*, int *lower*, int *upper*)

special version of the previous method because AMPL makes no distinction between integer and binary variables that occur in nonlinear expressions.

The actual type ('B' or 'I') must be inferred from the variable bounds.

Parameters

<i>osinstance,:</i>	a pointer to the OSInstance object
<i>lower,:</i>	index of the first variable to be set in this call
<i>upper,:</i>	set all variables from lower...upper-1

6.139.3.9 OSnLNode* OSnl2OS::walkTree (expr * *e*)

parse an nl tree structure holding a nonlinear expression

Returns

the AMPL nonlinear structure as an [OSnLNode](#).

6.139.4 Member Data Documentation**6.139.4.1 OSoLReader* OSnl2OS::osolreader**

we may need to parse an OSoL file if there is suffix information indicated in the AMPL nl content

Definition at line 160 of file OSnl2OS.h.

6.139.4.2 OSInstance* OSnl2OS::osinstance

osinstance is a pointer to the [OSInstance](#) object that gets created from the information in the nl file

Definition at line 172 of file OSnl2OS.h.

6.139.4.3 OSOption* OSnl2OS::osoption

osoption is a pointer to the [OSOption](#) object that gets created from the information in the nl file (and the OSoL string if present)

Definition at line 177 of file OSnl2OS.h.

6.139.4.4 std::string OSnl2OS::osol

osol is a string containing the content of the OS option file (it may be empty if no option file was provided).

If osoption is NULL, the option information is found in osol.

Definition at line 183 of file OSnl2OS.h.

6.139.4.5 std::string OSnl2OS::jobID

jobID is a string containing a jobID that may have been supplied on the command line (it may be empty).

If osoption is not NULL, the jobID has been duplicated to osoption.

Definition at line 189 of file OSnl2OS.h.

6.139.4.6 std::vector<std::string> OSnl2OS::op_type

Definition at line 191 of file OSnl2OS.h.

6.139.4.7 std::vector<double> OSnl2OS::operand

Definition at line 192 of file OSnl2OS.h.

6.139.4.8 int OSnl2OS::numkount

Definition at line 193 of file OSnl2OS.h.

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

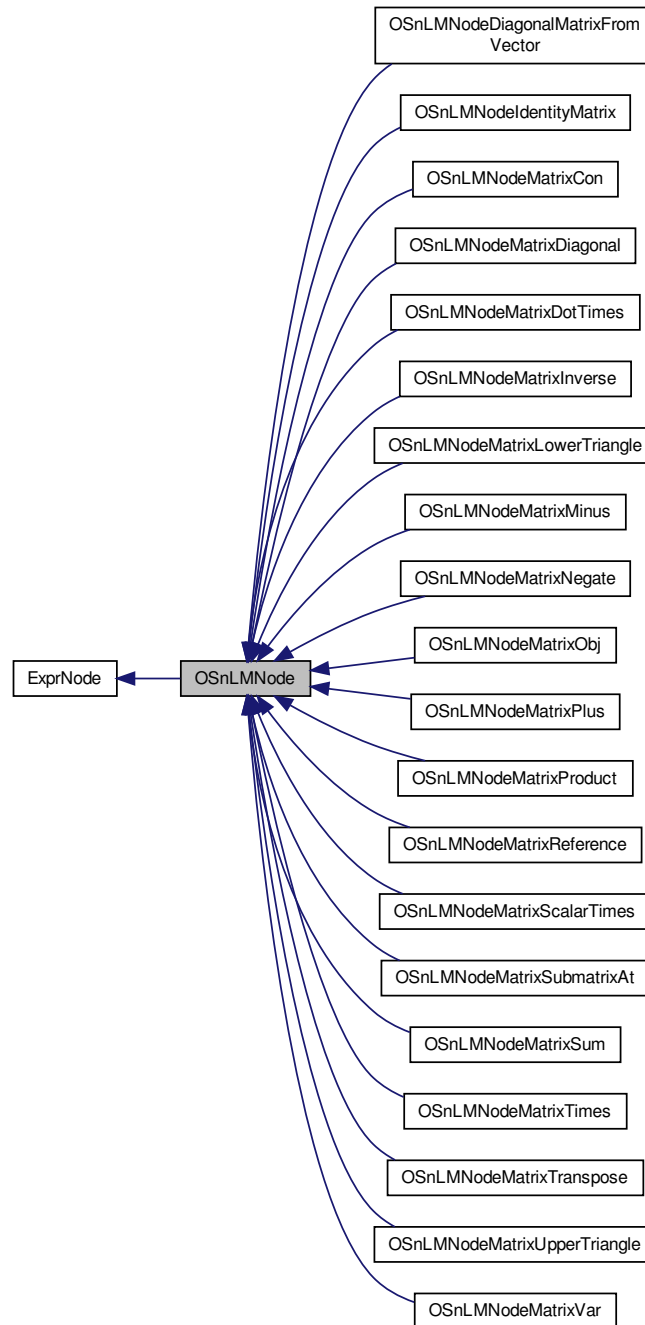
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/[OSnl2OS.h](#)

6.140 OSnLMNode Class Reference

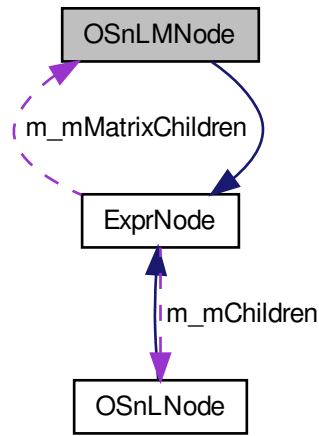
The [OSnLMNode](#) Class for nonlinear expressions involving matrices.


```
#include <OSnLMNode.h>
```

Inheritance diagram for OSnLMNode:



Collaboration diagram for OSnLMNode:



Public Member Functions

- [OSnLMNode \(\)](#)
default constructor.
- [virtual ~OSnLMNode \(\)](#)
default destructor.
- [OSnLMNode * createExpressionTreeFromPrefix](#) (std::vector< [ExprNode *](#) > nINodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a matrix-valued [OSExpression-Tree](#) root node.
- [std::vector< ExprNode * > getPrefixFromExpressionTree \(\)](#)
Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the (matrix-valued) expression tree in prefix format.
- [std::vector< ExprNode * > preOrderOSnLNodeTraversal](#) (std::vector< [ExprNode *](#) > *prefixVector)
Called by [getPrefixFromExpressionTree\(\)](#).
- [OSnLMNode * createExpressionTreeFromPostfix](#) (std::vector< [ExprNode *](#) > nINodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a matrix-valued [OSExpression-Tree](#) root node.
- [std::vector< ExprNode * > getPostfixFromExpressionTree \(\)](#)
Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.
- [std::vector< ExprNode * > postOrderOSnLNodeTraversal](#) (std::vector< [ExprNode *](#) > *postfixVector)
Called by [getPostfixFromExpressionTree\(\)](#).
- [OSnLMNode * copyNodeAndDescendants \(\)](#)
make a copy of this node and all its descendants
- [bool isEqual](#) ([OSnLMNode *](#)that)
A function to check for the equality of two objects.

Additional Inherited Members

6.140.1 Detailed Description

The [OSnLMNode](#) Class for nonlinear expressions involving matrices.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Date

11/06/2014

Since

OS2.8

Definition at line 1749 of file OSnLMNode.h.

6.140.2 Constructor & Destructor Documentation

6.140.2.1 OSnLMNode::OSnLMNode ()

default constructor.

6.140.2.2 virtual OSnLMNode::~~OSnLMNode () [virtual]

default destructor.

6.140.3 Member Function Documentation

6.140.3.1 OSnLMNode* OSnLMNode::createExpressionTreeFromPrefix (std::vector< ExprNode * > nNodeVec)

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a matrix-valued [OSExpressionTree](#) root node.

Parameters

<i>nNodeVec</i>	holds a vector of pointers to OSnLNodes and OSnLMNodes in prefix format
-----------------	---

Returns

a pointer to an [OSnLMNode](#) which is the root of an [OSExpressionTree](#).

6.140.3.2 std::vector<ExprNode*> OSnLMNode::getPrefixFromExpressionTree () [virtual]

Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the (matrix-valued) expression tree in prefix format.

Returns

the expression tree as a vector of ExprNodes in prefix.

Reimplemented from [ExprNode](#).

6.140.3.3 `std::vector<ExprNode*> OSnLMNode::preOrderOSnLNodeTraversal (std::vector< ExprNode * > * prefixVector)`
[virtual]

Called by [getPrefixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to [ExprNode](#) in prefix

Parameters

<i>a</i>	pointer prefixVector to a vector of pointers of ExprNodes
----------	---

Returns

a vector of pointers to [ExprNode](#) in prefix.

Reimplemented from [ExprNode](#).

6.140.3.4 `OSnLMNode* OSnLMNode::createExpressionTreeFromPostfix (std::vector< ExprNode * > nINodeVec)`

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a matrix-valued [OS-ExpressionTree](#) root node.

Parameters

<i>nINodeVec</i>	holds a vector of pointers to OSnLNodes and OSnLMNodes in postfix format
------------------	--

Returns

a pointer to an [OSnLMNode](#) which is the root of an [OSExpressionTree](#).

6.140.3.5 `std::vector<ExprNode*> OSnLMNode::getPostfixFromExpressionTree ()` [virtual]

Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.

Returns

the expression tree as a vector of ExprNodes in postfix.

Reimplemented from [ExprNode](#).

6.140.3.6 `std::vector<ExprNode*> OSnLMNode::postOrderOSnLNodeTraversal (std::vector< ExprNode * > * postfixVector)`
[virtual]

Called by [getPostfixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to ExprNodes in postfix.

Parameters

<i>a</i>	pointer postfixVector to a vector of pointers of ExprNodes
----------	--

Returns

a vector of pointers to ExprNodes in postfix.

Reimplemented from [ExprNode](#).

6.140.3.7 OSnLMNode* OSnLMNode::copyNodeAndDescendants ()

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

6.140.3.8 bool OSnLMNode::isEqual (OSnLMNode * that)

A function to check for the equality of two objects.

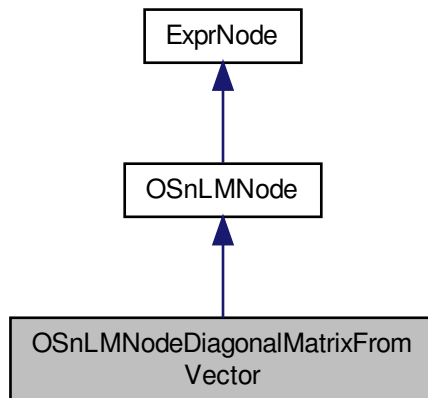
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

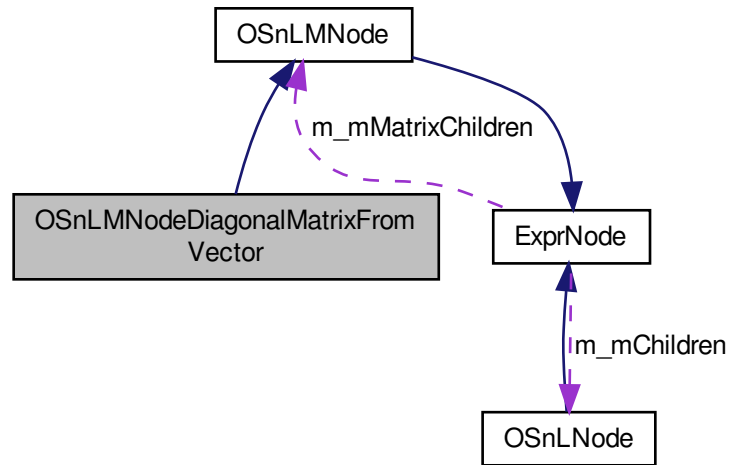
6.141 OSnLMNodeDiagonalMatrixFromVector Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeDiagonalMatrixFromVector:



Collaboration diagram for OSnLMNodeDiagonalMatrixFromVector:



Public Member Functions

- [OSnLMNodeDiagonalMatrixFromVector \(\)](#)
default constructor.
- [~OSnLMNodeDiagonalMatrixFromVector \(\)](#)
default destructor.
- virtual `std::string` [getTokenName \(\)](#)
- virtual `OSnLMNode *` [cloneExprNode \(\)](#)
Create or clone a node of this type.

Additional Inherited Members

6.141.1 Detailed Description

Definition at line 2359 of file OSnLNode.h.

6.141.2 Constructor & Destructor Documentation

6.141.2.1 OSnLMNodeDiagonalMatrixFromVector::OSnLMNodeDiagonalMatrixFromVector ()

default constructor.

6.141.2.2 OSnLMNodeDiagonalMatrixFromVector::~~OSnLMNodeDiagonalMatrixFromVector ()

default destructor.

6.141.3 Member Function Documentation

6.141.3.1 `virtual std::string OSnLMNodeDiagonalMatrixFromVector::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

6.141.3.2 `virtual OSnLMNode* OSnLMNodeDiagonalMatrixFromVector::cloneExprNode () [virtual]`

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

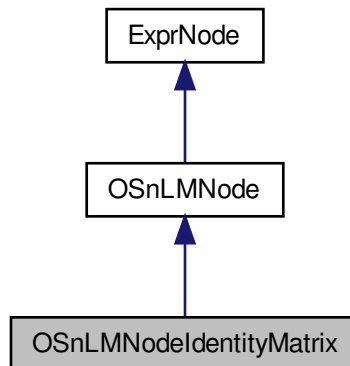
The documentation for this class was generated from the following file:

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

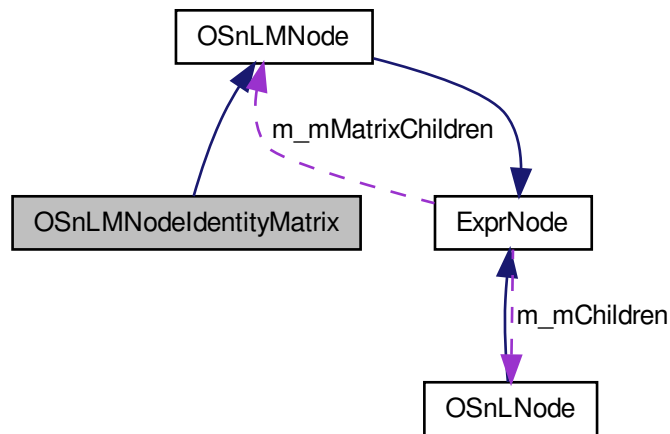
6.142 OSnLMNodeIdentityMatrix Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeIdentityMatrix:



Collaboration diagram for OSnLMNodeIdentityMatrix:



Public Member Functions

- [OSnLMNodeIdentityMatrix \(\)](#)
default constructor.
- [~OSnLMNodeIdentityMatrix \(\)](#)
default destructor.
- virtual `std::string` [getTokenName \(\)](#)
- virtual `OSnLMNode *` [cloneExprNode \(\)](#)
Create or clone a node of this type.

Additional Inherited Members

6.142.1 Detailed Description

Definition at line 2186 of file OSnLNode.h.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 OSnLMNodeIdentityMatrix::OSnLMNodeIdentityMatrix ()

default constructor.

6.142.2.2 OSnLMNodeIdentityMatrix::~~OSnLMNodeIdentityMatrix ()

default destructor.

6.142.3 Member Function Documentation

6.142.3.1 `virtual std::string OSnLMNodeIdentityMatrix::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.142.3.2 `virtual OSnLMNode* OSnLMNodeIdentityMatrix::cloneExprNode ()` [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

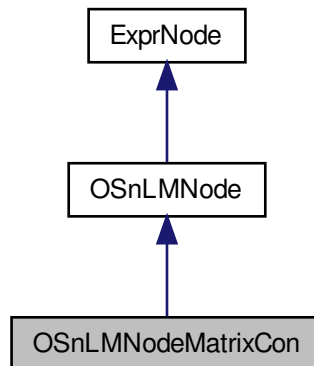
The documentation for this class was generated from the following file:

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

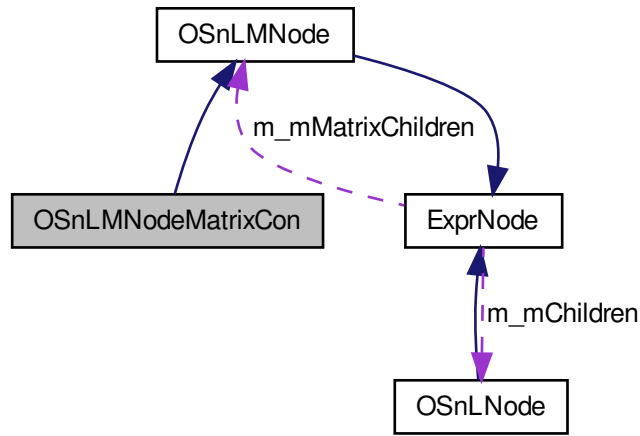
6.143 OSnLMNodeMatrixCon Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixCon:



Collaboration diagram for OSnLMNodeMatrixCon:



Public Member Functions

- [OSnLMNodeMatrixCon \(\)](#)
default constructor.
- [~OSnLMNodeMatrixCon \(\)](#)
default destructor.
- virtual std::string [getTokenName \(\)](#)
- virtual std::string [getTokenNumber \(\)](#)
- virtual std::string [getNonlinearExpressionInXML \(\)](#)
- virtual [OSnLMNode *](#) [cloneExprNode \(\)](#)
Create or clone a node of this type.
- virtual bool [IsEqual \(OSnLMNodeMatrixCon *that\)](#)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
The index of the matrixCon.

6.143.1 Detailed Description

Definition at line 2632 of file OSnLNode.h.

6.143.2 Constructor & Destructor Documentation

6.143.2.1 OSnLMNodeMatrixCon::OSnLMNodeMatrixCon ()

default constructor.

6.143.2.2 OSnLMNodeMatrixCon::~OSnLMNodeMatrixCon ()

default destructor.

6.143.3 Member Function Documentation

6.143.3.1 virtual std::string OSnLMNodeMatrixCon::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.143.3.2 virtual std::string OSnLMNodeMatrixCon::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.143.3.3 virtual std::string OSnLMNodeMatrixCon::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the [OSnLMNode](#) <matrixCon>.

Reimplemented from [ExprNode](#).

6.143.3.4 virtual OSnLMNode* OSnLMNodeMatrixCon::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.143.3.5 virtual bool OSnLMNodeMatrixCon::isEqual (OSnLMNodeMatrixCon * *that*) [virtual]

A function to check for the equality of two objects.

6.143.4 Member Data Documentation

6.143.4.1 int OSnLMNodeMatrixCon::idx

The index of the matrixCon.

Definition at line 2638 of file OSnLNode.h.

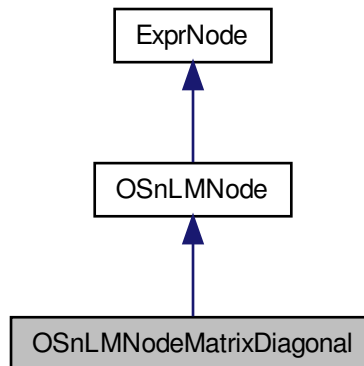
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

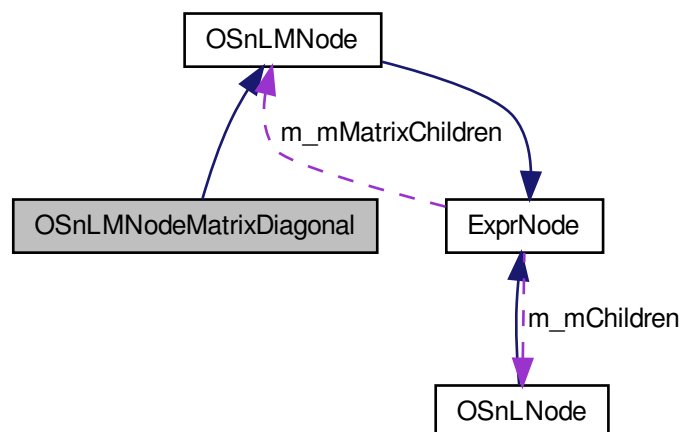
6.144 OSnLMNodeMatrixDiagonal Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixDiagonal:



Collaboration diagram for OSnLMNodeMatrixDiagonal:



Public Member Functions

- [OSnLMNodeMatrixDiagonal](#) ()
default constructor.
- [~OSnLMNodeMatrixDiagonal](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.144.1 Detailed Description

Definition at line 2321 of file OSnLNode.h.

6.144.2 Constructor & Destructor Documentation

6.144.2.1 OSnLMNodeMatrixDiagonal::OSnLMNodeMatrixDiagonal ()

default constructor.

6.144.2.2 OSnLMNodeMatrixDiagonal::~~OSnLMNodeMatrixDiagonal ()

default destructor.

6.144.3 Member Function Documentation

6.144.3.1 virtual std::string OSnLMNodeMatrixDiagonal::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.144.3.2 virtual OSnLMNode* OSnLMNodeMatrixDiagonal::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

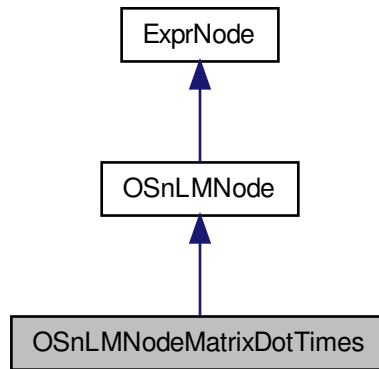
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

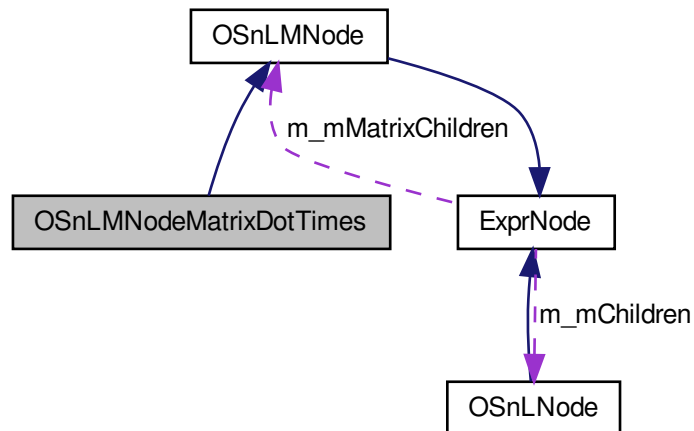
6.145 OSnLMNodeMatrixDotTimes Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixDotTimes:



Collaboration diagram for OSnLMNodeMatrixDotTimes:



Public Member Functions

- [OSnLMNodeMatrixDotTimes](#) ()
default constructor.
- [~OSnLMNodeMatrixDotTimes](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.145.1 Detailed Description

Definition at line 2148 of file OSnLNode.h.

6.145.2 Constructor & Destructor Documentation

6.145.2.1 OSnLMNodeMatrixDotTimes::OSnLMNodeMatrixDotTimes ()

default constructor.

6.145.2.2 OSnLMNodeMatrixDotTimes::~~OSnLMNodeMatrixDotTimes ()

default destructor.

6.145.3 Member Function Documentation

6.145.3.1 virtual std::string OSnLMNodeMatrixDotTimes::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.145.3.2 virtual OSnLMNode* OSnLMNodeMatrixDotTimes::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

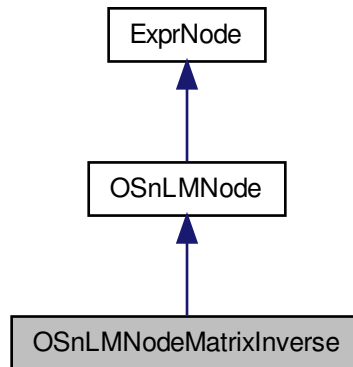
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

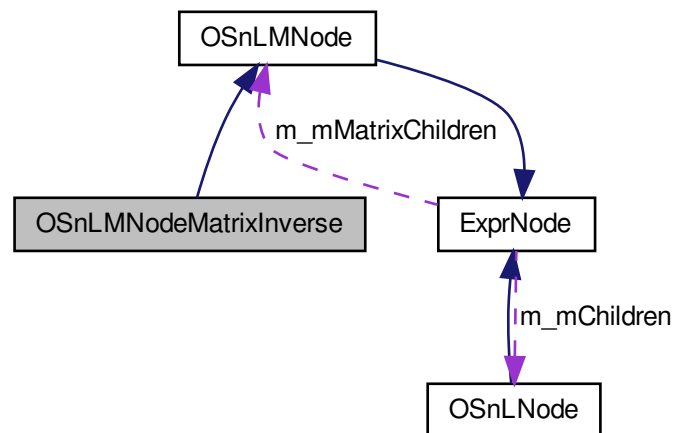
6.146 OSnLMNodeMatrixInverse Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixInverse:



Collaboration diagram for OSnLMNodeMatrixInverse:



Public Member Functions

- [OSnLMNodeMatrixInverse](#) ()
default constructor.
- [~OSnLMNodeMatrixInverse](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.146.1 Detailed Description

Definition at line 2034 of file OSnLNode.h.

6.146.2 Constructor & Destructor Documentation

6.146.2.1 OSnLMNodeMatrixInverse::OSnLMNodeMatrixInverse ()

default constructor.

6.146.2.2 OSnLMNodeMatrixInverse::~~OSnLMNodeMatrixInverse ()

default destructor.

6.146.3 Member Function Documentation

6.146.3.1 virtual std::string OSnLMNodeMatrixInverse::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.146.3.2 virtual OSnLMNode* OSnLMNodeMatrixInverse::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

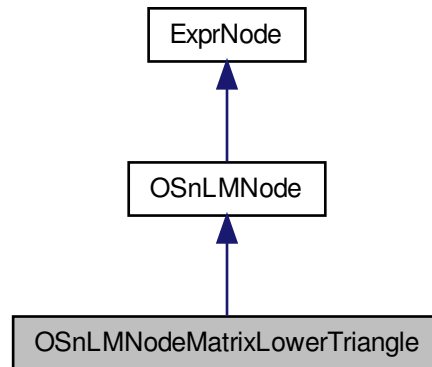
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

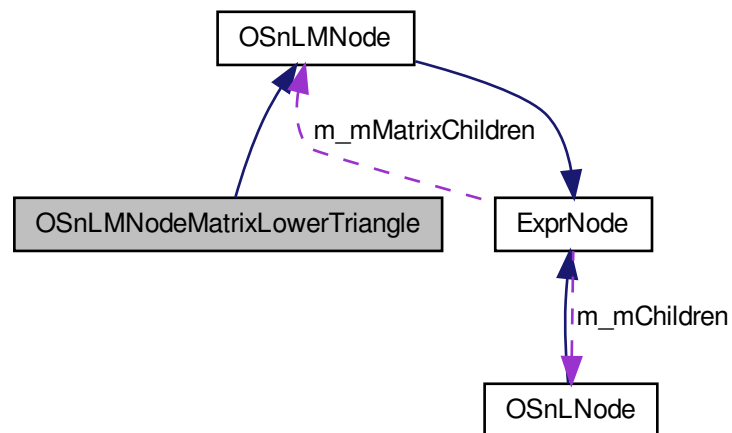
6.147 OSnLMNodeMatrixLowerTriangle Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixLowerTriangle:



Collaboration diagram for OSnLMNodeMatrixLowerTriangle:



Public Member Functions

- [OSnLMNodeMatrixLowerTriangle \(\)](#)
default constructor.
- [~OSnLMNodeMatrixLowerTriangle \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.
- virtual bool [isEqual](#) ([OSnLMNodeMatrixLowerTriangle](#) *that)
A function to check for the equality of two objects.

Public Attributes

- bool [includeDiagonal](#)
A boolean to express whether the diagonal is to be part of the upper triangle or not.

6.147.1 Detailed Description

Definition at line 2224 of file OSnLMNode.h.

6.147.2 Constructor & Destructor Documentation

6.147.2.1 OSnLMNodeMatrixLowerTriangle::OSnLMNodeMatrixLowerTriangle ()

default constructor.

6.147.2.2 OSnLMNodeMatrixLowerTriangle::~OSnLMNodeMatrixLowerTriangle ()

default destructor.

6.147.3 Member Function Documentation

6.147.3.1 virtual std::string OSnLMNodeMatrixLowerTriangle::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.147.3.2 virtual std::string OSnLMNodeMatrixLowerTriangle::getNonlinearExpressionInXML () [virtual]

Returns

a string token that corresponds to the [OSnLMNode](#).
the OSiL XML for the [OSnLMNode](#) <matrix>.

Reimplemented from [ExprNode](#).

6.147.3.3 virtual OSnLMNode* OSnLMNodeMatrixLowerTriangle::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.147.3.4 virtual bool OSnLMNodeMatrixLowerTriangle::isEqual (OSnLMNodeMatrixLowerTriangle * *that*) [virtual]

A function to check for the equality of two objects.

6.147.4 Member Data Documentation

6.147.4.1 bool OSnLMNodeMatrixLowerTriangle::includeDiagonal

A boolean to express whether the diagonal is to be part of the upper triangle or not.

Definition at line 2230 of file OSnLNode.h.

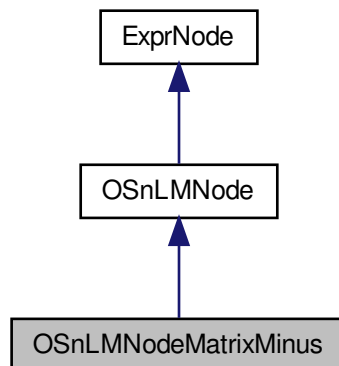
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

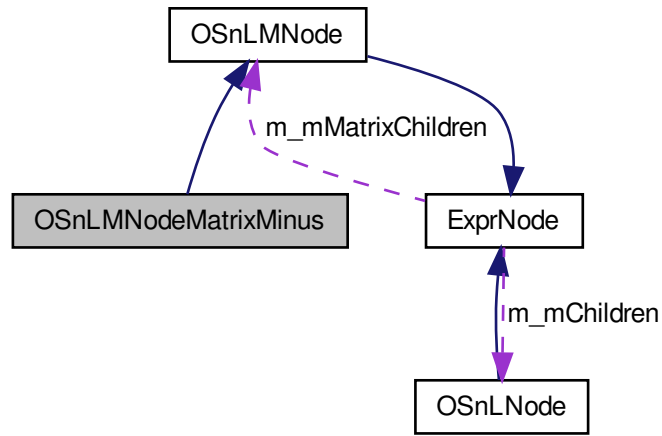
6.148 OSnLMNodeMatrixMinus Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixMinus:



Collaboration diagram for OSnLMNodeMatrixMinus:



Public Member Functions

- [OSnLMNodeMatrixMinus \(\)](#)
default constructor.
- [~OSnLMNodeMatrixMinus \(\)](#)
default destructor.
- virtual `std::string` [getTokenName \(\)](#)
- virtual `OSnLMNode *` [cloneExprNode \(\)](#)
Create or clone a node of this type.

Additional Inherited Members

6.148.1 Detailed Description

Definition at line 1919 of file OSnLNode.h.

6.148.2 Constructor & Destructor Documentation

6.148.2.1 OSnLMNodeMatrixMinus::OSnLMNodeMatrixMinus ()

default constructor.

6.148.2.2 OSnLMNodeMatrixMinus::~~OSnLMNodeMatrixMinus ()

default destructor.

6.148.3 Member Function Documentation

6.148.3.1 `virtual std::string OSnLMNodeMatrixMinus::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.148.3.2 `virtual OSnLMNode* OSnLMNodeMatrixMinus::cloneExprNode ()` [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

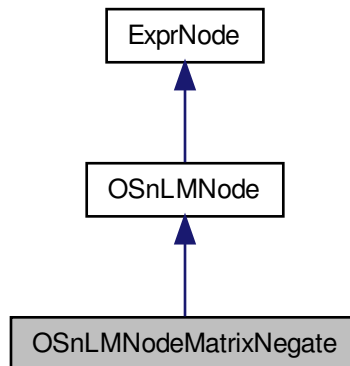
The documentation for this class was generated from the following file:

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

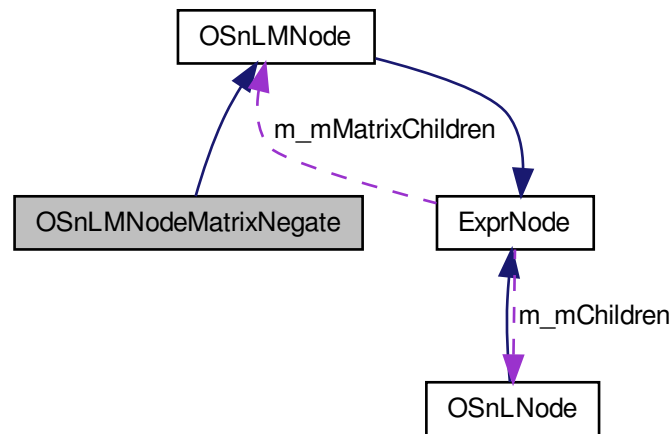
6.149 OSnLMNodeMatrixNegate Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixNegate:



Collaboration diagram for OSnLMNodeMatrixNegate:



Public Member Functions

- [OSnLMNodeMatrixNegate \(\)](#)
default constructor.
- [~OSnLMNodeMatrixNegate \(\)](#)
default destructor.
- virtual `std::string` [getTokenName \(\)](#)
- virtual `OSnLMNode *` [cloneExprNode \(\)](#)
Create or clone a node of this type.

Additional Inherited Members

6.149.1 Detailed Description

Definition at line 1957 of file OSnLNode.h.

6.149.2 Constructor & Destructor Documentation

6.149.2.1 OSnLMNodeMatrixNegate::OSnLMNodeMatrixNegate ()

default constructor.

6.149.2.2 OSnLMNodeMatrixNegate::~~OSnLMNodeMatrixNegate ()

default destructor.

6.149.3 Member Function Documentation

6.149.3.1 `virtual std::string OSnLMNodeMatrixNegate::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.149.3.2 `virtual OSnLMNode* OSnLMNodeMatrixNegate::cloneExprNode ()` [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

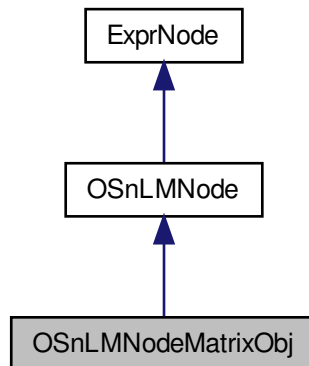
The documentation for this class was generated from the following file:

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

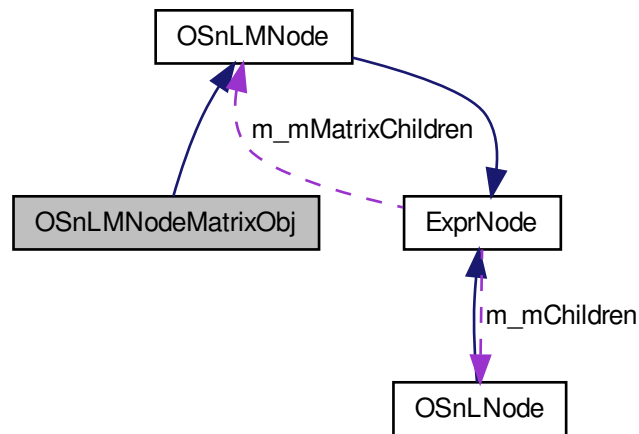
6.150 OSnLMNodeMatrixObj Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixObj:



Collaboration diagram for OSnLMNodeMatrixObj:



Public Member Functions

- [OSnLMNodeMatrixObj \(\)](#)
default constructor.
- [~OSnLMNodeMatrixObj \(\)](#)
default destructor.
- virtual [std::string getTokenName \(\)](#)
- virtual [std::string getTokenNumber \(\)](#)
- virtual [std::string getNonlinearExpressionInXML \(\)](#)
- virtual [OSnLMNode * cloneExprNode \(\)](#)
Create or clone a node of this type.
- virtual [bool isEqual \(OSnLMNodeMatrixObj *that\)](#)
A function to check for the equality of two objects.

Public Attributes

- [int idx](#)
The index of the matrixObj.

6.150.1 Detailed Description

Definition at line 2567 of file OSnLNode.h.

6.150.2 Constructor & Destructor Documentation

6.150.2.1 OSnLMNodeMatrixObj::OSnLMNodeMatrixObj ()

default constructor.

6.150.2.2 OSnLMNodeMatrixObj::~~OSnLMNodeMatrixObj ()

default destructor.

6.150.3 Member Function Documentation

6.150.3.1 virtual std::string OSnLMNodeMatrixObj::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.150.3.2 virtual std::string OSnLMNodeMatrixObj::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.150.3.3 virtual std::string OSnLMNodeMatrixObj::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the [OSnLMNode](#) <matrixObj>.

Reimplemented from [ExprNode](#).

6.150.3.4 virtual OSnLMNode* OSnLMNodeMatrixObj::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.150.3.5 virtual bool OSnLMNodeMatrixObj::isEqual (OSnLMNodeMatrixObj * that) [virtual]

A function to check for the equality of two objects.

6.150.4 Member Data Documentation

6.150.4.1 int OSnLMNodeMatrixObj::idx

The index of the matrixObj.

Definition at line 2573 of file OSnLNode.h.

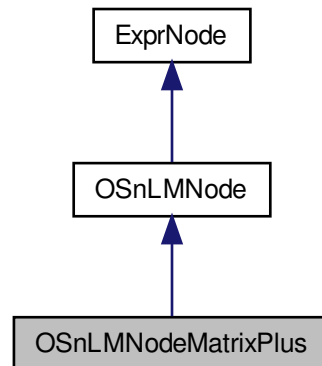
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

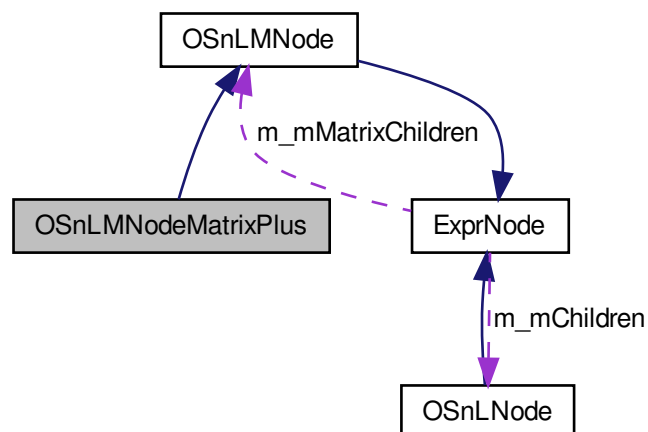
6.151 OSnLMNodeMatrixPlus Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixPlus:



Collaboration diagram for OSnLMNodeMatrixPlus:



Public Member Functions

- [OSnLMNodeMatrixPlus](#) ()
default constructor.
- [~OSnLMNodeMatrixPlus](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.151.1 Detailed Description

Definition at line 1835 of file OSnLNode.h.

6.151.2 Constructor & Destructor Documentation

6.151.2.1 OSnLMNodeMatrixPlus::OSnLMNodeMatrixPlus ()

default constructor.

6.151.2.2 OSnLMNodeMatrixPlus::~~OSnLMNodeMatrixPlus ()

default destructor.

6.151.3 Member Function Documentation

6.151.3.1 virtual std::string OSnLMNodeMatrixPlus::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.151.3.2 virtual OSnLMNode* OSnLMNodeMatrixPlus::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

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

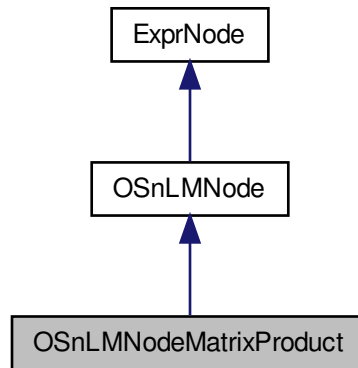
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.152 OSnLMNodeMatrixProduct Class Reference

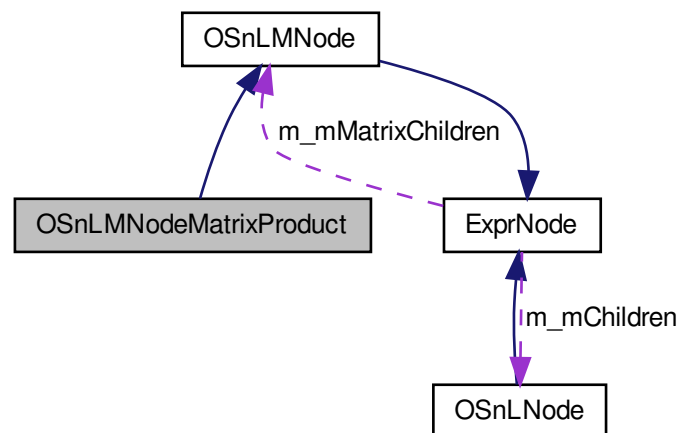
The [OSnLMNodeMatrixProduct](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixProduct:



Collaboration diagram for OSnLMNodeMatrixProduct:



Public Member Functions

- [OSnLMNodeMatrixProduct](#) ()
default constructor.
- [~OSnLMNodeMatrixProduct](#) ()

default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

The implementation of the virtual functions.

Additional Inherited Members

6.152.1 Detailed Description

The [OSnLMNodeMatrixProduct](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 08/Dec/2014

Since

OS2.9

Remarks

The in-memory representation of the OSnL element <matrixProduct>

Definition at line 2708 of file OSnLNode.h.

6.152.2 Constructor & Destructor Documentation

6.152.2.1 OSnLMNodeMatrixProduct::OSnLMNodeMatrixProduct ()

default constructor.

6.152.2.2 OSnLMNodeMatrixProduct::~OSnLMNodeMatrixProduct ()

default destructor.

6.152.3 Member Function Documentation

6.152.3.1 virtual std::string OSnLMNodeMatrixProduct::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.152.3.2 OSnLMNode * OSnLMNodeMatrixProduct::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLMNode](#) of the proper type.

Implements [ExprNode](#).

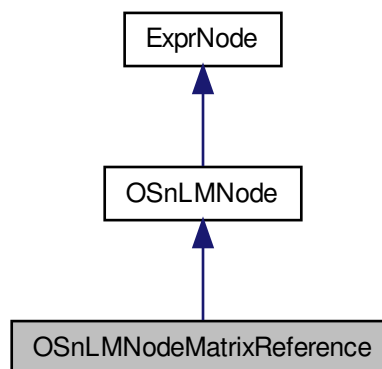
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLMNode.h](#)

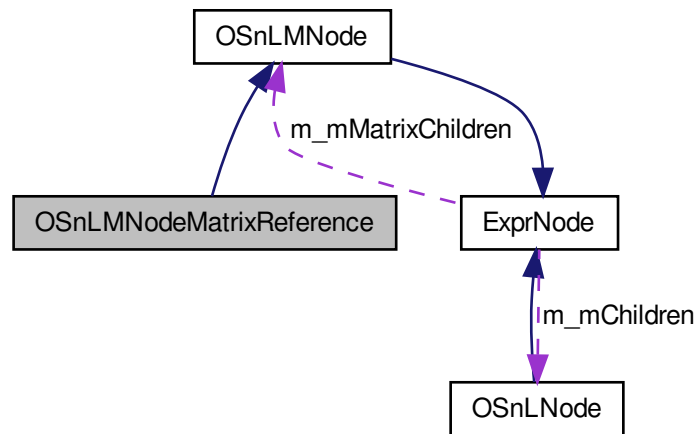
6.153 OSnLMNodeMatrixReference Class Reference

```
#include <OSnLMNode.h>
```

Inheritance diagram for OSnLMNodeMatrixReference:



Collaboration diagram for OSnLMNodeMatrixReference:



Public Member Functions

- [OSnLMNodeMatrixReference \(\)](#)
default constructor.
- [~OSnLMNodeMatrixReference \(\)](#)
default destructor.
- virtual std::string [getTokenName \(\)](#)
- virtual std::string [getTokenNumber \(\)](#)
- virtual std::string [getNonlinearExpressionInXML \(\)](#)
- virtual [OSnLMNode *](#) [cloneExprNode \(\)](#)
Create or clone a node of this type.
- virtual bool [isEqual \(OSnLMNodeMatrixReference *that\)](#)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
The index of the matrix.

6.153.1 Detailed Description

Definition at line 2437 of file OSnLNode.h.

6.153.2 Constructor & Destructor Documentation

6.153.2.1 OSnLMNodeMatrixReference::OSnLMNodeMatrixReference ()

default constructor.

6.153.2.2 OSnLMNodeMatrixReference::~OSnLMNodeMatrixReference ()

default destructor.

6.153.3 Member Function Documentation

6.153.3.1 virtual std::string OSnLMNodeMatrixReference::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.153.3.2 virtual std::string OSnLMNodeMatrixReference::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.153.3.3 virtual std::string OSnLMNodeMatrixReference::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the [OSnLMNode](#) <matrixReference>.

Reimplemented from [ExprNode](#).

6.153.3.4 virtual OSnLMNode* OSnLMNodeMatrixReference::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.153.3.5 virtual bool OSnLMNodeMatrixReference::isEqual (OSnLMNodeMatrixReference * *that*) [virtual]

A function to check for the equality of two objects.

6.153.4 Member Data Documentation

6.153.4.1 int OSnLMNodeMatrixReference::idx

The index of the matrix.

Definition at line 2443 of file OSnLNode.h.

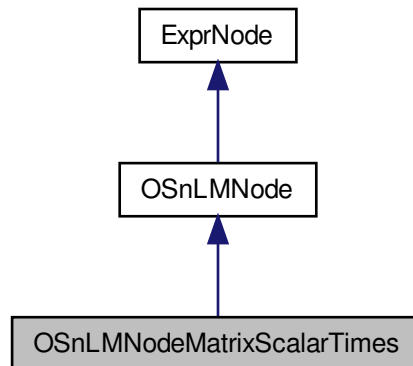
The documentation for this class was generated from the following file:

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

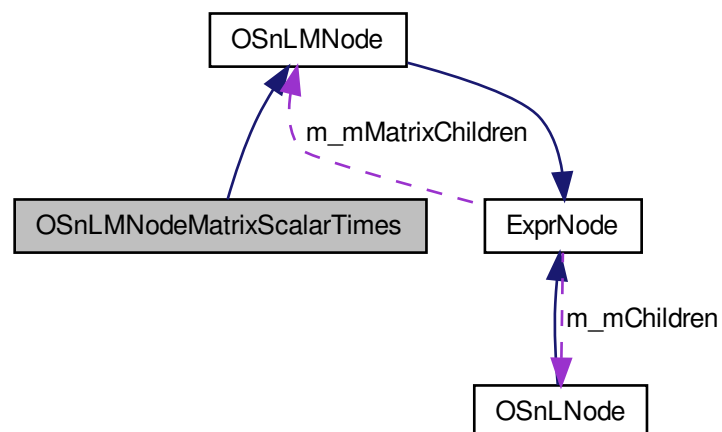
6.154 OSnLMNodeMatrixScalarTimes Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixScalarTimes:



Collaboration diagram for OSnLMNodeMatrixScalarTimes:



Public Member Functions

- [OSnLMNodeMatrixScalarTimes](#) ()
default constructor.
- [~OSnLMNodeMatrixScalarTimes](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.154.1 Detailed Description

Definition at line 2110 of file OSnLNode.h.

6.154.2 Constructor & Destructor Documentation

6.154.2.1 OSnLMNodeMatrixScalarTimes::OSnLMNodeMatrixScalarTimes ()

default constructor.

6.154.2.2 OSnLMNodeMatrixScalarTimes::~~OSnLMNodeMatrixScalarTimes ()

default destructor.

6.154.3 Member Function Documentation

6.154.3.1 virtual std::string OSnLMNodeMatrixScalarTimes::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.154.3.2 virtual OSnLMNode* OSnLMNodeMatrixScalarTimes::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

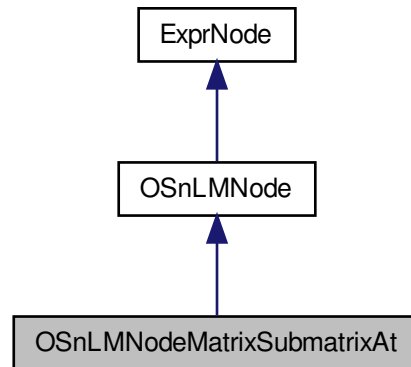
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

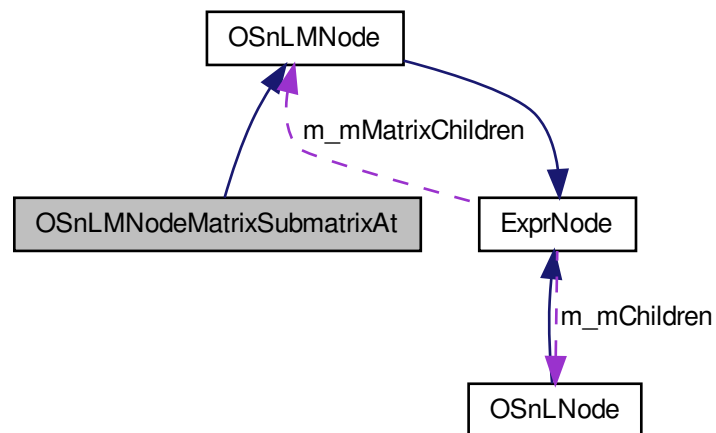
6.155 OSnLMNodeMatrixSubmatrixAt Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixSubmatrixAt:



Collaboration diagram for OSnLMNodeMatrixSubmatrixAt:



Public Member Functions

- [OSnLMNodeMatrixSubmatrixAt](#) ()
default constructor.
- [~OSnLMNodeMatrixSubmatrixAt](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.155.1 Detailed Description

Definition at line 2398 of file OSnLNode.h.

6.155.2 Constructor & Destructor Documentation

6.155.2.1 OSnLMNodeMatrixSubmatrixAt::OSnLMNodeMatrixSubmatrixAt ()

default constructor.

6.155.2.2 OSnLMNodeMatrixSubmatrixAt::~OSnLMNodeMatrixSubmatrixAt ()

default destructor.

6.155.3 Member Function Documentation

6.155.3.1 virtual std::string OSnLMNodeMatrixSubmatrixAt::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.155.3.2 virtual OSnLMNode* OSnLMNodeMatrixSubmatrixAt::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

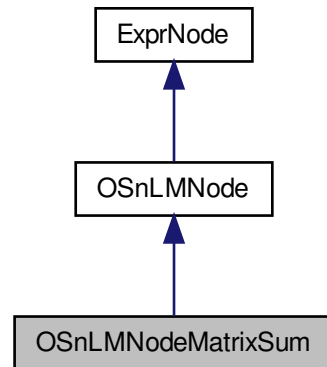
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

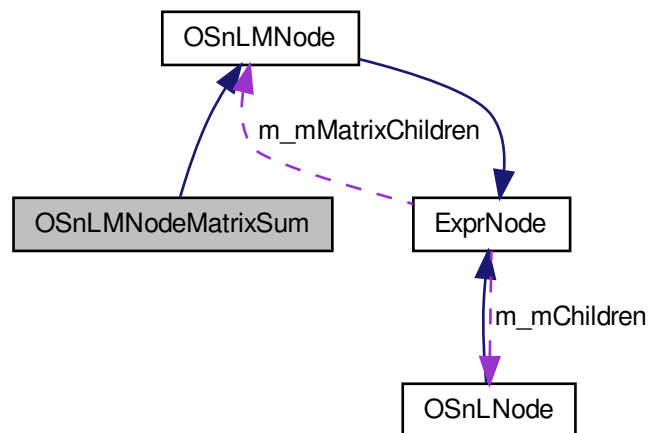
6.156 OSnLMNodeMatrixSum Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixSum:



Collaboration diagram for OSnLMNodeMatrixSum:



Public Member Functions

- [OSnLMNodeMatrixSum \(\)](#)
default constructor.
- [~OSnLMNodeMatrixSum \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.156.1 Detailed Description

Definition at line 1873 of file OSnLNode.h.

6.156.2 Constructor & Destructor Documentation

6.156.2.1 OSnLMNodeMatrixSum::OSnLMNodeMatrixSum ()

default constructor.

6.156.2.2 OSnLMNodeMatrixSum::~OSnLMNodeMatrixSum ()

default destructor.

6.156.3 Member Function Documentation

6.156.3.1 virtual std::string OSnLMNodeMatrixSum::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.156.3.2 virtual OSnLMNode* OSnLMNodeMatrixSum::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

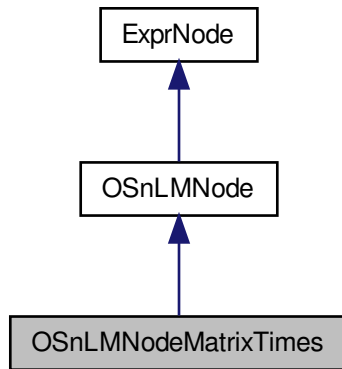
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

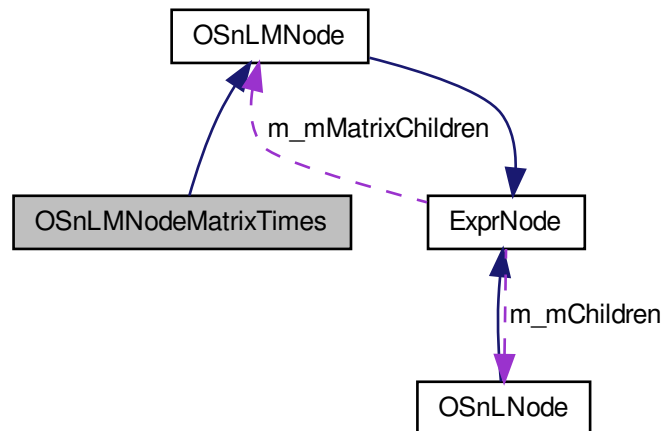
6.157 OSnLMNodeMatrixTimes Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixTimes:



Collaboration diagram for OSnLMNodeMatrixTimes:



Public Member Functions

- [OSnLMNodeMatrixTimes](#) ()
default constructor.
- [~OSnLMNodeMatrixTimes](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.157.1 Detailed Description

Definition at line 1996 of file OSnLNode.h.

6.157.2 Constructor & Destructor Documentation

6.157.2.1 OSnLMNodeMatrixTimes::OSnLMNodeMatrixTimes ()

default constructor.

6.157.2.2 OSnLMNodeMatrixTimes::~OSnLMNodeMatrixTimes ()

default destructor.

6.157.3 Member Function Documentation

6.157.3.1 virtual std::string OSnLMNodeMatrixTimes::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.157.3.2 virtual OSnLMNode* OSnLMNodeMatrixTimes::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

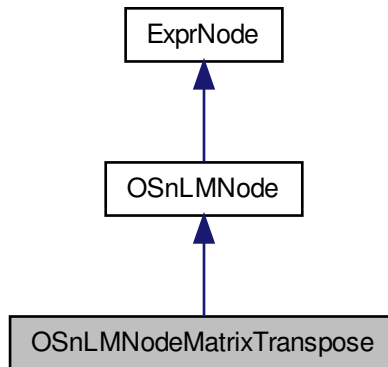
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

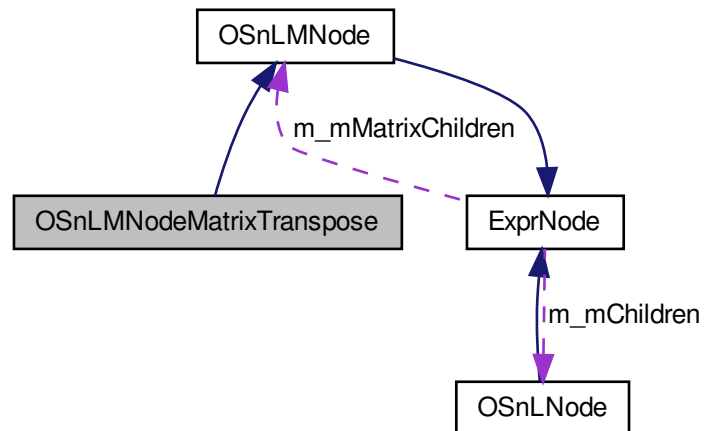
6.158 OSnLMNodeMatrixTranspose Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixTranspose:



Collaboration diagram for OSnLMNodeMatrixTranspose:



Public Member Functions

- [OSnLMNodeMatrixTranspose\(\)](#)
default constructor.
- [~OSnLMNodeMatrixTranspose\(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()

Create or clone a node of this type.

Additional Inherited Members

6.158.1 Detailed Description

Definition at line 2072 of file OSnLNode.h.

6.158.2 Constructor & Destructor Documentation

6.158.2.1 OSnLMNodeMatrixTranspose::OSnLMNodeMatrixTranspose ()

default constructor.

6.158.2.2 OSnLMNodeMatrixTranspose::~~OSnLMNodeMatrixTranspose ()

default destructor.

6.158.3 Member Function Documentation

6.158.3.1 virtual std::string OSnLMNodeMatrixTranspose::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.158.3.2 virtual OSnLMNode* OSnLMNodeMatrixTranspose::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

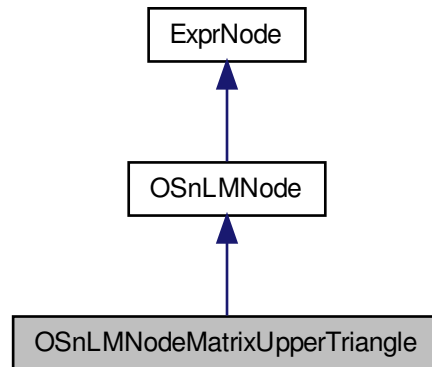
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

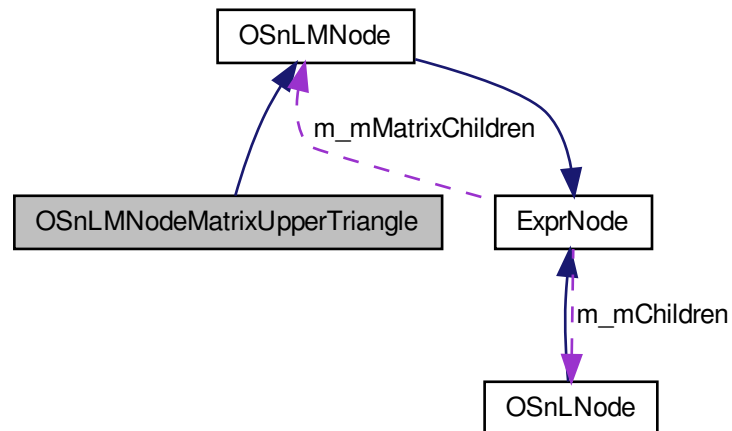
6.159 OSnLMNodeMatrixUpperTriangle Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixUpperTriangle:



Collaboration diagram for OSnLMNodeMatrixUpperTriangle:



Public Member Functions

- [OSnLMNodeMatrixUpperTriangle \(\)](#)
default constructor.
- [~OSnLMNodeMatrixUpperTriangle \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual [OSnLMNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.
- virtual bool [isEqual](#) ([OSnLMNodeMatrixUpperTriangle](#) *that)
A function to check for the equality of two objects.

Public Attributes

- bool [includeDiagonal](#)
A boolean to express whether the diagonal is to be part of the upper triangle or not.

6.159.1 Detailed Description

Definition at line 2272 of file OSnLMNode.h.

6.159.2 Constructor & Destructor Documentation

6.159.2.1 OSnLMNodeMatrixUpperTriangle::OSnLMNodeMatrixUpperTriangle ()

default constructor.

6.159.2.2 OSnLMNodeMatrixUpperTriangle::~~OSnLMNodeMatrixUpperTriangle ()

default destructor.

6.159.3 Member Function Documentation

6.159.3.1 virtual std::string OSnLMNodeMatrixUpperTriangle::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.159.3.2 virtual std::string OSnLMNodeMatrixUpperTriangle::getNonlinearExpressionInXML () [virtual]

Returns

a string token that corresponds to the [OSnLMNode](#).
the OSiL XML for the [OSnLMNode](#) <matrix>.

Reimplemented from [ExprNode](#).

6.159.3.3 virtual OSnLMNode* OSnLMNodeMatrixUpperTriangle::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.159.3.4 virtual bool OSnLMNodeMatrixUpperTriangle::isEqual (OSnLMNodeMatrixUpperTriangle * *that*) [virtual]

A function to check for the equality of two objects.

6.159.4 Member Data Documentation

6.159.4.1 bool OSnLMNodeMatrixUpperTriangle::includeDiagonal

A boolean to express whether the diagonal is to be part of the upper triangle or not.

Definition at line 2278 of file OSnLNode.h.

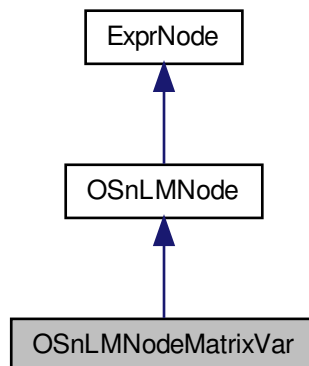
The documentation for this class was generated from the following file:

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

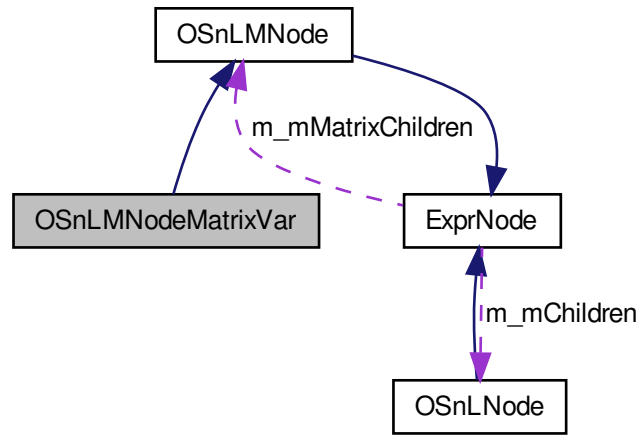
6.160 OSnLMNodeMatrixVar Class Reference

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLMNodeMatrixVar:



Collaboration diagram for OSnLMNodeMatrixVar:



Public Member Functions

- [OSnLMNodeMatrixVar \(\)](#)
default constructor.
- [~OSnLMNodeMatrixVar \(\)](#)
default destructor.
- virtual [std::string getTokenName \(\)](#)
- virtual [std::string getTokenNumber \(\)](#)
- virtual [std::string getNonlinearExpressionInXML \(\)](#)
- virtual [OSnLMNode * cloneExprNode \(\)](#)
Create or clone a node of this type.
- virtual [bool isEqual \(OSnLMNodeMatrixVar *that\)](#)
A function to check for the equality of two objects.

Public Attributes

- [int idx](#)
The index of the matrixVar.

6.160.1 Detailed Description

Definition at line 2502 of file OSnLNode.h.

6.160.2 Constructor & Destructor Documentation

6.160.2.1 OSnLMNodeMatrixVar::OSnLMNodeMatrixVar ()

default constructor.

6.160.2.2 OSnLMNodeMatrixVar::~~OSnLMNodeMatrixVar ()

default destructor.

6.160.3 Member Function Documentation

6.160.3.1 virtual std::string OSnLMNodeMatrixVar::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.160.3.2 virtual std::string OSnLMNodeMatrixVar::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.160.3.3 virtual std::string OSnLMNodeMatrixVar::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the [OSnLMNode](#) <matrixReference>.

Reimplemented from [ExprNode](#).

6.160.3.4 virtual OSnLMNode* OSnLMNodeMatrixVar::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

6.160.3.5 virtual bool OSnLMNodeMatrixVar::isEqual (OSnLMNodeMatrixVar * *that*) [virtual]

A function to check for the equality of two objects.

6.160.4 Member Data Documentation

6.160.4.1 int OSnLMNodeMatrixVar::idx

The index of the matrixVar.

Definition at line 2508 of file OSnLNode.h.

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

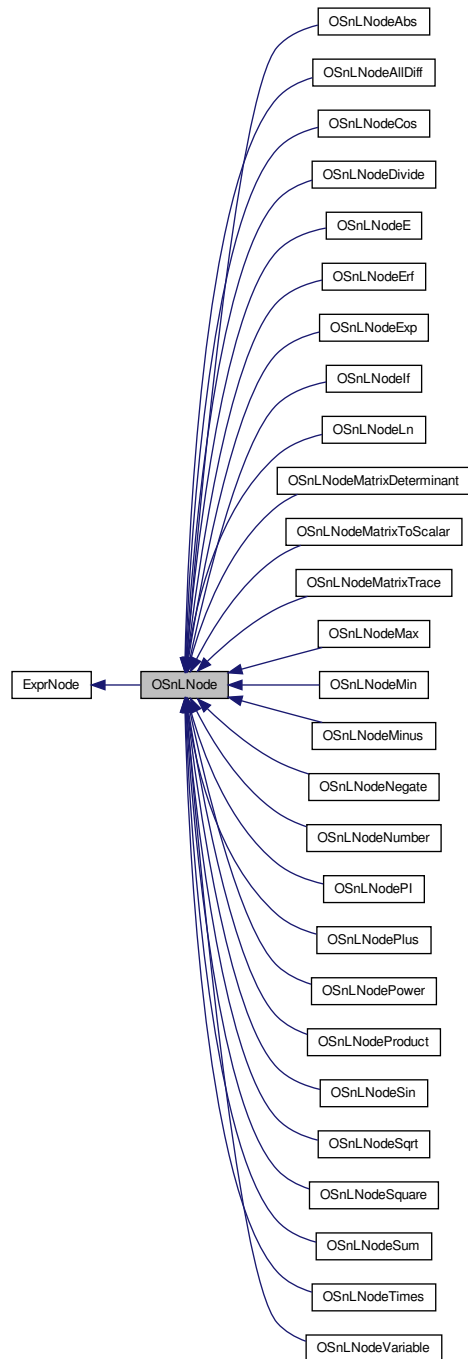
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.161 OSnLNode Class Reference

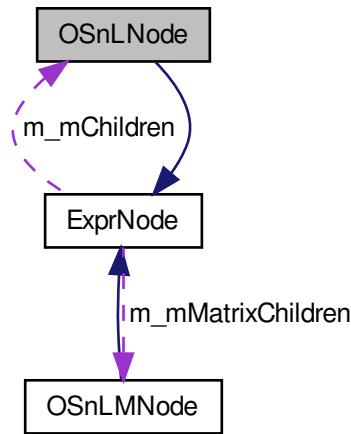
The [OSnLNode](#) Class for nonlinear expressions.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNode:



Collaboration diagram for OSnLNode:



Public Member Functions

- [OSnLNode](#) ()
default constructor.
- virtual [~OSnLNode](#) ()
default destructor.
- virtual void [getVariableIndexMap](#) (std::map< int, int > *varIdx)
*varIdx is a map where the key is the index of an [OSnLNodeVariable](#) and (*varIdx)[idx] is the kth variable in the map, e.g.*
- virtual double [calculateFunction](#) (double *x)=0
Calculate the function value given the current variable values.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)=0
Create the AD tape to be evaluated by AD.
- [OSnLNode](#) * [createExpressionTreeFromPrefix](#) (std::vector< [ExprNode](#) * > nINodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a scalar-valued [OSExpression-Tree](#) root node.
- virtual std::vector< [ExprNode](#) * > [getPrefixFromExpressionTree](#) ()
Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the (scalar-valued or matrix-valued) expression tree in prefix format.
- virtual std::vector< [ExprNode](#) * > [preOrderOSnLNodeTraversal](#) (std::vector< [ExprNode](#) * > *prefixVector)
Called by [getPrefixFromExpressionTree](#)().
- [OSnLNode](#) * [createExpressionTreeFromPostfix](#) (std::vector< [ExprNode](#) * > nINodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a scalar-valued [OSExpression-Tree](#) root node.
- virtual std::vector< [ExprNode](#) * > [getPostfixFromExpressionTree](#) ()
Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.
- virtual std::vector< [ExprNode](#) * > [postOrderOSnLNodeTraversal](#) (std::vector< [ExprNode](#) * > *postfixVector)

Called by [getPostfixFromExpressionTree\(\)](#).

- [OSnLNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- bool [isEqual](#) ([OSnLNode](#) *that)
A function to check for the equality of two objects.

Public Attributes

- double [m_dFunctionValue](#)
m_dFunctionValue holds the function value given the current variable values.
- [ADdouble](#) [m_ADtape](#)
m_ADtape stores the expression tree for the this [OSnLNode](#) as an ADdouble.

6.161.1 Detailed Description

The [OSnLNode](#) Class for nonlinear expressions.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Definition at line 179 of file [OSnLNode.h](#).

6.161.2 Constructor & Destructor Documentation

6.161.2.1 [OSnLNode::OSnLNode](#) ()

default constructor.

6.161.2.2 [virtual OSnLNode::~~OSnLNode](#) () [\[virtual\]](#)

default destructor.

6.161.3 Member Function Documentation

6.161.3.1 [virtual void OSnLNode::getVariableIndexMap](#) ([std::map](#)< int, int > * [varIdx](#)) [\[virtual\]](#)

[varIdx](#) is a map where the key is the index of an [OSnLNodeVariable](#) and ([*varIdx](#))[[idx](#)] is the kth variable in the map, e.g.

([*varIdx](#))[5] = 2 means that variable indexed by 5 is the second variable in the [OSnLNode](#) and all of its children

Parameters

<i>a</i>	pointer to a map of the variables in the OSnLNode and its children
----------	--

Reimplemented in [OSnLNodeVariable](#).

6.161.3.2 virtual double OSnLNode::calculateFunction (double * *x*) [pure virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implemented in [OSnLNodeMatrixToScalar](#), [OSnLNodeMatrixTrace](#), [OSnLNodeMatrixDeterminant](#), [OSnLNodeAllDiff](#), [OSnLNodeVariable](#), [OSnLNodePI](#), [OSnLNodeE](#), [OSnLNodeNumber](#), [OSnLNodeIf](#), [OSnLNodeErf](#), [OSnLNodeAbs](#), [OSnLNodeExp](#), [OSnLNodeSin](#), [OSnLNodeCos](#), [OSnLNodeSquare](#), [OSnLNodeSqrt](#), [OSnLNodeLn](#), [OSnLNodeProduct](#), [OSnLNodePower](#), [OSnLNodeDivide](#), [OSnLNodeTimes](#), [OSnLNodeNegate](#), [OSnLNodeMinus](#), [OSnLNodeMin](#), [OSnLNodeMax](#), [OSnLNodeSum](#), and [OSnLNodePlus](#).

6.161.3.3 virtual ADdouble OSnLNode::constructADTape (std::map< int, int > * *ADIdx*, ADvector * *XAD*) [pure virtual]

Create the AD tape to be evaluated by AD.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Returns

the expression tree.

Implemented in [OSnLNodeMatrixToScalar](#), [OSnLNodeMatrixTrace](#), [OSnLNodeMatrixDeterminant](#), [OSnLNodeAllDiff](#), [OSnLNodeVariable](#), [OSnLNodePI](#), [OSnLNodeE](#), [OSnLNodeNumber](#), [OSnLNodeIf](#), [OSnLNodeErf](#), [OSnLNodeAbs](#), [OSnLNodeExp](#), [OSnLNodeSin](#), [OSnLNodeCos](#), [OSnLNodeSquare](#), [OSnLNodeSqrt](#), [OSnLNodeLn](#), [OSnLNodeProduct](#), [OSnLNodePower](#), [OSnLNodeDivide](#), [OSnLNodeTimes](#), [OSnLNodeNegate](#), [OSnLNodeMinus](#), [OSnLNodeMin](#), [OSnLNodeMax](#), [OSnLNodeSum](#), and [OSnLNodePlus](#).

6.161.3.4 OSnLNode* OSnLNode::createExpressionTreeFromPrefix (std::vector< ExprNode * > *nINodeVec*)

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a scalar-valued [OSExpressionTree](#) root node.

Parameters

<i>nINodeVec</i>	holds a vector of pointers to OSnLNodes and OSnLMNodes in prefix format
------------------	---

Returns

a pointer to an [OSnLNode](#) which is the root of an [OSExpressionTree](#).

6.161.3.5 `virtual std::vector<ExprNode*> OSnLNode::getPrefixFromExpressionTree () [virtual]`

Get a vector of pointers to OSnLNodes and OSnLMNodes that correspond to the (scalar-valued or matrix-valued) expression tree in prefix format.

Returns

the expression tree as a vector of ExprNodes in prefix.

Reimplemented from [ExprNode](#).

6.161.3.6 `virtual std::vector<ExprNode*> OSnLNode::preOrderOSnLNodeTraversal (std::vector< ExprNode * > * prefixVector) [virtual]`

Called by [getPrefixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to [ExprNode](#) in prefix

Parameters

<i>a</i>	pointer prefixVector to a vector of pointers of ExprNodes
----------	---

Returns

a vector of pointers to [ExprNode](#) in prefix.

Reimplemented from [ExprNode](#).

6.161.3.7 `OSnLNode* OSnLNode::createExpressionTreeFromPostfix (std::vector< ExprNode * > nINodeVec)`

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a scalar-valued [OS-ExpressionTree](#) root node.

Parameters

<i>nINodeVec</i>	holds a vector of pointers to OSnLNodes in postfix format
------------------	---

Returns

a pointer to an [OSnLNode](#) which is the root of an [OSExpressionTree](#).

6.161.3.8 `virtual std::vector<ExprNode*> OSnLNode::getPostfixFromExpressionTree () [virtual]`

Get a vector of pointers to ExprNodes that correspond to the expression tree in postfix format.

Returns

the expression tree as a vector of ExprNodes in postfix.

Reimplemented from [ExprNode](#).

6.161.3.9 `virtual std::vector<ExprNode*> OSnLNode::postOrderOSnLNodeTraversal (std::vector< ExprNode * > * postfixVector) [virtual]`

Called by [getPostfixFromExpressionTree\(\)](#).

This method calls itself recursively and generates a vector of pointers to ExprNodes in postfix.

Parameters

<i>a</i>	pointer postfixVector to a vector of pointers of ExprNodes
----------	--

Returns

a vector of pointers to ExprNodes in postfix.

Reimplemented from [ExprNode](#).

6.161.3.10 OSnLNode* OSnLNode::copyNodeAndDescendants ()

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

6.161.3.11 bool OSnLNode::isEqual (OSnLNode * *that*)

A function to check for the equality of two objects.

6.161.4 Member Data Documentation

6.161.4.1 double OSnLNode::m_dFunctionValue

m_dFunctionValue holds the function value given the current variable values.

Definition at line 185 of file OSnLNode.h.

6.161.4.2 ADdouble OSnLNode::m_ADtape

m_ADtape stores the expression tree for the this [OSnLNode](#) as an ADdouble.

Definition at line 190 of file OSnLNode.h.

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

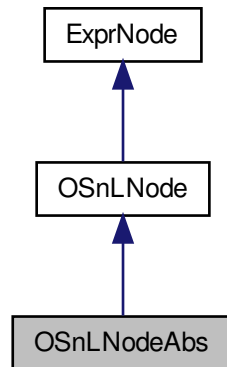
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.162 OSnLNodeAbs Class Reference

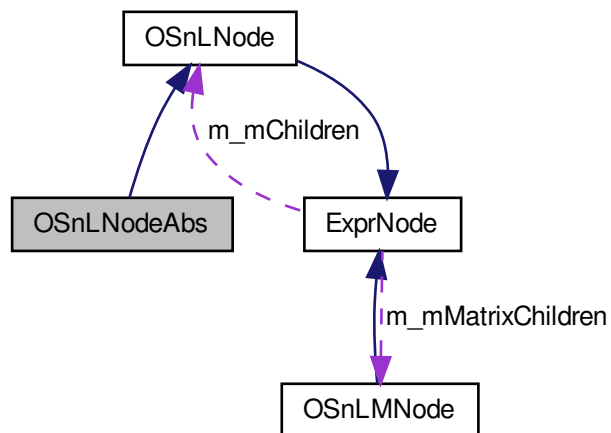
The [OSnLNodeAbs](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeAbs:



Collaboration diagram for OSnLNodeAbs:



Public Member Functions

- [OSnLNodeAbs](#) ()
default constructor.
- [~OSnLNodeAbs](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.162.1 Detailed Description

The [OSnLNodeAbs](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <abs>

Definition at line 1112 of file OSnLNode.h.

6.162.2 Constructor & Destructor Documentation

6.162.2.1 OSnLNodeAbs::OSnLNodeAbs ()

default constructor.

6.162.2.2 OSnLNodeAbs::~OSnLNodeAbs ()

default destructor.

6.162.3 Member Function Documentation

6.162.3.1 virtual std::string OSnLNodeAbs::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.162.3.2 virtual double OSnLNodeAbs::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.162.3.3 OSnLNode * OSnLNodeAbs::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.162.3.4 double OSnLNodeAbs::constructADTape (std::map< int, int > * AD/dx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

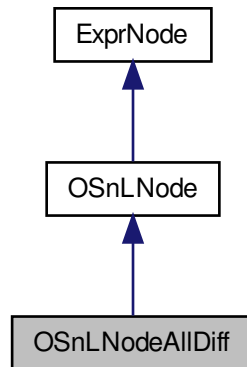
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.163 OSnLNodeAlldiff Class Reference

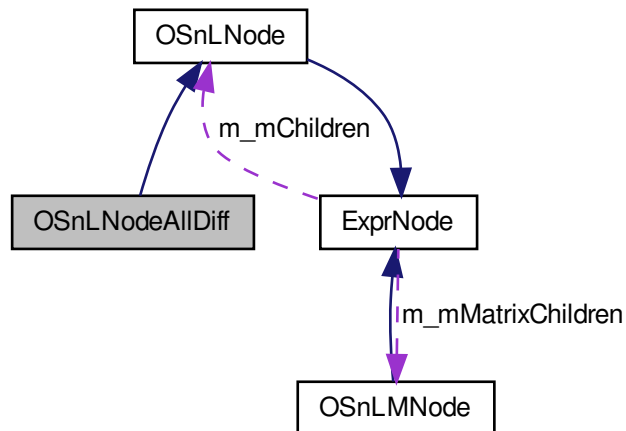
The [OSnLNodeAlldiff](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeAllDiff:



Collaboration diagram for OSnLNodeAllDiff:



Public Member Functions

- [OSnLNodeAllDiff\(\)](#)
default constructor.
- [~OSnLNodeAllDiff\(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.163.1 Detailed Description

The [OSnLNodeAlldiff](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <alldiff>

Definition at line 1549 of file OSnLNode.h.

6.163.2 Constructor & Destructor Documentation

6.163.2.1 OSnLNodeAlldiff::OSnLNodeAlldiff ()

default constructor.

6.163.2.2 OSnLNodeAlldiff::~~OSnLNodeAlldiff ()

default destructor.

6.163.3 Member Function Documentation

6.163.3.1 virtual std::string OSnLNodeAlldiff::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.163.3.2 virtual double OSnLNodeAllDiff::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.163.3.3 OSnLNode * OSnLNodeAllDiff::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.163.3.4 double OSnLNodeAllDiff::constructADTape (std::map< int, int > * *AD/dx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

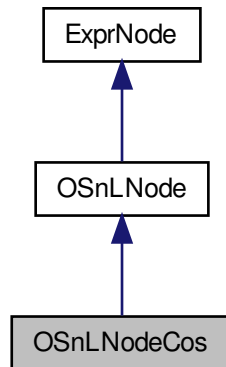
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.164 OSnLNodeCos Class Reference

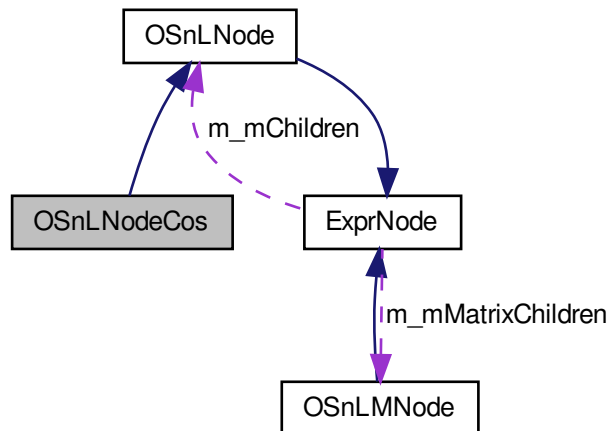
The [OSnLNodeCos](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeCos:



Collaboration diagram for OSnLNodeCos:



Public Member Functions

- [OSnLNodeCos](#) ()
default constructor.
- [~OSnLNodeCos](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.164.1 Detailed Description

The [OSnLNodeCos](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <cos>

Definition at line 962 of file OSnLNode.h.

6.164.2 Constructor & Destructor Documentation

6.164.2.1 OSnLNodeCos::OSnLNodeCos ()

default constructor.

6.164.2.2 OSnLNodeCos::~~OSnLNodeCos ()

default destructor.

6.164.3 Member Function Documentation

6.164.3.1 virtual std::string OSnLNodeCos::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.164.3.2 virtual double OSnLNodeCos::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.164.3.3 OSnLNode * OSnLNodeCos::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.164.3.4 double OSnLNodeCos::constructADTape (std::map< int, int > * AD/dx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

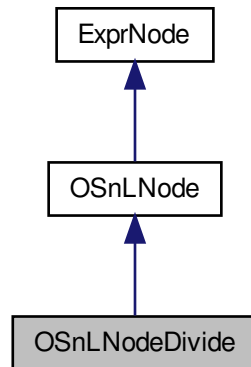
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.165 OSnLNodeDivide Class Reference

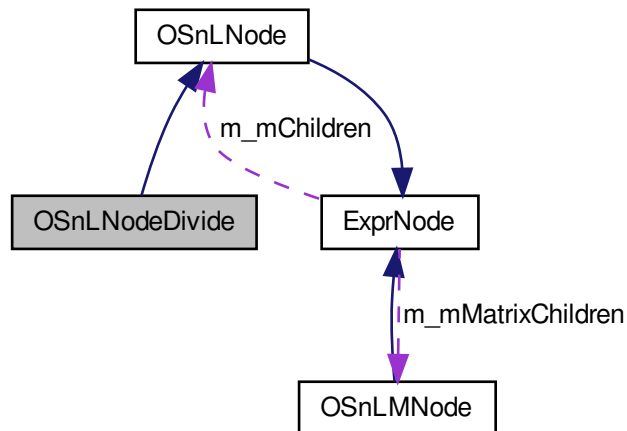
The [OSnLNodeDivide](#) Class.

```
#include <OSnLNode.h>
```


Inheritance diagram for OSnLNodeDivide:



Collaboration diagram for OSnLNodeDivide:



Public Member Functions

- [OSnLNodeDivide](#) ()
default constructor.
- [~OSnLNodeDivide](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.165.1 Detailed Description

The [OSnLNodeDivide](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <divide>

Definition at line 668 of file OSnLNode.h.

6.165.2 Constructor & Destructor Documentation

6.165.2.1 OSnLNodeDivide::OSnLNodeDivide ()

default constructor.

6.165.2.2 OSnLNodeDivide::~~OSnLNodeDivide ()

default destructor.

6.165.3 Member Function Documentation

6.165.3.1 virtual std::string OSnLNodeDivide::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.165.3.2 virtual double OSnLNodeDivide::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.165.3.3 OSnLNode * OSnLNodeDivide::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.165.3.4 double OSnLNodeDivide::constructADTape (std::map< int, int > * *AD/dx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

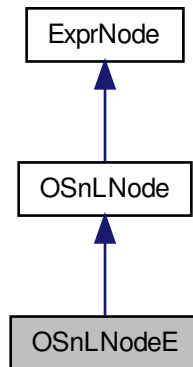
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.166 OSnLNodeE Class Reference

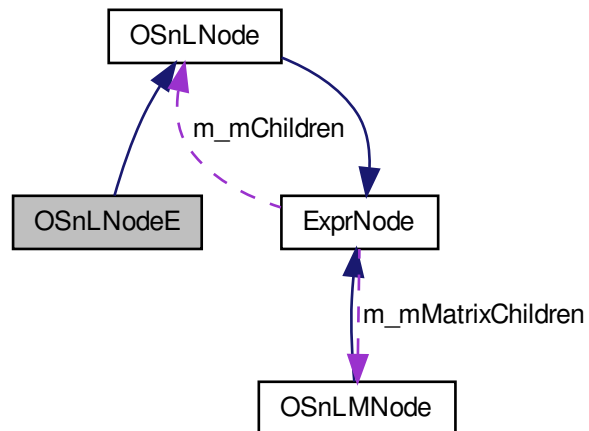
The [OSnLNodeE](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeE:



Collaboration diagram for OSnLNodeE:



Public Member Functions

- [OSnLNodeE](#) ()
default constructor.
- [~OSnLNodeE](#) ()
default destructor.

- virtual std::string [getTokenNumber](#) ()
- virtual std::string [getTokenName](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.166.1 Detailed Description

The [OSnLNodeE](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <E>

Definition at line 1341 of file OSnLNode.h.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 OSnLNodeE::OSnLNodeE ()

default constructor.

6.166.2.2 OSnLNodeE::~~OSnLNodeE ()

default destructor.

6.166.3 Member Function Documentation

6.166.3.1 virtual std::string OSnLNodeE::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.166.3.2 `virtual std::string OSnLNodeE::getTokenName () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

6.166.3.3 `virtual std::string OSnLNodeE::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

6.166.3.4 `virtual double OSnLNodeE::calculateFunction (double * x) [virtual]`

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.166.3.5 `OSnLNode * OSnLNodeE::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.166.3.6 `double OSnLNodeE::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

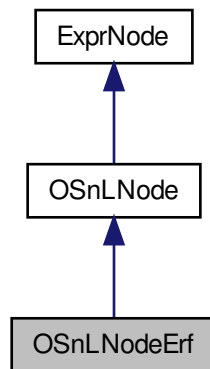
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

6.167 OSnLNodeErf Class Reference

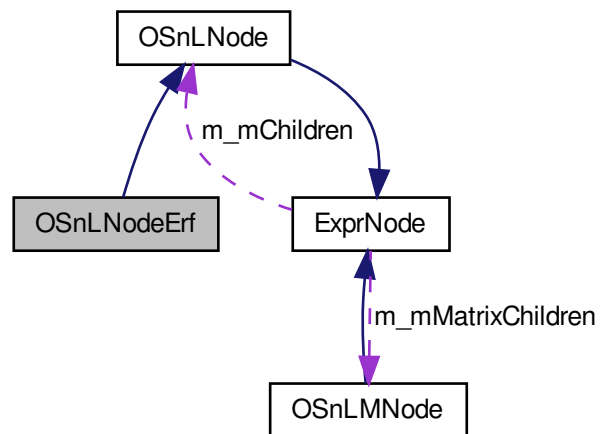
The [OSnLNodeErf](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeErf:



Collaboration diagram for OSnLNodeErf:



Public Member Functions

- [OSnLNodeErf](#) ()
default constructor.
- [~OSnLNodeErf](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
Create the AD tape to be evaluated by AD.

Additional Inherited Members**6.167.1 Detailed Description**

The [OSnLNodeErf](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <In>

Definition at line 1162 of file OSnLNode.h.

6.167.2 Constructor & Destructor Documentation**6.167.2.1 OSnLNodeErf::OSnLNodeErf ()**

default constructor.

6.167.2.2 OSnLNodeErf::~~OSnLNodeErf ()

default destructor.

6.167.3 Member Function Documentation

6.167.3.1 virtual std::string OSnLNodeErf::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.167.3.2 virtual double OSnLNodeErf::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.167.3.3 OSnLNode * OSnLNodeErf::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.167.3.4 virtual ADdouble OSnLNodeErf::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

Create the AD tape to be evaluated by AD.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Returns

the expression tree.

Implements [OSnLNode](#).

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

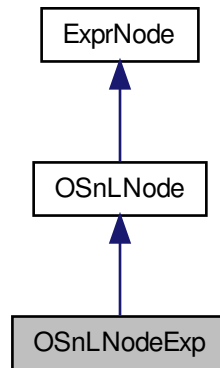
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.168 OSnLNodeExp Class Reference

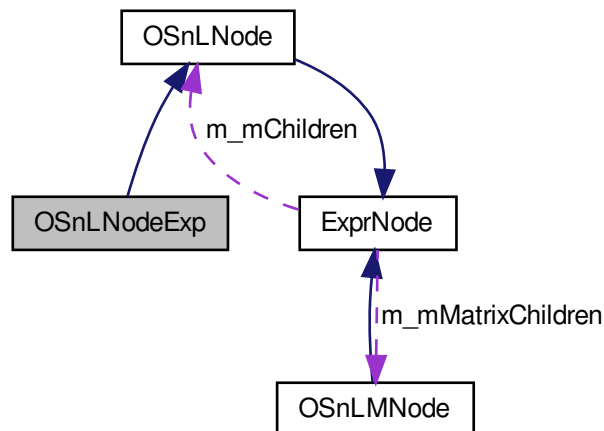
The [OSnLNodeExp](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeExp:



Collaboration diagram for OSnLNodeExp:



Public Member Functions

- [OSnLNodeExp](#) ()
default constructor.
- [~OSnLNodeExp](#) ()

default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)

Calculate the function value given the current variable values.

- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)

The implementation of the virtual functions.

- virtual [OSnLNode](#) * [cloneExprNode](#) ()

The implementation of the virtual functions.

Additional Inherited Members

6.168.1 Detailed Description

The [OSnLNodeExp](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <exp>

Definition at line 1062 of file OSnLNode.h.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 OSnLNodeExp::OSnLNodeExp ()

default constructor.

6.168.2.2 OSnLNodeExp::~OSnLNodeExp ()

default destructor.

6.168.3 Member Function Documentation

6.168.3.1 virtual std::string OSnLNodeExp::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.168.3.2 virtual double OSnLNodeExp::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.168.3.3 double OSnLNodeExp::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.168.3.4 OSnLNode * OSnLNodeExp::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

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

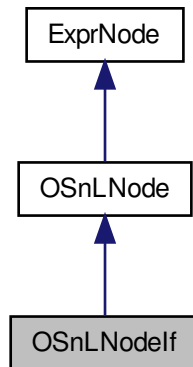
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.169 OSnLNodelf Class Reference

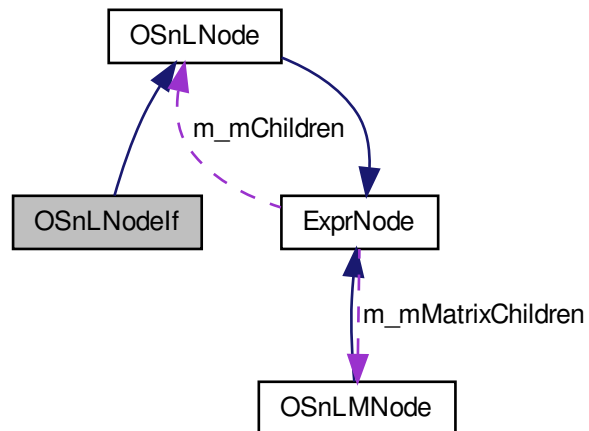
The [OSnLNodelf](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeIf:



Collaboration diagram for OSnLNodeIf:



Public Member Functions

- [OSnLNodeIf](#) ()
default constructor.
- [~OSnLNodeIf](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.169.1 Detailed Description

The [OSnLNodeIf](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <if>

Definition at line 1212 of file OSnLNode.h.

6.169.2 Constructor & Destructor Documentation

6.169.2.1 OSnLNodeIf::OSnLNodeIf ()

default constructor.

6.169.2.2 OSnLNodeIf::~~OSnLNodeIf ()

default destructor.

6.169.3 Member Function Documentation

6.169.3.1 virtual std::string OSnLNodeIf::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.169.3.2 virtual double OSnLNodeIf::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.169.3.3 OSnLNode * OSnLNodeIf::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.169.3.4 double OSnLNodeIf::constructADTape (std::map< int, int > * AD/dx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

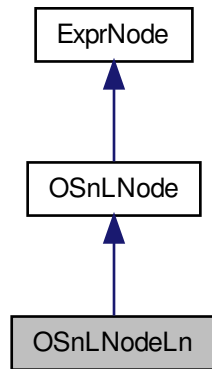
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.170 OSnLNodeLn Class Reference

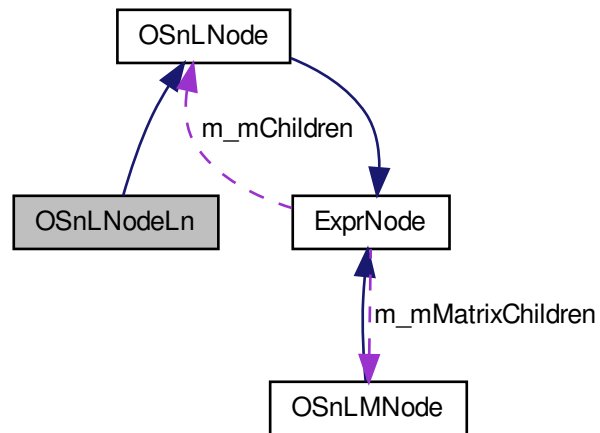
The [OSnLNodeLn](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeLn:



Collaboration diagram for OSnLNodeLn:



Public Member Functions

- [OSnLNodeLn](#) ()
default constructor.
- [~OSnLNodeLn](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.170.1 Detailed Description

The [OSnLNodeLn](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <In>

Definition at line 815 of file OSnLNode.h.

6.170.2 Constructor & Destructor Documentation

6.170.2.1 OSnLNodeLn::OSnLNodeLn ()

default constructor.

6.170.2.2 OSnLNodeLn::~~OSnLNodeLn ()

default destructor.

6.170.3 Member Function Documentation

6.170.3.1 virtual std::string OSnLNodeLn::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.170.3.2 virtual double OSnLNodeLn::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.170.3.3 OSnLNode * OSnLNodeLn::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.170.3.4 double OSnLNodeLn::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

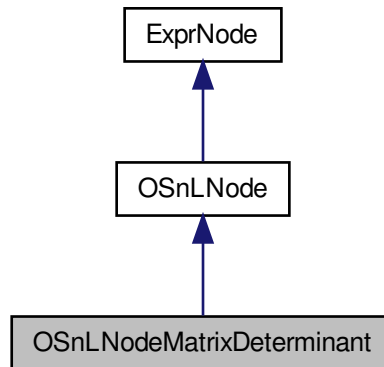
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.171 OSnLNodeMatrixDeterminant Class Reference

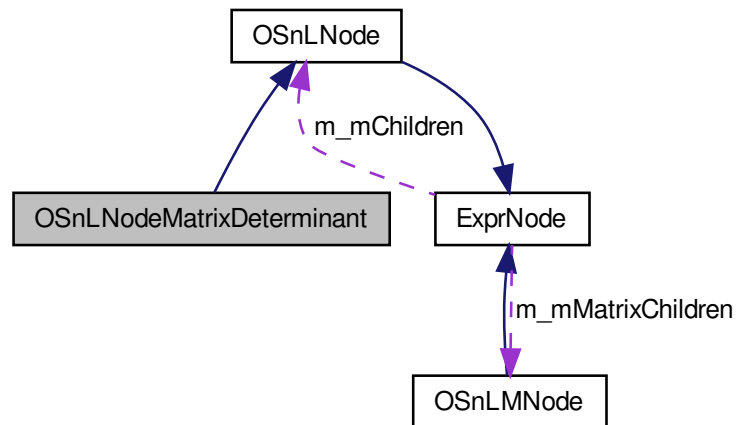
The next few nodes evaluate to a scalar even though one or more of its arguments are matrices.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeMatrixDeterminant:



Collaboration diagram for OSnLNodeMatrixDeterminant:



Public Member Functions

- [OSnLNodeMatrixDeterminant \(\)](#)
default constructor.
- [~OSnLNodeMatrixDeterminant \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.171.1 Detailed Description

The next few nodes evaluate to a scalar even though one or more of its arguments are matrices.

The [OSnLNodeMatrixDeterminant](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Date

11/06/2014

Since

OS2.8

Remarks

The in-memory representation of the OSnL element <matrixDeterminant>

Definition at line 1601 of file OSnLNode.h.

6.171.2 Constructor & Destructor Documentation

6.171.2.1 OSnLNodeMatrixDeterminant::OSnLNodeMatrixDeterminant ()

default constructor.

6.171.2.2 OSnLNodeMatrixDeterminant::~~OSnLNodeMatrixDeterminant ()

default destructor.

6.171.3 Member Function Documentation

6.171.3.1 virtual std::string OSnLNodeMatrixDeterminant::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.171.3.2 `double OSnLNodeMatrixDeterminant::calculateFunction (double * x) [virtual]`

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

6.171.3.3 `double OSnLNodeMatrixDeterminant::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.171.3.4 `virtual OSnLNode* OSnLNodeMatrixDeterminant::cloneExprNode () [virtual]`

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

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

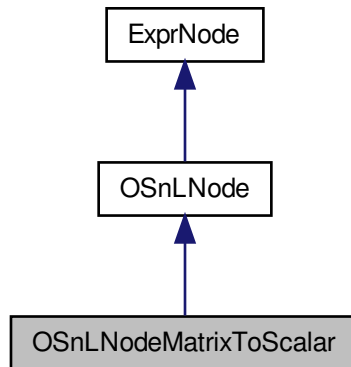
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h`

6.172 OSnLNodeMatrixToScalar Class Reference

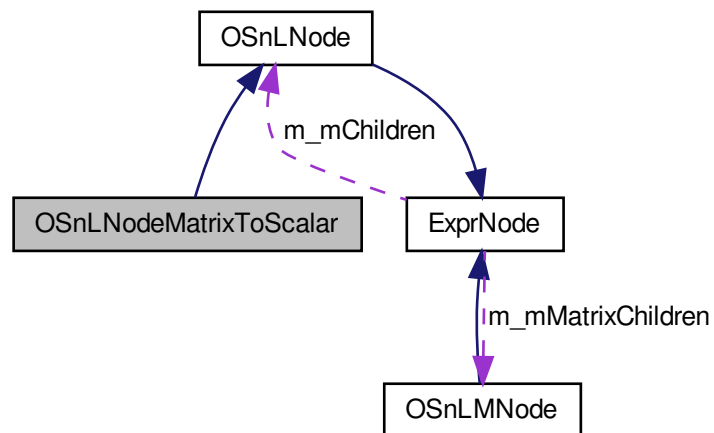
The [OSnLNodeMatrixTrace](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeMatrixToScalar:



Collaboration diagram for OSnLNodeMatrixToScalar:



Public Member Functions

- [OSnLNodeMatrixToScalar](#) ()
default constructor.
- [~OSnLNodeMatrixToScalar](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
Create the AD tape to be evaluated by AD.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.172.1 Detailed Description

The [OSnLNodeMatrixTrace](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Date

11/06/2014

Since

OS2.8

Remarks

The in-memory representation of the OSnL element <matrixToScalar>

Definition at line 1701 of file OSnLNode.h.

6.172.2 Constructor & Destructor Documentation

6.172.2.1 OSnLNodeMatrixToScalar::OSnLNodeMatrixToScalar ()

default constructor.

6.172.2.2 OSnLNodeMatrixToScalar::~~OSnLNodeMatrixToScalar ()

default destructor.

6.172.3 Member Function Documentation

6.172.3.1 virtual std::string OSnLNodeMatrixToScalar::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.172.3.2 virtual double OSnLNodeMatrixToScalar::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.172.3.3 virtual ADdouble OSnLNodeMatrixToScalar::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

Create the AD tape to be evaluated by AD.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Returns

the expression tree.

Implements [OSnLNode](#).

6.172.3.4 virtual OSnLNode* OSnLNodeMatrixToScalar::cloneExprNode () [virtual]

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

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

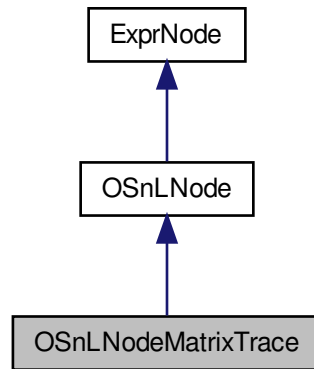
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.173 OSnLNodeMatrixTrace Class Reference

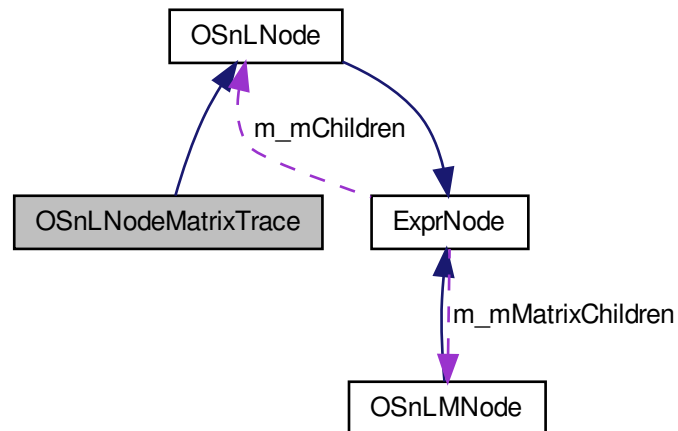
The [OSnLNodeMatrixTrace](#) Class.

```
#include <OSnLNode.h>
```


Inheritance diagram for OSnLNodeMatrixTrace:



Collaboration diagram for OSnLNodeMatrixTrace:



Public Member Functions

- [OSnLNodeMatrixTrace](#) ()
default constructor.
- [~OSnLNodeMatrixTrace](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
Create or clone a node of this type.

Additional Inherited Members

6.173.1 Detailed Description

The [OSnLNodeMatrixTrace](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Date

11/06/2014

Since

OS2.8

Remarks

The in-memory representation of the OSnL element <matrixTrace>

Definition at line 1651 of file OSnLNode.h.

6.173.2 Constructor & Destructor Documentation

6.173.2.1 OSnLNodeMatrixTrace::OSnLNodeMatrixTrace ()

default constructor.

6.173.2.2 OSnLNodeMatrixTrace::~~OSnLNodeMatrixTrace ()

default destructor.

6.173.3 Member Function Documentation

6.173.3.1 virtual std::string OSnLNodeMatrixTrace::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.173.3.2 `double OSnLNodeMatrixTrace::calculateFunction (double * x) [virtual]`

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

6.173.3.3 `double OSnLNodeMatrixTrace::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.173.3.4 `virtual OSnLNode* OSnLNodeMatrixTrace::cloneExprNode () [virtual]`

Create or clone a node of this type.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [ExprNode](#).

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

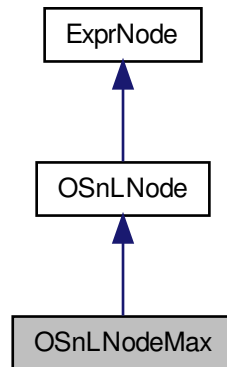
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.174 OSnLNodeMax Class Reference

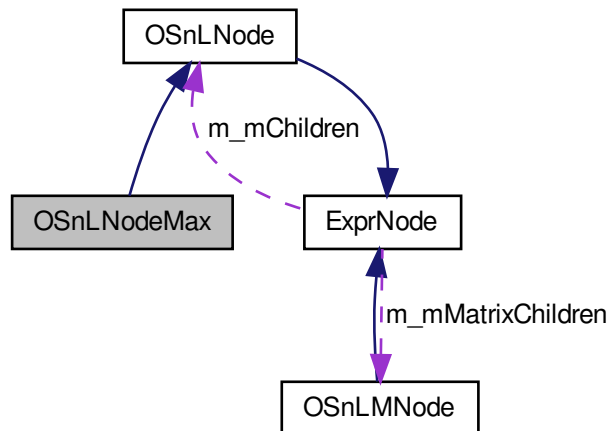
The [OSnLNodeMax](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeMax:



Collaboration diagram for OSnLNodeMax:



Public Member Functions

- [OSnLNodeMax](#) ()
default constructor.
- [~OSnLNodeMax](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.174.1 Detailed Description

The [OSnLNodeMax](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <max>

Definition at line 414 of file OSnLNode.h.

6.174.2 Constructor & Destructor Documentation

6.174.2.1 OSnLNodeMax::OSnLNodeMax ()

default constructor.

6.174.2.2 OSnLNodeMax::~OSnLNodeMax ()

default destructor.

6.174.3 Member Function Documentation

6.174.3.1 virtual std::string OSnLNodeMax::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.174.3.2 virtual double OSnLNodeMax::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.174.3.3 OSnLNode * OSnLNodeMax::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.174.3.4 double OSnLNodeMax::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

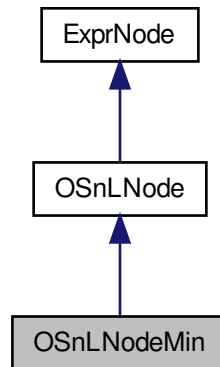
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.175 OSnLNodeMin Class Reference

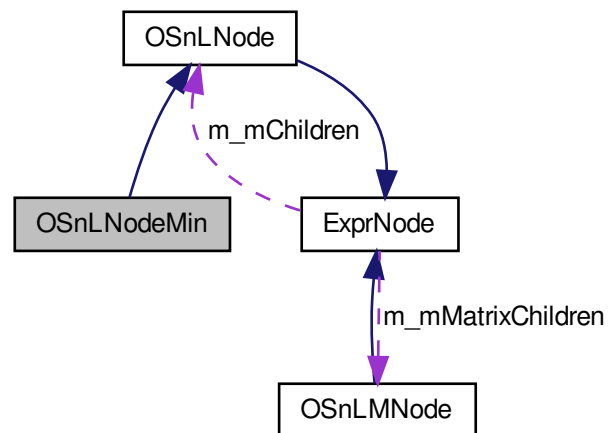
The [OSnLNodeMin](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeMin:



Collaboration diagram for OSnLNodeMin:



Public Member Functions

- [OSnLNodeMin](#) ()
default constructor.
- [~OSnLNodeMin](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.175.1 Detailed Description

The [OSnLNodeMin](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <min>

Definition at line 463 of file OSnLNode.h.

6.175.2 Constructor & Destructor Documentation

6.175.2.1 OSnLNodeMin::OSnLNodeMin ()

default constructor.

6.175.2.2 OSnLNodeMin::~OSnLNodeMin ()

default destructor.

6.175.3 Member Function Documentation

6.175.3.1 virtual std::string OSnLNodeMin::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.175.3.2 virtual double OSnLNodeMin::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.175.3.3 OSnLNode * OSnLNodeMin::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.175.3.4 double OSnLNodeMin::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

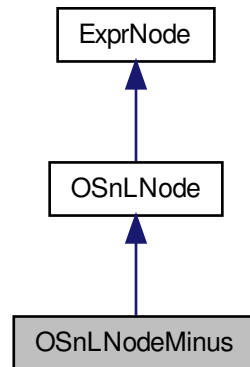
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.176 OSnLNodeMinus Class Reference

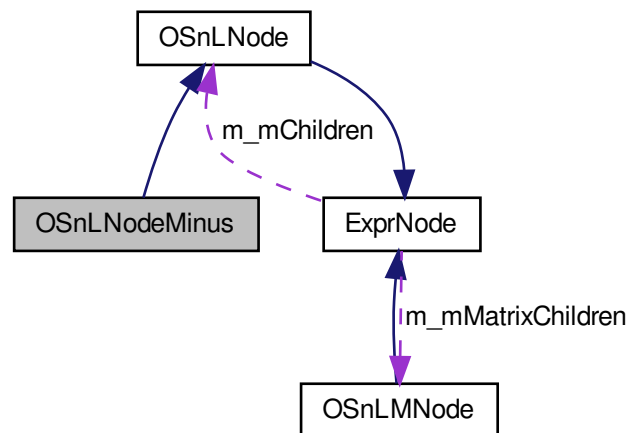
The [OSnLNodeMinus](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeMinus:



Collaboration diagram for OSnLNodeMinus:



Public Member Functions

- [OSnLNodeMinus](#) ()
default constructor.
- [~OSnLNodeMinus](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.176.1 Detailed Description

The [OSnLNodeMinus](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <minus>

Definition at line 515 of file OSnLNode.h.

6.176.2 Constructor & Destructor Documentation

6.176.2.1 OSnLNodeMinus::OSnLNodeMinus ()

default constructor.

6.176.2.2 OSnLNodeMinus::~OSnLNodeMinus ()

default destructor.

6.176.3 Member Function Documentation

6.176.3.1 virtual std::string OSnLNodeMinus::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.176.3.2 virtual double OSnLNodeMinus::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.176.3.3 OSnLNode * OSnLNodeMinus::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.176.3.4 double OSnLNodeMinus::constructADTape (std::map< int, int > * *AD/dx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

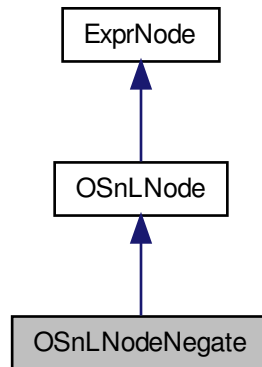
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.177 OSnLNodeNegate Class Reference

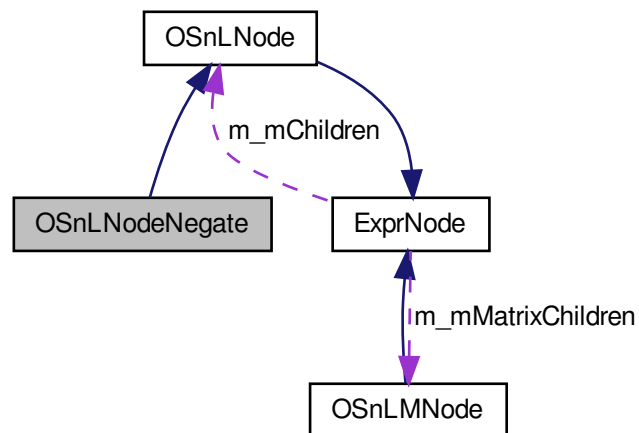
The [OSnLNodeNegate](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeNegate:



Collaboration diagram for OSnLNodeNegate:



Public Member Functions

- [OSnLNodeNegate \(\)](#)
default constructor.
- [~OSnLNodeNegate \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.177.1 Detailed Description

The [OSnLNodeNegate](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <negate>

Definition at line 566 of file OSnLNode.h.

6.177.2 Constructor & Destructor Documentation

6.177.2.1 OSnLNodeNegate::OSnLNodeNegate ()

default constructor.

6.177.2.2 OSnLNodeNegate::~OSnLNodeNegate ()

default destructor.

6.177.3 Member Function Documentation

6.177.3.1 virtual std::string OSnLNodeNegate::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.177.3.2 virtual double OSnLNodeNegate::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.177.3.3 OSnLNode * OSnLNodeNegate::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.177.3.4 double OSnLNodeNegate::constructADTape (std::map< int, int > * *ADIdx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

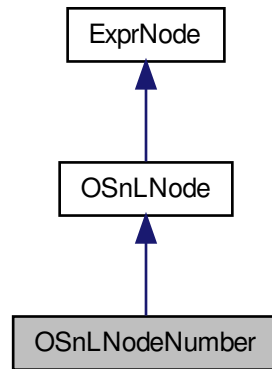
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.178 OSnLNodeNumber Class Reference

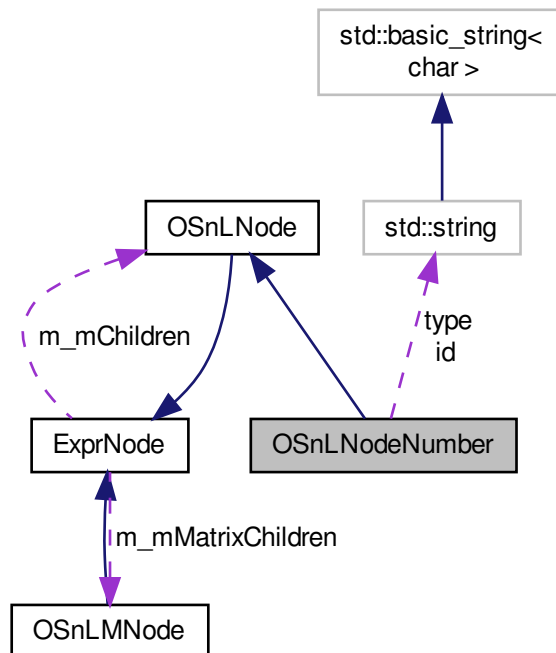
The [OSnLNodeNumber](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeNumber:



Collaboration diagram for OSnLNodeNumber:



Public Member Functions

- [OSnLNodeNumber](#) ()
default constructor.
- [~OSnLNodeNumber](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual std::string [getTokenNumber](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.
- virtual bool [isEqual](#) ([OSnLNodeNumber](#) *that)
A function to check for the equality of two objects.

Public Attributes

- double [value](#)
value is the value of the number
- std::string [type](#)
in the C++ type is real
- std::string [id](#)
later, e.g.

6.178.1 Detailed Description

The [OSnLNodeNumber](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <number>

Definition at line 1262 of file OSnLNode.h.

6.178.2 Constructor & Destructor Documentation

6.178.2.1 OSnLNodeNumber::OSnLNodeNumber ()

default constructor.

6.178.2.2 OSnLNodeNumber::~~OSnLNodeNumber ()

default destructor.

6.178.3 Member Function Documentation

6.178.3.1 virtual std::string OSnLNodeNumber::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.178.3.2 virtual std::string OSnLNodeNumber::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.178.3.3 virtual std::string OSnLNodeNumber::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

6.178.3.4 virtual double OSnLNodeNumber::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.178.3.5 OSnLNode * OSnLNodeNumber::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.178.3.6 `double OSnLNodeNumber::constructADTape (std::map< int, int > * ADidx, ADvector * XAD)` `[virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.178.3.7 `virtual bool OSnLNodeNumber::isEqual (OSnLNodeNumber * that)` `[virtual]`

A function to check for the equality of two objects.

6.178.4 Member Data Documentation

6.178.4.1 `double OSnLNodeNumber::value`

value is the value of the number

Definition at line 1266 of file OSnLNode.h.

6.178.4.2 `std::string OSnLNodeNumber::type`

in the C++ type is real

Definition at line 1269 of file OSnLNode.h.

6.178.4.3 `std::string OSnLNodeNumber::id`

later, e.g.

stochastic programming, we may wish to give an id to a number

Definition at line 1274 of file OSnLNode.h.

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

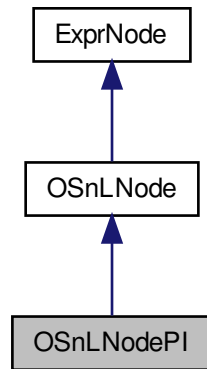
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.179 OSnLNodePI Class Reference

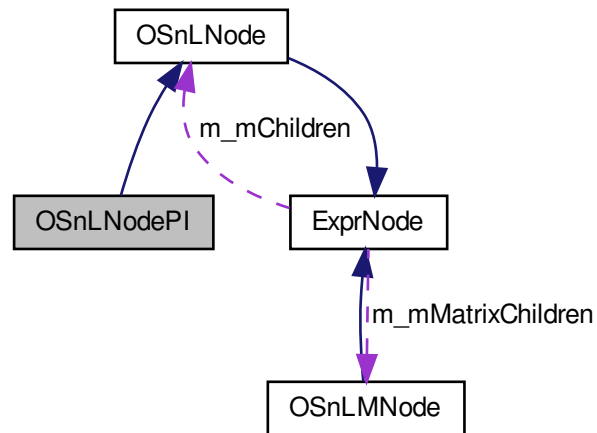
The [OSnLNodePI](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodePI:



Collaboration diagram for OSnLNodePI:



Public Member Functions

- [OSnLNodePI](#) ()
default constructor.
- [~OSnLNodePI](#) ()
default destructor.

- virtual std::string [getTokenNumber](#) ()
- virtual std::string [getTokenName](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.179.1 Detailed Description

The [OSnLNodePI](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <pi>

Definition at line 1407 of file OSnLNode.h.

6.179.2 Constructor & Destructor Documentation

6.179.2.1 [OSnLNodePI::OSnLNodePI](#) ()

default constructor.

6.179.2.2 [OSnLNodePI::~~OSnLNodePI](#) ()

default destructor.

6.179.3 Member Function Documentation

6.179.3.1 virtual std::string [OSnLNodePI::getTokenNumber](#) () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.179.3.2 `virtual std::string OSnLNodePI::getTokenName () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

6.179.3.3 `virtual std::string OSnLNodePI::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

6.179.3.4 `virtual double OSnLNodePI::calculateFunction (double * x) [virtual]`

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.179.3.5 `OSnLNode * OSnLNodePI::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.179.3.6 `double OSnLNodePI::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

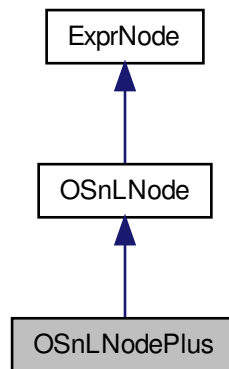
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.180 OSnLNodePlus Class Reference

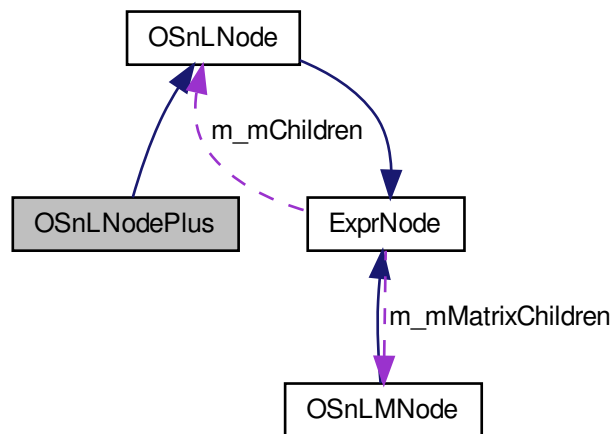
The [OSnLNodePlus](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodePlus:



Collaboration diagram for OSnLNodePlus:



Public Member Functions

- [OSnLNodePlus](#) ()
default constructor.
- [~OSnLNodePlus](#) ()
default destructor.
- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.

Additional Inherited Members**6.180.1 Detailed Description**

The [OSnLNodePlus](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <plus>

Definition at line 315 of file OSnLNode.h.

6.180.2 Constructor & Destructor Documentation**6.180.2.1 OSnLNodePlus::OSnLNodePlus ()**

default constructor.

6.180.2.2 OSnLNodePlus::~~OSnLNodePlus ()

default destructor.

6.180.3 Member Function Documentation

6.180.3.1 `virtual std::string OSnLNodePlus::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

6.180.3.2 `double OSnLNodePlus::calculateFunction (double * x) [virtual]`

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

6.180.3.3 `double OSnLNodePlus::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.180.3.4 `OSnLNode * OSnLNodePlus::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

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

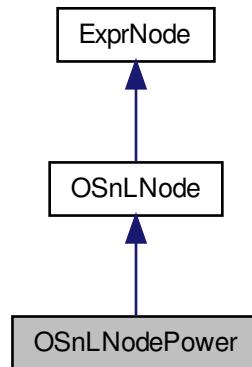
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.181 OSnLNodePower Class Reference

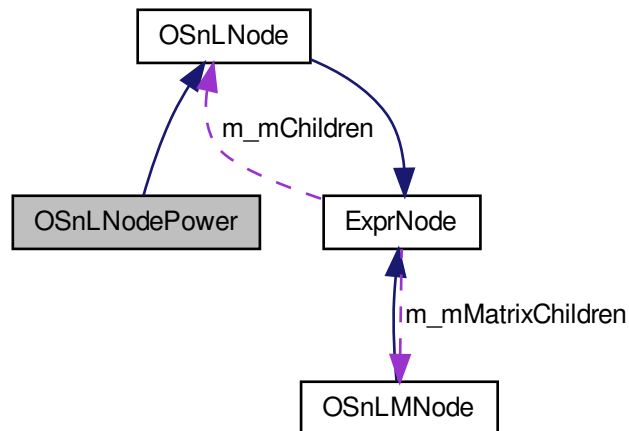
The [OSnLNodePower](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodePower:



Collaboration diagram for OSnLNodePower:



Public Member Functions

- [OSnLNodePower](#) ()
default constructor.
- [~OSnLNodePower](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.181.1 Detailed Description

The [OSnLNodePower](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <power>

Definition at line 717 of file OSnLNode.h.

6.181.2 Constructor & Destructor Documentation

6.181.2.1 OSnLNodePower::OSnLNodePower ()

default constructor.

6.181.2.2 OSnLNodePower::~~OSnLNodePower ()

default destructor.

6.181.3 Member Function Documentation

6.181.3.1 virtual std::string OSnLNodePower::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.181.3.2 virtual double OSnLNodePower::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.181.3.3 OSnLNode * OSnLNodePower::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.181.3.4 double OSnLNodePower::constructADTape (std::map< int, int > * *AD/dx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

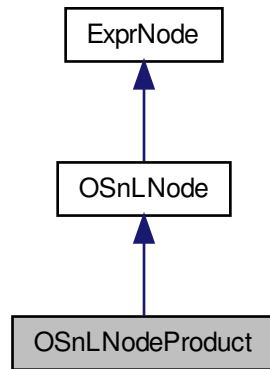
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.182 OSnLNodeProduct Class Reference

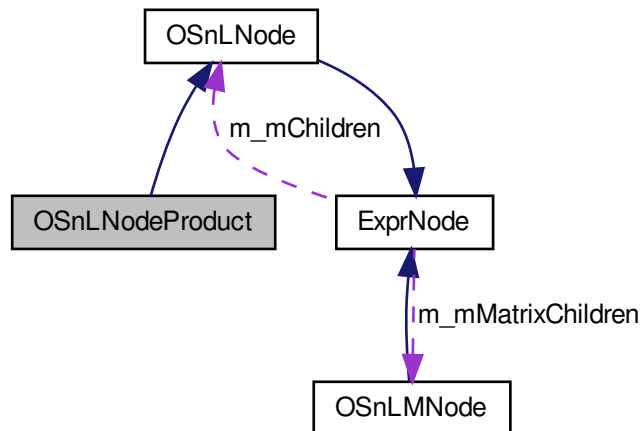
The [OSnLNodeProduct](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeProduct:



Collaboration diagram for OSnLNodeProduct:



Public Member Functions

- [OSnLNodeProduct](#) ()
default constructor.
- [~OSnLNodeProduct](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.182.1 Detailed Description

The [OSnLNodeProduct](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <product>

Definition at line 766 of file OSnLNode.h.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 OSnLNodeProduct::OSnLNodeProduct ()

default constructor.

6.182.2.2 OSnLNodeProduct::~~OSnLNodeProduct ()

default destructor.

6.182.3 Member Function Documentation

6.182.3.1 virtual std::string OSnLNodeProduct::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.182.3.2 virtual double OSnLNodeProduct::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.182.3.3 OSnLNode * OSnLNodeProduct::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.182.3.4 double OSnLNodeProduct::constructADTape (std::map< int, int > * *ADId*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

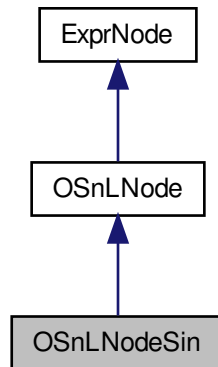
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.183 OSnLNodeSin Class Reference

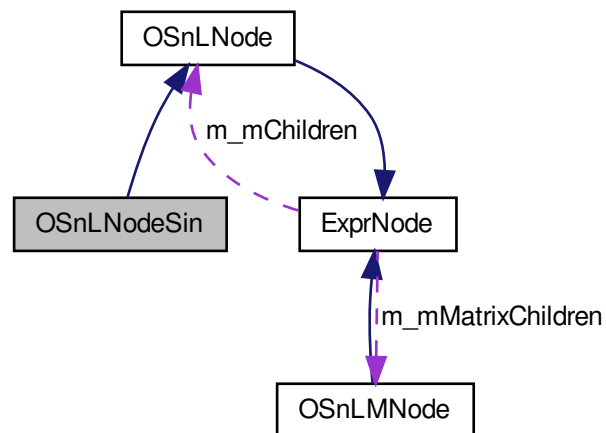
The [OSnLNodeSin](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeSin:



Collaboration diagram for OSnLNodeSin:



Public Member Functions

- [OSnLNodeSin](#) ()
default constructor.
- [~OSnLNodeSin](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.183.1 Detailed Description

The [OSnLNodeSin](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <sin>

Definition at line 1012 of file OSnLNode.h.

6.183.2 Constructor & Destructor Documentation

6.183.2.1 OSnLNodeSin::OSnLNodeSin ()

default constructor.

6.183.2.2 OSnLNodeSin::~~OSnLNodeSin ()

default destructor.

6.183.3 Member Function Documentation

6.183.3.1 virtual std::string OSnLNodeSin::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.183.3.2 virtual double OSnLNodeSin::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.183.3.3 OSnLNode * OSnLNodeSin::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.183.3.4 double OSnLNodeSin::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

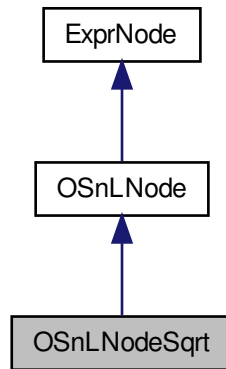
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.184 OSnLNodeSqrt Class Reference

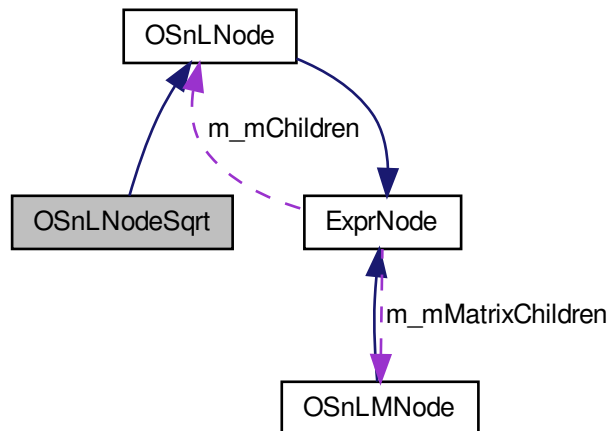
The [OSnLNodeSqrt](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeSqrt:



Collaboration diagram for OSnLNodeSqrt:



Public Member Functions

- [OSnLNodeSqrt \(\)](#)
default constructor.
- [~OSnLNodeSqrt \(\)](#)
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.184.1 Detailed Description

The [OSnLNodeSqrt](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <sqrt>

Definition at line 864 of file OSnLNode.h.

6.184.2 Constructor & Destructor Documentation

6.184.2.1 [OSnLNodeSqrt::OSnLNodeSqrt](#) ()

default constructor.

6.184.2.2 [OSnLNodeSqrt::~~OSnLNodeSqrt](#) ()

default destructor.

6.184.3 Member Function Documentation

6.184.3.1 virtual std::string [OSnLNodeSqrt::getTokenName](#) () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.184.3.2 virtual double OSnLNodeSqrt::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.184.3.3 OSnLNode * OSnLNodeSqrt::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.184.3.4 double OSnLNodeSqrt::constructADTape (std::map< int, int > * AD/dx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

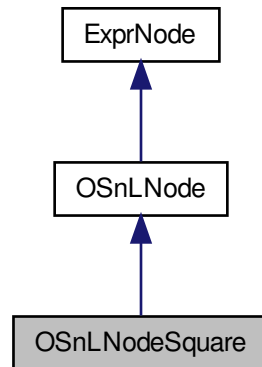
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.185 OSnLNodeSquare Class Reference

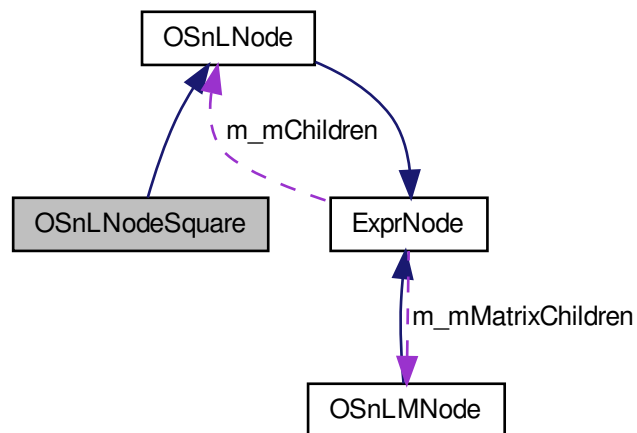
The [OSnLNodeSquare](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeSquare:



Collaboration diagram for OSnLNodeSquare:



Public Member Functions

- [OSnLNodeSquare](#) ()
default constructor.
- [~OSnLNodeSquare](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.185.1 Detailed Description

The [OSnLNodeSquare](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <square>

Definition at line 912 of file OSnLNode.h.

6.185.2 Constructor & Destructor Documentation

6.185.2.1 OSnLNodeSquare::OSnLNodeSquare ()

default constructor.

6.185.2.2 OSnLNodeSquare::~OSnLNodeSquare ()

default destructor.

6.185.3 Member Function Documentation

6.185.3.1 virtual std::string OSnLNodeSquare::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.185.3.2 virtual double OSnLNodeSquare::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.185.3.3 OSnLNode * OSnLNodeSquare::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.185.3.4 double OSnLNodeSquare::constructADTape (std::map< int, int > * *ADIdx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

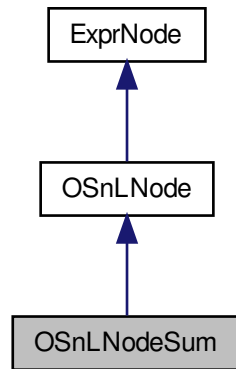
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.186 OSnLNodeSum Class Reference

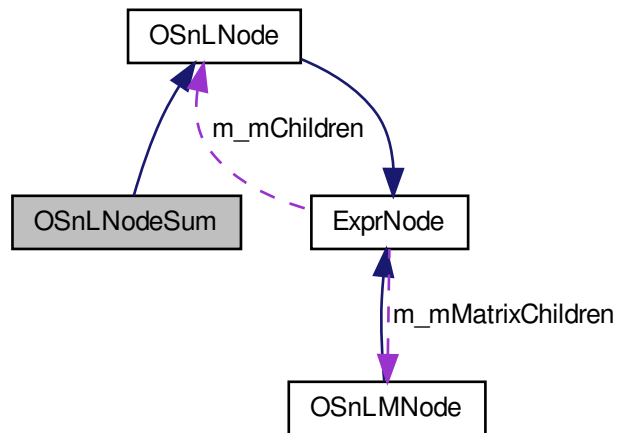
The [OSnLNodeSum](#) Class.

```
#include <OSnLNode.h>
```


Inheritance diagram for OSnLNodeSum:



Collaboration diagram for OSnLNodeSum:



Public Member Functions

- [OSnLNodeSum](#) ()
default constructor.
- [~OSnLNodeSum](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.

Additional Inherited Members

6.186.1 Detailed Description

The [OSnLNodeSum](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <sum>

Definition at line 365 of file OSnLNode.h.

6.186.2 Constructor & Destructor Documentation

6.186.2.1 OSnLNodeSum::OSnLNodeSum ()

default constructor.

6.186.2.2 OSnLNodeSum::~~OSnLNodeSum ()

default destructor.

6.186.3 Member Function Documentation

6.186.3.1 virtual std::string OSnLNodeSum::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.186.3.2 virtual double OSnLNodeSum::calculateFunction (double * *x*) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.186.3.3 OSnLNode * OSnLNodeSum::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.186.3.4 double OSnLNodeSum::constructADTape (std::map< int, int > * *AD/dx*, ADvector * *XAD*) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

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

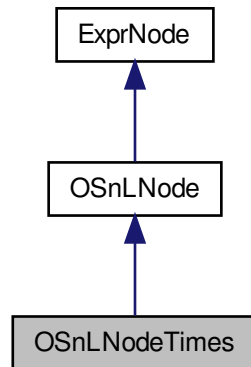
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.187 OSnLNodeTimes Class Reference

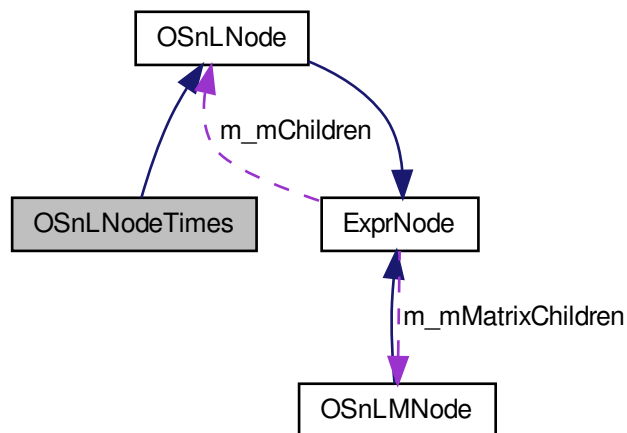
The [OSnLNodeTimes](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeTimes:



Collaboration diagram for OSnLNodeTimes:



Public Member Functions

- [OSnLNodeTimes](#) ()
default constructor.
- [~OSnLNodeTimes](#) ()
default destructor.

- virtual std::string [getTokenName](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
Create the AD tape to be evaluated by AD.

Additional Inherited Members

6.187.1 Detailed Description

The [OSnLNodeTimes](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <times>

Definition at line 617 of file OSnLNode.h.

6.187.2 Constructor & Destructor Documentation

6.187.2.1 OSnLNodeTimes::OSnLNodeTimes ()

default constructor.

6.187.2.2 OSnLNodeTimes::~OSnLNodeTimes ()

default destructor.

6.187.3 Member Function Documentation

6.187.3.1 virtual std::string OSnLNodeTimes::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

6.187.3.2 virtual double OSnLNodeTimes::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.187.3.3 OSnLNode * OSnLNodeTimes::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.187.3.4 virtual ADdouble OSnLNodeTimes::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

Create the AD tape to be evaluated by AD.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Returns

the expression tree.

Implements [OSnLNode](#).

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

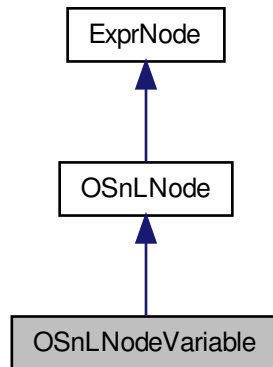
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSnLNode.h](#)

6.188 OSnLNodeVariable Class Reference

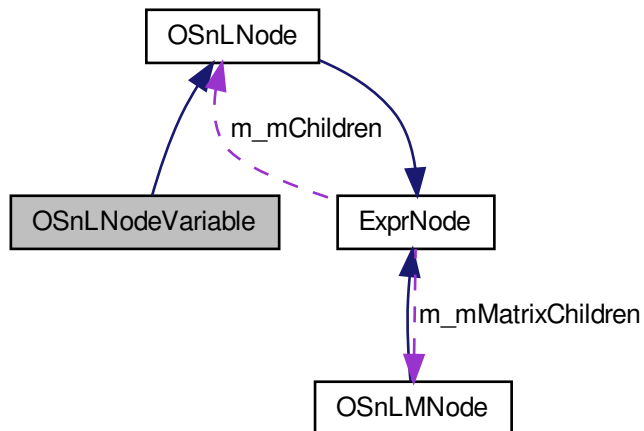
The [OSnLNodeVariable](#) Class.

```
#include <OSnLNode.h>
```

Inheritance diagram for OSnLNodeVariable:



Collaboration diagram for OSnLNodeVariable:



Public Member Functions

- [OSnLNodeVariable](#) ()
default constructor.
- [~OSnLNodeVariable](#) ()
default destructor.

- virtual void [getVariableIndexMap](#) (std::map< int, int > *varIdx)
*varIdx is a map where the key is the index of an [OSnLNodeVariable](#) and (*varIdx)[idx] is the kth variable in the map, e.g.*
- virtual std::string [getTokenNumber](#) ()
- virtual std::string [getTokenName](#) ()
- virtual std::string [getNonlinearExpressionInXML](#) ()
- virtual double [calculateFunction](#) (double *x)
Calculate the function value given the current variable values.
- virtual [OSnLNode](#) * [cloneExprNode](#) ()
The implementation of the virtual functions.
- virtual [ADdouble](#) [constructADTape](#) (std::map< int, int > *ADIdx, [ADvector](#) *XAD)
The implementation of the virtual functions.
- virtual bool [isEqual](#) ([OSnLNodeVariable](#) *that)
A function to check for the equality of two objects.

Public Attributes

- double [coef](#)
coef is an option coefficient on the variable, the default value is 1.0
- int [idx](#)
idx is the index of the variable

6.188.1 Detailed Description

The [OSnLNodeVariable](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 10/05/2005

Since

OS1.0

Remarks

The in-memory representation of the OSnL element <variable>

Definition at line 1473 of file OSnLNode.h.

6.188.2 Constructor & Destructor Documentation

6.188.2.1 OSnLNodeVariable::OSnLNodeVariable ()

default constructor.

6.188.2.2 OSnLNodeVariable::~~OSnLNodeVariable ()

default destructor.

6.188.3 Member Function Documentation

6.188.3.1 virtual void OSnLNodeVariable::getVariableIndexMap (std::map< int, int > * *varIdx*) [virtual]

varIdx is a map where the key is the index of an [OSnLNodeVariable](#) and (*varIdx)[idx] is the kth variable in the map, e.g.

(*varIdx)[5] = 2 means that variable indexed by 5 is the second variable in the [OSnLNode](#) and all of its children

Parameters

a	pointer to a map of the variables in the OSnLNode and its children
---	--

Reimplemented from [OSnLNode](#).

6.188.3.2 virtual std::string OSnLNodeVariable::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

6.188.3.3 virtual std::string OSnLNodeVariable::getTokenName () [virtual]

Returns

a std::string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

6.188.3.4 virtual std::string OSnLNodeVariable::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the variable node.

Reimplemented from [ExprNode](#).

6.188.3.5 virtual double OSnLNodeVariable::calculateFunction (double * x) [virtual]

Calculate the function value given the current variable values.

This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this [OSnLNode](#) class.

Parameters

x	holds the values of the variables in a double array.
---	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

6.188.3.6 OSnLNode * OSnLNodeVariable::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

a pointer to a new [OSnLNode](#) of the proper type.

Implements [ExprNode](#).

6.188.3.7 double OSnLNodeVariable::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

6.188.3.8 virtual bool OSnLNodeVariable::isEqual (OSnLNodeVariable * that) [virtual]

A function to check for the equality of two objects.

6.188.4 Member Data Documentation**6.188.4.1 double OSnLNodeVariable::coef**

coef is an option coefficient on the variable, the default value is 1.0

Definition at line 1480 of file OSnLNode.h.

6.188.4.2 int OSnLNodeVariable::idx

idx is the index of the variable

Definition at line 1483 of file OSnLNode.h.

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

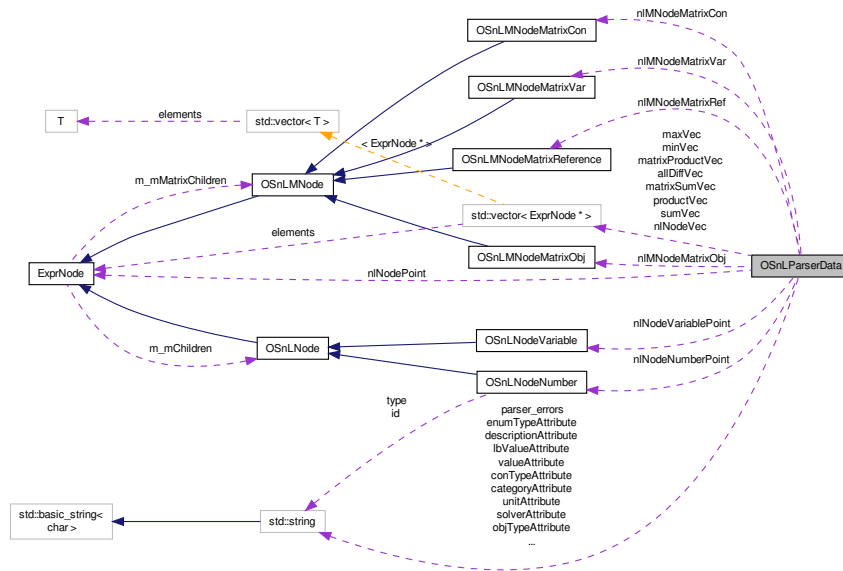
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h](#)

6.189 OSnLParserData Class Reference

The [OSnLParserData](#) Class.

```
#include <OSnLParserData.h>
```

Collaboration diagram for OSnLParserData:



Public Member Functions

- [OSnLParserData \(\)](#)
the *OSnLParserData* class constructor
- [~OSnLParserData \(\)](#)
the *OSnLParserData* class destructor

Public Attributes

- bool [categoryAttributePresent](#)
generic attributes
- std::string [categoryAttribute](#)
- bool [typeAttributePresent](#)
- std::string [typeAttribute](#)
- bool [varTypeAttributePresent](#)
- std::string [varTypeAttribute](#)
- bool [objTypeAttributePresent](#)
- std::string [objTypeAttribute](#)
- bool [conTypeAttributePresent](#)
- std::string [conTypeAttribute](#)
- bool [enumTypeAttributePresent](#)
- std::string [enumTypeAttribute](#)
- bool [nameAttributePresent](#)
- std::string [nameAttribute](#)
- bool [valueAttributePresent](#)
- std::string [valueAttribute](#)

- bool [lbValueAttributePresent](#)
- std::string [lbValueAttribute](#)
- bool [ubValueAttributePresent](#)
- std::string [ubValueAttribute](#)
- bool [descriptionAttributePresent](#)
- std::string [descriptionAttribute](#)
- bool [solverAttributePresent](#)
- std::string [solverAttribute](#)
- bool [unitAttributePresent](#)
- std::string [unitAttribute](#)
- bool [idxAttributePresent](#)
- int [idxAttribute](#)
- bool [shapeAttributePresent](#)
- int [templnt](#)
- some temporary items to facilitate code sharing*
- int [numberOf](#)
- int [kounter](#)
- int [iOther](#)
- int [iOption](#)
- double [tempVal](#)
- std::string [tempStr](#)
- [ExprNode](#) * [nlNodePoint](#)
- These entities are used for parsing <nonlinearExpressions>*
- [OSnLNodeVariable](#) * [nlNodeVariablePoint](#)
- a pointer to an [OSnLNode](#) object that is a variable*
- [OSnLNodeNumber](#) * [nlNodeNumberPoint](#)
- a pointer to an [OSnLNode](#) object that is a number*
- [OSnLMNodeMatrixReference](#) * [nlMNodeMatrixRef](#)
- a pointer to an [OSnLMNode](#) object that is a simple matrix reference*
- [OSnLMNodeMatrixVar](#) * [nlMNodeMatrixVar](#)
- a pointer to an [OSnLMNode](#) object that is a matrixVar reference*
- [OSnLMNodeMatrixObj](#) * [nlMNodeMatrixObj](#)
- a pointer to an [OSnLMNode](#) object that is a matrixObj reference*
- [OSnLMNodeMatrixCon](#) * [nlMNodeMatrixCon](#)
- a pointer to an [OSnLMNode](#) object that is a matrixCon reference*
- int [nlnodenumber](#)
- nlnodenumber is the number of nl nodes in the instance*
- int [tmpnlcount](#)
- tmpnlcount counts the number of nl nodes actually found.*
- bool [numbertypeattON](#)
- numbertypeattON is set to true if the type attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one number attribute*
- bool [numbervalueattON](#)
- numbervalueattON is set to true if the value attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one value attribute*
- bool [numberidattON](#)
- numberidattON is set to true if the id attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one id attribute*
- bool [variableidxattON](#)

variableidxattON is set to true if the *idx* attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one *idx* attribute

- bool [variablecoefattON](#)

variablecoefattON is set to true if the *coeff* attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one *coeff* attribute

- std::vector< [ExprNode](#) * > [nlNodeVec](#)

nlNodeVec holds a vector of pointers to [OSnLNodes](#) and [OSnLMNodes](#). In order to build the expression tree correctly from the prefix notation found in the OSiL file, we need to store both [OSnLNodes](#) and [OSnLMNodes](#) into the same vector of [ExprNodes](#)

- std::vector< [ExprNode](#) * > [sumVec](#)

the [OSnLNodeSum](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [allDiffVec](#)

the [OSnLNodeallDiff](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [productVec](#)

the [OSnLNodeProduct](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [maxVec](#)

the [OSnLNodeMax](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [minVec](#)

the [OSnLNodeMin](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [matrixSumVec](#)

the [OSnLMNodeMatrixSum](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- std::vector< [ExprNode](#) * > [matrixProductVec](#)

the [OSnLMNodeProduct](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

- bool [matrixreftypeattON](#)

matrixreftypeattON is set to true if the *type* attribute has been parsed for an [OSnLMNodeMatrixReference](#) object, an exception is thrown if there is more than one *matrixref* attribute

- bool [matrixidxattON](#)

matrixidxattON is set to true if the *idx* attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one *idx* attribute

- bool [includeDiagonalAttributePresent](#)

Attributes and other data items associated with parsing the [OSnLMNodes](#).

- bool [includeDiagonalAttribute](#)

- char * [errorText](#)

if the parser finds invalid text it is held here and we delete if the file was not valid

- std::string [parser_errors](#)

used to accumulate error message so the parser does not die on the first error encountered

- bool [ignoreDataAfterErrors](#)

two booleans to govern the behavior after an error has been encountered

- bool [suppressFurtherErrorMessage](#)

6.189.1 Detailed Description

The [OSnLParserData](#) Class.

Remarks

The [OSnLParserData](#) class is used to temporarily hold data found in parsing the OSnL schema elements. We do this so we can have a reentrant parser.

Definition at line 29 of file OSnLParserData.h.

6.189.2 Constructor & Destructor Documentation

6.189.2.1 OSnLParserData::OSnLParserData ()

the [OSnLParserData](#) class constructor

6.189.2.2 OSnLParserData::~OSnLParserData ()

the [OSnLParserData](#) class destructor

6.189.3 Member Data Documentation

6.189.3.1 bool OSnLParserData::categoryAttributePresent

generic attributes

Definition at line 34 of file OSnLParserData.h.

6.189.3.2 std::string OSnLParserData::categoryAttribute

Definition at line 35 of file OSnLParserData.h.

6.189.3.3 bool OSnLParserData::typeAttributePresent

Definition at line 36 of file OSnLParserData.h.

6.189.3.4 std::string OSnLParserData::typeAttribute

Definition at line 37 of file OSnLParserData.h.

6.189.3.5 bool OSnLParserData::varTypeAttributePresent

Definition at line 38 of file OSnLParserData.h.

6.189.3.6 std::string OSnLParserData::varTypeAttribute

Definition at line 39 of file OSnLParserData.h.

6.189.3.7 bool OSnLParserData::objTypeAttributePresent

Definition at line 40 of file OSnLParserData.h.

6.189.3.8 std::string OSnLParserData::objTypeAttribute

Definition at line 41 of file OSnLParserData.h.

6.189.3.9 bool OSnLParserData::conTypeAttributePresent

Definition at line 42 of file OSnLParserData.h.

6.189.3.10 std::string OSnLParserData::conTypeAttribute

Definition at line 43 of file OSnLParserData.h.

6.189.3.11 bool OSnLParserData::enumTypeAttributePresent

Definition at line 44 of file OSnLParserData.h.

6.189.3.12 std::string OSnLParserData::enumTypeAttribute

Definition at line 45 of file OSnLParserData.h.

6.189.3.13 bool OSnLParserData::nameAttributePresent

Definition at line 46 of file OSnLParserData.h.

6.189.3.14 std::string OSnLParserData::nameAttribute

Definition at line 47 of file OSnLParserData.h.

6.189.3.15 bool OSnLParserData::valueAttributePresent

Definition at line 48 of file OSnLParserData.h.

6.189.3.16 std::string OSnLParserData::valueAttribute

Definition at line 49 of file OSnLParserData.h.

6.189.3.17 bool OSnLParserData::lbValueAttributePresent

Definition at line 50 of file OSnLParserData.h.

6.189.3.18 std::string OSnLParserData::lbValueAttribute

Definition at line 51 of file OSnLParserData.h.

6.189.3.19 bool OSnLParserData::ubValueAttributePresent

Definition at line 52 of file OSnLParserData.h.

6.189.3.20 std::string OSnLParserData::ubValueAttribute

Definition at line 53 of file OSnLParserData.h.

6.189.3.21 bool OSnLParserData::descriptionAttributePresent

Definition at line 54 of file OSnLParserData.h.

6.189.3.22 std::string OSnLParserData::descriptionAttribute

Definition at line 55 of file OSnLParserData.h.

6.189.3.23 bool OSnLParserData::solverAttributePresent

Definition at line 56 of file OSnLParserData.h.

6.189.3.24 std::string OSnLParserData::solverAttribute

Definition at line 57 of file OSnLParserData.h.

6.189.3.25 bool OSnLParserData::unitAttributePresent

Definition at line 58 of file OSnLParserData.h.

6.189.3.26 std::string OSnLParserData::unitAttribute

Definition at line 59 of file OSnLParserData.h.

6.189.3.27 bool OSnLParserData::idxAttributePresent

Definition at line 60 of file OSnLParserData.h.

6.189.3.28 int OSnLParserData::idxAttribute

Definition at line 61 of file OSnLParserData.h.

6.189.3.29 bool OSnLParserData::shapeAttributePresent

Definition at line 62 of file OSnLParserData.h.

6.189.3.30 int OSnLParserData::templnt

some temporary items to facilitate code sharing

Definition at line 65 of file OSnLParserData.h.

6.189.3.31 int OSnLParserData::numberOf

Definition at line 66 of file OSnLParserData.h.

6.189.3.32 int OSnLParserData::kounter

Definition at line 67 of file OSnLParserData.h.

6.189.3.33 int OSnLParserData::iOther

Definition at line 68 of file OSnLParserData.h.

6.189.3.34 int OSnLParserData::iOption

Definition at line 69 of file OSnLParserData.h.

6.189.3.35 double OSnLParserData::tempVal

Definition at line 70 of file OSnLParserData.h.

6.189.3.36 std::string OSnLParserData::tempStr

Definition at line 71 of file OSnLParserData.h.

6.189.3.37 ExprNode* OSnLParserData::nlNodePoint

These entities are used for parsing <nonlinearExpressions>

a pointer to an [OSnLNode](#) object

Definition at line 83 of file OSnLParserData.h.

6.189.3.38 OSnLNodeVariable* OSnLParserData::nlNodeVariablePoint

a pointer to an [OSnLNode](#) object that is a variable

Definition at line 86 of file OSnLParserData.h.

6.189.3.39 OSnLNodeNumber* OSnLParserData::nlNodeNumberPoint

a pointer to an [OSnLNode](#) object that is a number

Definition at line 89 of file OSnLParserData.h.

6.189.3.40 OSnLMNodeMatrixReference* OSnLParserData::nlMNodeMatrixRef

a pointer to an [OSnLMNode](#) object that is a simple matrix reference

Definition at line 92 of file OSnLParserData.h.

6.189.3.41 OSnLMNodeMatrixVar* OSnLParserData::nlMNodeMatrixVar

a pointer to an [OSnLMNode](#) object that is a matrixVar reference

Definition at line 95 of file OSnLParserData.h.

6.189.3.42 OSnLMNodeMatrixObj* OSnLParserData::nlMNodeMatrixObj

a pointer to an [OSnLMNode](#) object that is a matrixObj reference

Definition at line 98 of file OSnLParserData.h.

6.189.3.43 OSnLMNodeMatrixCon* OSnLParserData::nlMNodeMatrixCon

a pointer to an [OSnLMNode](#) object that is a matrixCon reference

Definition at line 101 of file OSnLParserData.h.

6.189.3.44 int OSnLParserData::nlnodenumber

nlnodenumber is the number of nl nodes in the instance

Definition at line 104 of file OSnLParserData.h.

6.189.3.45 int OSnLParserData::tmpnlcount

tmpnlcount counts the number of nl nodes actually found.

If this number differs from nlnodenumber, then an exception is thrown

Definition at line 109 of file OSnLParserData.h.

6.189.3.46 bool OSnLParserData::numbertypeattON

numbertypeattON is set to true if the type attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one number attribute

Definition at line 115 of file OSnLParserData.h.

6.189.3.47 bool OSnLParserData::numbervalueattON

numbervalueattON is set to true if the value attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one value attribute

Definition at line 121 of file OSnLParserData.h.

6.189.3.48 bool OSnLParserData::numberidattON

numberidattON is set to true if the id attribute has been parsed for an [OSnLNodeNumber](#) object, an exception is thrown if there is more than one id attribute

Definition at line 127 of file OSnLParserData.h.

6.189.3.49 bool OSnLParserData::variableidxattON

variableidxattON is set to true if the idx attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one idx attribute

Definition at line 133 of file OSnLParserData.h.

6.189.3.50 bool OSnLParserData::variablecoefattON

variablecoefattON is set to true if the coeff attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one coeff attribute

Definition at line 139 of file OSnLParserData.h.

6.189.3.51 std::vector<ExprNode*> OSnLParserData::nlNodeVec

nlNodeVec holds a vector of pointers to OSnLNodes and OSnLMNodes In order to build the expression tree correctly from the prefix notation found in the OSiL file, we need to store both OSnLNodes and OSnLMNodes into the same vector of ExprNodes

Definition at line 146 of file OSnLParserData.h.

6.189.3.52 std::vector<ExprNode*> OSnLParserData::sumVec

the [OSnLNodeSum](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 152 of file OSnLParserData.h.

6.189.3.53 std::vector<ExprNode*> OSnLParserData::allDiffVec

the OSnLNodeallDiff node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 158 of file OSnLParserData.h.

6.189.3.54 std::vector<ExprNode*> OSnLParserData::productVec

the [OSnLNodeProduct](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 164 of file OSnLParserData.h.

6.189.3.55 `std::vector<ExprNode*> OSnLParserData::maxVec`

the [OSnLNodeMax](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 170 of file OSnLParserData.h.

6.189.3.56 `std::vector<ExprNode*> OSnLParserData::minVec`

the [OSnLNodeMin](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 176 of file OSnLParserData.h.

6.189.3.57 `std::vector<ExprNode*> OSnLParserData::matrixSumVec`

the [OSnLMNodeMatrixSum](#) node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 182 of file OSnLParserData.h.

6.189.3.58 `std::vector<ExprNode*> OSnLParserData::matrixProductVec`

the OSnLMNodeProduct node can have any number of children, including other children with an indeterminate number of children so when parsing we need to temporarily store all of its children

Definition at line 188 of file OSnLParserData.h.

6.189.3.59 `bool OSnLParserData::matrixreftypeattON`

matrixreftypeattON is set to true if the type attribute has been parsed for an [OSnLMNodeMatrixReference](#) object, an exception is thrown if there is more than one matrixref attribute

Definition at line 194 of file OSnLParserData.h.

6.189.3.60 `bool OSnLParserData::matrixidxattON`

matrixidxattON is set to true if the idx attribute has been parsed for an [OSnLNodeVariable](#), an exception is thrown if there is more than one idx attribute

Definition at line 200 of file OSnLParserData.h.

6.189.3.61 `bool OSnLParserData::includeDiagonalAttributePresent`

Attributes and other data items associated with parsing the OSnLMNodes.

Definition at line 203 of file OSnLParserData.h.

6.189.3.62 `bool OSnLParserData::includeDiagonalAttribute`

Definition at line 204 of file OSnLParserData.h.

6.189.3.63 `char* OSnLParserData::errorText`

if the parser finds invalid text it is held here and we delete if the file was not valid

Definition at line 209 of file OSnLParserData.h.

6.189.3.64 `std::string OSnLParserData::parser_errors`

used to accumulate error message so the parser does not die on the first error encountered

Definition at line 214 of file OSnLParserData.h.

6.189.3.65 bool OSnLParserData::ignoreDataAfterErrors

two booleans to govern the behavior after an error has been encountered

Definition at line 217 of file OSnLParserData.h.

6.189.3.66 bool OSnLParserData::suppressFurtherErrorMessage

Definition at line 218 of file OSnLParserData.h.

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

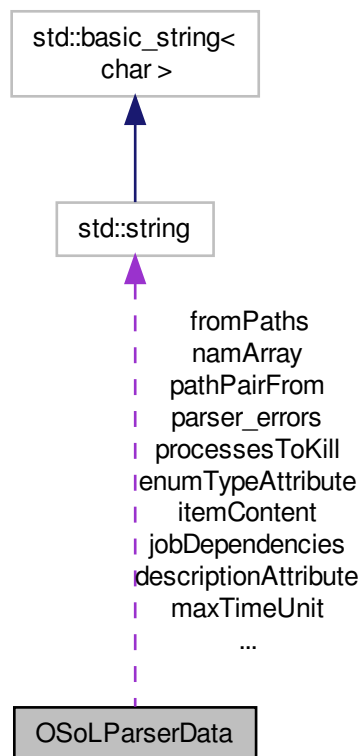
- </home/ted/COIN/trunk/OS/src/OSParsers/OSnLParserData.h>

6.190 OSoLParserData Class Reference

The [OSoLParserData](#) Class.

```
#include <OSoLParserData.h>
```

Collaboration diagram for OSoLParserData:



Public Member Functions

- [OSoLParserData \(\)](#)
the [OSoLParserData](#) class constructor
- [~OSoLParserData \(\)](#)
the [OSoLParserData](#) class destructor

Public Attributes

- bool [osolgeneralPresent](#)
track which child elements are present
- bool [osolsystemPresent](#)
- bool [osolservicePresent](#)
- bool [osoljobPresent](#)
- bool [osoloptimizationPresent](#)
- bool [serviceURIPresent](#)
children of the <general> element
- bool [serviceNamePresent](#)
- bool [instanceNamePresent](#)
- bool [instanceLocationPresent](#)
- bool [instanceLocationTypeattON](#)
- bool [jobIDPresent](#)
- bool [solverToInvokePresent](#)
- bool [licensePresent](#)
- bool [usernamePresent](#)
- bool [passwordPresent](#)
- bool [contactPresent](#)
- bool [transportTypeattON](#)
- bool [otherGeneralOptionsPresent](#)
- int [numberOfOtherGeneralOptions](#)
- bool [minDiskSpacePresent](#)
children of the <system> element
- bool [minDiskSpaceUnitPresent](#)
- bool [minMemoryPresent](#)
- bool [minMemoryUnitPresent](#)
- bool [minCPUSpeedPresent](#)
- bool [minCPUSpeedUnitPresent](#)
- bool [minCPUNumberPresent](#)
- bool [otherSystemOptionsPresent](#)
- int [numberOfOtherSystemOptions](#)
- bool [serviceTypePresent](#)
children of the <service> element
- bool [otherServiceOptionsPresent](#)
- int [numberOfOtherServiceOptions](#)
- bool [maxTimePresent](#)
children of the <job> element
- std::string [maxTimeUnit](#)
- bool [maxTimeUnitPresent](#)
- double [maxTimeValue](#)

- bool requestedStartTimePresent
- std::string requestedStartTime
- bool dependenciesPresent
- int numberOfDependencies
- bool requiredDirectoriesPresent
- int numberOfRequiredDirectories
- bool requiredFilesPresent
- int numberOfRequiredFiles
- bool directoriesToMakePresent
- int numberOfDirectoriesToMake
- bool filesToMakePresent
- int numberOfFilesToMake
- std::string pathPairFrom
- std::string pathPairTo
- bool pathPairFromPresent
- bool pathPairToPresent
- bool pathPairMakeCopyPresent
- bool pathPairMakeCopy
- bool inputDirectoriesToMovePresent
- int numberOfInputDirectoriesToMove
- bool inputFilesToMovePresent
- int numberOfInputFilesToMove
- bool outputDirectoriesToMovePresent
- int numberOfOutputDirectoriesToMove
- bool outputFilesToMovePresent
- int numberOfOutputFilesToMove
- bool directoriesToDeletePresent
- int numberOfDirectoriesToDelete
- bool filesToDeletePresent
- int numberOfFilesToDelete
- bool processesToKillPresent
- int numberOfProcessesToKill
- bool otherJobOptionsPresent
- int numberOfOtherJobOptions
- int numberOfPathPairs
- int numberOfVariables

children of the <optimization> element

- bool numberOfVariablesPresent
- int numberOfObjectives
- bool numberOfObjectivesPresent
- int numberOfConstraints
- bool numberOfConstraintsPresent
- bool variablesPresent
- bool objectivesPresent
- bool constraintsPresent
- bool solverOptionsPresent
- bool idxAttributePresent
- bool valAttributePresent
- bool lbValAttributePresent
- bool ubValAttributePresent

- int [numberOfOtherVariableOptions](#)
- bool [initialVariableValuesPresent](#)
- int [numberOfVar](#)
- bool [initialVariableValuesStringPresent](#)
- int [numberOfVarStr](#)
- bool [initialBasisStatusPresent](#)
- int [numberOfBasVar](#)
- bool [sosIdxAttributePresent](#)
- bool [groupWeightAttributePresent](#)
- bool [numberOfVarAttributePresent](#)
- bool [numberOfObjAttributePresent](#)
- bool [numberOfConAttributePresent](#)
- bool [numberOfEnumerationsAttributePresent](#)
- int [numberOfIntWt](#)
- int [numberOfSOS](#)
- int [currentSOS](#)
- int [sosIdx](#)
- int [numberOfSOSVar](#)
- int [numberOfObj](#)
- int [numberOfOtherObjectiveOptions](#)
- bool [initialObjectiveValuesPresent](#)
- int [numberOfObjValues](#)
- bool [initialObjectiveBoundsPresent](#)
- int [numberOfObjBounds](#)
- int [numberOfCon](#)
- int [numberOfOtherConstraintOptions](#)
- bool [initialConstraintValuesPresent](#)
- bool [initialDualVariableValuesPresent](#)
- int [numberOfDuals](#)
- int [numberOfSolverOptions](#)
- double [groupWeight](#)
- double [lbDualValue](#)
- double [ubDualValue](#)
- int [numberOfEnumerations](#)
- int [otherOptionType](#)
- bool [otherOptionNumberPresent](#)

attributes of <other> options

- bool [otherOptionNamePresent](#)
- bool [otherOptionValuePresent](#)
- bool [otherOptionSolverPresent](#)
- bool [otherOptionCategoryPresent](#)
- bool [otherOptionTypePresent](#)
- bool [otherOptionDescriptionPresent](#)
- bool [numberOfItemsPresent](#)
- int [numberOfItems](#)
- bool [solverOptionNamePresent](#)

attributes of <solverOptions> element

- bool [solverOptionValuePresent](#)
- bool [solverOptionSolverPresent](#)
- bool [solverOptionCategoryPresent](#)

- bool [solverOptionTypePresent](#)
- bool [solverOptionDescriptionPresent](#)
- std::string [itemContent](#)
- bool [categoryAttributePresent](#)
- generic attributes*
- std::string [categoryAttribute](#)
- bool [typeAttributePresent](#)
- std::string [typeAttribute](#)
- bool [varTypeAttributePresent](#)
- std::string [varTypeAttribute](#)
- bool [objTypeAttributePresent](#)
- std::string [objTypeAttribute](#)
- bool [conTypeAttributePresent](#)
- std::string [conTypeAttribute](#)
- bool [enumTypeAttributePresent](#)
- std::string [enumTypeAttribute](#)
- bool [nameAttributePresent](#)
- std::string [nameAttribute](#)
- bool [valueAttributePresent](#)
- std::string [valueAttribute](#)
- bool [lbValueAttributePresent](#)
- std::string [lbValueAttribute](#)
- bool [ubValueAttributePresent](#)
- std::string [ubValueAttribute](#)
- bool [descriptionAttributePresent](#)
- std::string [descriptionAttribute](#)
- bool [solverAttributePresent](#)
- std::string [solverAttribute](#)
- bool [unitAttributePresent](#)
- std::string [unitAttribute](#)
- int [idxAttribute](#)
- std::string * [jobDependencies](#)
- all arrays are collected here*
- std::string * [paths](#)
- std::string * [fromPaths](#)
- std::string * [toPaths](#)
- std::string * [processesToKill](#)
- std::string * [valueString](#)
- std::string * [lbValueString](#)
- std::string * [ubValueString](#)
- std::string * [itemList](#)
- bool * [makeCopy](#)
- int * [idxArray](#)
- double * [valArray](#)
- double * [lbValArray](#)
- double * [ubValArray](#)
- std::string * [namArray](#)
- int [templnt](#)
- some temporary items to facilitate code sharing*
- int [numberOf](#)

- int [kounter](#)
- int [iOther](#)
- int [iOption](#)
- double [tempVal](#)
- std::string [tempStr](#)
- std::string [statusType](#)
the status type of the result
- std::string [statusDescription](#)
the status Description of the solution
- void * [scanner](#)
scanner is used to store data in a reentrant lexer we use this to pass an [OSoLParserData](#) object to the parser
- char * [errorText](#)
if the parser finds invalid text it is held here and we delete if the file was not valid
- std::string [parser_errors](#)
used to accumulate error message so the parser does not die on the first error encountered
- bool [ignoreDataAfterErrors](#)
two booleans to govern the behavior after an error has been encountered
- bool [suppressFurtherErrorMessage](#)

6.190.1 Detailed Description

The [OSoLParserData](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 08/29/2008

Since

OS 1.1

Remarks

the [OSoLParserData](#) class is used to temporarily hold data found in parsing the OSOL instance we do this so we can have a reentrant parser.

Definition at line 33 of file OSoLParserData.h.

6.190.2 Constructor & Destructor Documentation

6.190.2.1 OSoLParserData::OSoLParserData ()

the [OSoLParserData](#) class constructor

6.190.2.2 OSoLParserData::~OSoLParserData ()

the [OSoLParserData](#) class destructor

6.190.3 Member Data Documentation

6.190.3.1 bool OSoLParserData::osolgeneralPresent

track which child elements are present

Definition at line 37 of file OSoLParserData.h.

6.190.3.2 bool OSoLParserData::osolsystemPresent

Definition at line 38 of file OSoLParserData.h.

6.190.3.3 bool OSoLParserData::osolservicePresent

Definition at line 39 of file OSoLParserData.h.

6.190.3.4 bool OSoLParserData::osoljobPresent

Definition at line 40 of file OSoLParserData.h.

6.190.3.5 bool OSoLParserData::osoloptimizationPresent

Definition at line 41 of file OSoLParserData.h.

6.190.3.6 bool OSoLParserData::serviceURIPresent

children of the <general> element

Definition at line 45 of file OSoLParserData.h.

6.190.3.7 bool OSoLParserData::serviceNamePresent

Definition at line 46 of file OSoLParserData.h.

6.190.3.8 bool OSoLParserData::instanceNamePresent

Definition at line 47 of file OSoLParserData.h.

6.190.3.9 bool OSoLParserData::instanceLocationPresent

Definition at line 48 of file OSoLParserData.h.

6.190.3.10 bool OSoLParserData::instanceLocationTypeattON

Definition at line 49 of file OSoLParserData.h.

6.190.3.11 bool OSoLParserData::jobIDPresent

Definition at line 50 of file OSoLParserData.h.

6.190.3.12 bool OSoLParserData::solverToInvokePresent

Definition at line 51 of file OSoLParserData.h.

6.190.3.13 bool OSoLParserData::licensePresent

Definition at line 52 of file OSoLParserData.h.

6.190.3.14 bool OSoLParserData::usernamePresent

Definition at line 53 of file OSoLParserData.h.

6.190.3.15 bool OSoLParserData::passwordPresent

Definition at line 54 of file OSoLParserData.h.

6.190.3.16 bool OSoLParserData::contactPresent

Definition at line 55 of file OSoLParserData.h.

6.190.3.17 bool OSoLParserData::transportTypeattON

Definition at line 56 of file OSoLParserData.h.

6.190.3.18 bool OSoLParserData::otherGeneralOptionsPresent

Definition at line 57 of file OSoLParserData.h.

6.190.3.19 int OSoLParserData::numberOfOtherGeneralOptions

Definition at line 58 of file OSoLParserData.h.

6.190.3.20 bool OSoLParserData::minDiskSpacePresent

children of the <system> element

Definition at line 62 of file OSoLParserData.h.

6.190.3.21 bool OSoLParserData::minDiskSpaceUnitPresent

Definition at line 63 of file OSoLParserData.h.

6.190.3.22 bool OSoLParserData::minMemoryPresent

Definition at line 65 of file OSoLParserData.h.

6.190.3.23 bool OSoLParserData::minMemoryUnitPresent

Definition at line 66 of file OSoLParserData.h.

6.190.3.24 bool OSoLParserData::minCPUSpeedPresent

Definition at line 68 of file OSoLParserData.h.

6.190.3.25 bool OSoLParserData::minCPUSpeedUnitPresent

Definition at line 69 of file OSoLParserData.h.

6.190.3.26 bool OSoLParserData::minCPUNumberPresent

Definition at line 71 of file OSoLParserData.h.

6.190.3.27 bool OSoLParserData::otherSystemOptionsPresent

Definition at line 73 of file OSoLParserData.h.

6.190.3.28 int OSoLParserData::numberOfOtherSystemOptions

Definition at line 74 of file OSoLParserData.h.

6.190.3.29 bool OSoLParserData::serviceTypePresent

children of the <service> element

Definition at line 78 of file OSoLParserData.h.

6.190.3.30 bool OSoLParserData::otherServiceOptionsPresent

Definition at line 80 of file OSoLParserData.h.

6.190.3.31 int OSoLParserData::numberOfOtherServiceOptions

Definition at line 81 of file OSoLParserData.h.

6.190.3.32 bool OSoLParserData::maxTimePresent

children of the <job> element

Definition at line 85 of file OSoLParserData.h.

6.190.3.33 std::string OSoLParserData::maxTimeUnit

Definition at line 86 of file OSoLParserData.h.

6.190.3.34 bool OSoLParserData::maxTimeUnitPresent

Definition at line 87 of file OSoLParserData.h.

6.190.3.35 double OSoLParserData::maxTimeValue

Definition at line 88 of file OSoLParserData.h.

6.190.3.36 bool OSoLParserData::requestedStartTimePresent

Definition at line 90 of file OSoLParserData.h.

6.190.3.37 std::string OSoLParserData::requestedStartTime

Definition at line 91 of file OSoLParserData.h.

6.190.3.38 bool OSoLParserData::dependenciesPresent

Definition at line 93 of file OSoLParserData.h.

6.190.3.39 int OSoLParserData::numberOfDependencies

Definition at line 94 of file OSoLParserData.h.

6.190.3.40 bool OSoLParserData::requiredDirectoriesPresent

Definition at line 96 of file OSoLParserData.h.

6.190.3.41 int OSoLParserData::numberOfRequiredDirectories

Definition at line 97 of file OSoLParserData.h.

6.190.3.42 `bool OSoLParserData::requiredFilesPresent`

Definition at line 99 of file OSoLParserData.h.

6.190.3.43 `int OSoLParserData::numberOfRequiredFiles`

Definition at line 100 of file OSoLParserData.h.

6.190.3.44 `bool OSoLParserData::directoriesToMakePresent`

Definition at line 102 of file OSoLParserData.h.

6.190.3.45 `int OSoLParserData::numberOfDirectoriesToMake`

Definition at line 103 of file OSoLParserData.h.

6.190.3.46 `bool OSoLParserData::filesToMakePresent`

Definition at line 105 of file OSoLParserData.h.

6.190.3.47 `int OSoLParserData::numberOfFilesToMake`

Definition at line 106 of file OSoLParserData.h.

6.190.3.48 `std::string OSoLParserData::pathPairFrom`

Definition at line 108 of file OSoLParserData.h.

6.190.3.49 `std::string OSoLParserData::pathPairTo`

Definition at line 109 of file OSoLParserData.h.

6.190.3.50 `bool OSoLParserData::pathPairFromPresent`

Definition at line 111 of file OSoLParserData.h.

6.190.3.51 `bool OSoLParserData::pathPairToPresent`

Definition at line 112 of file OSoLParserData.h.

6.190.3.52 `bool OSoLParserData::pathPairMakeCopyPresent`

Definition at line 113 of file OSoLParserData.h.

6.190.3.53 `bool OSoLParserData::pathPairMakeCopy`

Definition at line 114 of file OSoLParserData.h.

6.190.3.54 `bool OSoLParserData::inputDirectoriesToMovePresent`

Definition at line 116 of file OSoLParserData.h.

6.190.3.55 `int OSoLParserData::numberOfInputDirectoriesToMove`

Definition at line 117 of file OSoLParserData.h.

6.190.3.56 `bool OSoLParserData::inputFilesToMovePresent`

Definition at line 119 of file OSoLParserData.h.

6.190.3.57 `int OSoLParserData::numberOfInputFilesToMove`

Definition at line 120 of file OSoLParserData.h.

6.190.3.58 `bool OSoLParserData::outputDirectoriesToMovePresent`

Definition at line 122 of file OSoLParserData.h.

6.190.3.59 `int OSoLParserData::numberOfOutputDirectoriesToMove`

Definition at line 123 of file OSoLParserData.h.

6.190.3.60 `bool OSoLParserData::outputFilesToMovePresent`

Definition at line 125 of file OSoLParserData.h.

6.190.3.61 `int OSoLParserData::numberOfOutputFilesToMove`

Definition at line 126 of file OSoLParserData.h.

6.190.3.62 `bool OSoLParserData::directoriesToDeletePresent`

Definition at line 128 of file OSoLParserData.h.

6.190.3.63 `int OSoLParserData::numberOfDirectoriesToDelete`

Definition at line 129 of file OSoLParserData.h.

6.190.3.64 `bool OSoLParserData::filesToDeletePresent`

Definition at line 131 of file OSoLParserData.h.

6.190.3.65 `int OSoLParserData::numberOfFilesToDelete`

Definition at line 132 of file OSoLParserData.h.

6.190.3.66 `bool OSoLParserData::processesToKillPresent`

Definition at line 134 of file OSoLParserData.h.

6.190.3.67 `int OSoLParserData::numberOfProcessesToKill`

Definition at line 135 of file OSoLParserData.h.

6.190.3.68 `bool OSoLParserData::otherJobOptionsPresent`

Definition at line 137 of file OSoLParserData.h.

6.190.3.69 `int OSoLParserData::numberOfOtherJobOptions`

Definition at line 138 of file OSoLParserData.h.

6.190.3.70 `int OSoLParserData::numberOfPathPairs`

Definition at line 140 of file OSoLParserData.h.

6.190.3.71 `int OSoLParserData::numberOfVariables`

children of the <optimization> element

Definition at line 143 of file OSoLParserData.h.

6.190.3.72 `bool OSoLParserData::numberOfVariablesPresent`

Definition at line 144 of file OSoLParserData.h.

6.190.3.73 `int OSoLParserData::numberOfObjectives`

Definition at line 145 of file OSoLParserData.h.

6.190.3.74 `bool OSoLParserData::numberOfObjectivesPresent`

Definition at line 146 of file OSoLParserData.h.

6.190.3.75 `int OSoLParserData::numberOfConstraints`

Definition at line 147 of file OSoLParserData.h.

6.190.3.76 `bool OSoLParserData::numberOfConstraintsPresent`

Definition at line 148 of file OSoLParserData.h.

6.190.3.77 `bool OSoLParserData::variablesPresent`

Definition at line 149 of file OSoLParserData.h.

6.190.3.78 `bool OSoLParserData::objectivesPresent`

Definition at line 150 of file OSoLParserData.h.

6.190.3.79 `bool OSoLParserData::constraintsPresent`

Definition at line 151 of file OSoLParserData.h.

6.190.3.80 `bool OSoLParserData::solverOptionsPresent`

Definition at line 152 of file OSoLParserData.h.

6.190.3.81 `bool OSoLParserData::idxAttributePresent`

Definition at line 153 of file OSoLParserData.h.

6.190.3.82 `bool OSoLParserData::valAttributePresent`

Definition at line 154 of file OSoLParserData.h.

6.190.3.83 `bool OSoLParserData::lbValAttributePresent`

Definition at line 155 of file OSoLParserData.h.

6.190.3.84 bool OSoLParserData::ubValAttributePresent

Definition at line 156 of file OSoLParserData.h.

6.190.3.85 int OSoLParserData::numberOfOtherVariableOptions

Definition at line 157 of file OSoLParserData.h.

6.190.3.86 bool OSoLParserData::initialVariableValuesPresent

Definition at line 158 of file OSoLParserData.h.

6.190.3.87 int OSoLParserData::numberOfVar

Definition at line 159 of file OSoLParserData.h.

6.190.3.88 bool OSoLParserData::initialVariableValuesStringPresent

Definition at line 160 of file OSoLParserData.h.

6.190.3.89 int OSoLParserData::numberOfVarStr

Definition at line 161 of file OSoLParserData.h.

6.190.3.90 bool OSoLParserData::initialBasisStatusPresent

Definition at line 162 of file OSoLParserData.h.

6.190.3.91 int OSoLParserData::numberOfBasVar

Definition at line 163 of file OSoLParserData.h.

6.190.3.92 bool OSoLParserData::sosIdxAttributePresent

Definition at line 164 of file OSoLParserData.h.

6.190.3.93 bool OSoLParserData::groupWeightAttributePresent

Definition at line 165 of file OSoLParserData.h.

6.190.3.94 bool OSoLParserData::numberOfVarAttributePresent

Definition at line 166 of file OSoLParserData.h.

6.190.3.95 bool OSoLParserData::numberOfObjAttributePresent

Definition at line 167 of file OSoLParserData.h.

6.190.3.96 bool OSoLParserData::numberOfConAttributePresent

Definition at line 168 of file OSoLParserData.h.

6.190.3.97 bool OSoLParserData::numberOfEnumerationsAttributePresent

Definition at line 169 of file OSoLParserData.h.

6.190.3.98 `int OSoLParserData::numberOfIntWt`

Definition at line 170 of file OSoLParserData.h.

6.190.3.99 `int OSoLParserData::numberOfSOS`

Definition at line 171 of file OSoLParserData.h.

6.190.3.100 `int OSoLParserData::currentSOS`

Definition at line 172 of file OSoLParserData.h.

6.190.3.101 `int OSoLParserData::sosIdx`

Definition at line 173 of file OSoLParserData.h.

6.190.3.102 `int OSoLParserData::numberOfSOSVar`

Definition at line 174 of file OSoLParserData.h.

6.190.3.103 `int OSoLParserData::numberOfObj`

Definition at line 175 of file OSoLParserData.h.

6.190.3.104 `int OSoLParserData::numberOfOtherObjectiveOptions`

Definition at line 176 of file OSoLParserData.h.

6.190.3.105 `bool OSoLParserData::initialObjectiveValuesPresent`

Definition at line 177 of file OSoLParserData.h.

6.190.3.106 `int OSoLParserData::numberOfObjValues`

Definition at line 178 of file OSoLParserData.h.

6.190.3.107 `bool OSoLParserData::initialObjectiveBoundsPresent`

Definition at line 179 of file OSoLParserData.h.

6.190.3.108 `int OSoLParserData::numberOfObjBounds`

Definition at line 180 of file OSoLParserData.h.

6.190.3.109 `int OSoLParserData::numberOfCon`

Definition at line 181 of file OSoLParserData.h.

6.190.3.110 `int OSoLParserData::numberOfOtherConstraintOptions`

Definition at line 182 of file OSoLParserData.h.

6.190.3.111 `bool OSoLParserData::initialConstraintValuesPresent`

Definition at line 183 of file OSoLParserData.h.

6.190.3.112 `bool OSoLParserData::initialDualVariableValuesPresent`

Definition at line 184 of file OSoLParserData.h.

6.190.3.113 `int OSoLParserData::numberOfDuals`

Definition at line 185 of file OSoLParserData.h.

6.190.3.114 `int OSoLParserData::numberOfSolverOptions`

Definition at line 186 of file OSoLParserData.h.

6.190.3.115 `double OSoLParserData::groupWeight`

Definition at line 187 of file OSoLParserData.h.

6.190.3.116 `double OSoLParserData::lbDualValue`

Definition at line 188 of file OSoLParserData.h.

6.190.3.117 `double OSoLParserData::ubDualValue`

Definition at line 189 of file OSoLParserData.h.

6.190.3.118 `int OSoLParserData::numberOfEnumerations`

Definition at line 190 of file OSoLParserData.h.

6.190.3.119 `int OSoLParserData::otherOptionType`

Definition at line 191 of file OSoLParserData.h.

6.190.3.120 `bool OSoLParserData::otherOptionNumberPresent`

attributes of <other> options

Definition at line 194 of file OSoLParserData.h.

6.190.3.121 `bool OSoLParserData::otherOptionNamePresent`

Definition at line 195 of file OSoLParserData.h.

6.190.3.122 `bool OSoLParserData::otherOptionValuePresent`

Definition at line 196 of file OSoLParserData.h.

6.190.3.123 `bool OSoLParserData::otherOptionSolverPresent`

Definition at line 197 of file OSoLParserData.h.

6.190.3.124 `bool OSoLParserData::otherOptionCategoryPresent`

Definition at line 198 of file OSoLParserData.h.

6.190.3.125 `bool OSoLParserData::otherOptionTypePresent`

Definition at line 199 of file OSoLParserData.h.

6.190.3.126 `bool OSoLParserData::otherOptionDescriptionPresent`

Definition at line 200 of file OSoLParserData.h.

6.190.3.127 `bool OSoLParserData::numberOfItemsPresent`

Definition at line 201 of file OSoLParserData.h.

6.190.3.128 `int OSoLParserData::numberOfItems`

Definition at line 202 of file OSoLParserData.h.

6.190.3.129 `bool OSoLParserData::solverOptionNamePresent`

attributes of <solverOptions> element

Definition at line 205 of file OSoLParserData.h.

6.190.3.130 `bool OSoLParserData::solverOptionValuePresent`

Definition at line 206 of file OSoLParserData.h.

6.190.3.131 `bool OSoLParserData::solverOptionSolverPresent`

Definition at line 207 of file OSoLParserData.h.

6.190.3.132 `bool OSoLParserData::solverOptionCategoryPresent`

Definition at line 208 of file OSoLParserData.h.

6.190.3.133 `bool OSoLParserData::solverOptionTypePresent`

Definition at line 209 of file OSoLParserData.h.

6.190.3.134 `bool OSoLParserData::solverOptionDescriptionPresent`

Definition at line 210 of file OSoLParserData.h.

6.190.3.135 `std::string OSoLParserData::itemContent`

Definition at line 211 of file OSoLParserData.h.

6.190.3.136 `bool OSoLParserData::categoryAttributePresent`

generic attributes

Definition at line 215 of file OSoLParserData.h.

6.190.3.137 `std::string OSoLParserData::categoryAttribute`

Definition at line 216 of file OSoLParserData.h.

6.190.3.138 `bool OSoLParserData::typeAttributePresent`

Definition at line 217 of file OSoLParserData.h.

6.190.3.139 `std::string OSoLParserData::typeAttribute`

Definition at line 218 of file OSoLParserData.h.

6.190.3.140 `bool OSoLParserData::varTypeAttributePresent`

Definition at line 219 of file OSoLParserData.h.

6.190.3.141 `std::string OSoLParserData::varTypeAttribute`

Definition at line 220 of file OSoLParserData.h.

6.190.3.142 `bool OSoLParserData::objTypeAttributePresent`

Definition at line 221 of file OSoLParserData.h.

6.190.3.143 `std::string OSoLParserData::objTypeAttribute`

Definition at line 222 of file OSoLParserData.h.

6.190.3.144 `bool OSoLParserData::conTypeAttributePresent`

Definition at line 223 of file OSoLParserData.h.

6.190.3.145 `std::string OSoLParserData::conTypeAttribute`

Definition at line 224 of file OSoLParserData.h.

6.190.3.146 `bool OSoLParserData::enumTypeAttributePresent`

Definition at line 225 of file OSoLParserData.h.

6.190.3.147 `std::string OSoLParserData::enumTypeAttribute`

Definition at line 226 of file OSoLParserData.h.

6.190.3.148 `bool OSoLParserData::nameAttributePresent`

Definition at line 227 of file OSoLParserData.h.

6.190.3.149 `std::string OSoLParserData::nameAttribute`

Definition at line 228 of file OSoLParserData.h.

6.190.3.150 `bool OSoLParserData::valueAttributePresent`

Definition at line 229 of file OSoLParserData.h.

6.190.3.151 `std::string OSoLParserData::valueAttribute`

Definition at line 230 of file OSoLParserData.h.

6.190.3.152 `bool OSoLParserData::lbValueAttributePresent`

Definition at line 231 of file OSoLParserData.h.

6.190.3.153 `std::string OSoLParserData::lbValueAttribute`

Definition at line 232 of file OSoLParserData.h.

6.190.3.154 `bool OSoLParserData::ubValueAttributePresent`

Definition at line 233 of file OSoLParserData.h.

6.190.3.155 `std::string OSoLParserData::ubValueAttribute`

Definition at line 234 of file OSoLParserData.h.

6.190.3.156 `bool OSoLParserData::descriptionAttributePresent`

Definition at line 235 of file OSoLParserData.h.

6.190.3.157 `std::string OSoLParserData::descriptionAttribute`

Definition at line 236 of file OSoLParserData.h.

6.190.3.158 `bool OSoLParserData::solverAttributePresent`

Definition at line 237 of file OSoLParserData.h.

6.190.3.159 `std::string OSoLParserData::solverAttribute`

Definition at line 238 of file OSoLParserData.h.

6.190.3.160 `bool OSoLParserData::unitAttributePresent`

Definition at line 239 of file OSoLParserData.h.

6.190.3.161 `std::string OSoLParserData::unitAttribute`

Definition at line 240 of file OSoLParserData.h.

6.190.3.162 `int OSoLParserData::idxAttribute`

Definition at line 241 of file OSoLParserData.h.

6.190.3.163 `std::string* OSoLParserData::jobDependencies`

all arrays are collected here

Definition at line 245 of file OSoLParserData.h.

6.190.3.164 `std::string* OSoLParserData::paths`

Definition at line 246 of file OSoLParserData.h.

6.190.3.165 `std::string* OSoLParserData::fromPaths`

Definition at line 247 of file OSoLParserData.h.

6.190.3.166 `std::string* OSoLParserData::toPaths`

Definition at line 248 of file OSoLParserData.h.

6.190.3.167 `std::string* OSoLParserData::processesToKill`

Definition at line 249 of file OSoLParserData.h.

6.190.3.168 `std::string* OSoLParserData::valueString`

Definition at line 251 of file OSoLParserData.h.

6.190.3.169 `std::string* OSoLParserData::lbValueString`

Definition at line 252 of file OSoLParserData.h.

6.190.3.170 `std::string* OSoLParserData::ubValueString`

Definition at line 253 of file OSoLParserData.h.

6.190.3.171 `std::string* OSoLParserData::itemList`

Definition at line 254 of file OSoLParserData.h.

6.190.3.172 `bool* OSoLParserData::makeCopy`

Definition at line 256 of file OSoLParserData.h.

6.190.3.173 `int* OSoLParserData::idxArray`

Definition at line 257 of file OSoLParserData.h.

6.190.3.174 `double* OSoLParserData::valArray`

Definition at line 259 of file OSoLParserData.h.

6.190.3.175 `double* OSoLParserData::lbValArray`

Definition at line 260 of file OSoLParserData.h.

6.190.3.176 `double* OSoLParserData::ubValArray`

Definition at line 261 of file OSoLParserData.h.

6.190.3.177 `std::string* OSoLParserData::namArray`

Definition at line 263 of file OSoLParserData.h.

6.190.3.178 `int OSoLParserData::templnt`

some temporary items to facilitate code sharing

Definition at line 266 of file OSoLParserData.h.

6.190.3.179 `int OSoLParserData::numberOf`

Definition at line 267 of file OSoLParserData.h.

6.190.3.180 `int OSoLParserData::kounter`

Definition at line 268 of file OSoLParserData.h.

6.190.3.181 `int OSoLParserData::iOther`

Definition at line 269 of file OSoLParserData.h.

6.190.3.182 int OSoLParserData::iOption

Definition at line 270 of file OSoLParserData.h.

6.190.3.183 double OSoLParserData::tempVal

Definition at line 271 of file OSoLParserData.h.

6.190.3.184 std::string OSoLParserData::tempStr

Definition at line 272 of file OSoLParserData.h.

6.190.3.185 std::string OSoLParserData::statusType

the status type of the result

Definition at line 281 of file OSoLParserData.h.

6.190.3.186 std::string OSoLParserData::statusDescription

the status Description of the solution

Definition at line 284 of file OSoLParserData.h.

6.190.3.187 void* OSoLParserData::scanner

scanner is used to store data in a reentrant lexer we use this to pass an [OSoLParserData](#) object to the parser

Definition at line 289 of file OSoLParserData.h.

6.190.3.188 char* OSoLParserData::errorText

if the parser finds invalid text it is held here and we delete if the file was not valid

Definition at line 294 of file OSoLParserData.h.

6.190.3.189 std::string OSoLParserData::parser_errors

used to accumulate error message so the parser does not die on the first error encountered

Definition at line 299 of file OSoLParserData.h.

6.190.3.190 bool OSoLParserData::ignoreDataAfterErrors

two booleans to govern the behavior after an error has been encountered

Definition at line 302 of file OSoLParserData.h.

6.190.3.191 bool OSoLParserData::suppressFurtherErrorMessages

Definition at line 303 of file OSoLParserData.h.

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

- [/home/ted/COIN/trunk/OS/src/OSParsers/OSoLParserData.h](#)

6.191 OSoLReader Class Reference

Used to read an OSoL string.

```
#include <OSoLReader.h>
```

Public Member Functions

- [OSoLReader](#) ()
Default constructor.
- [~OSoLReader](#) ()
Class destructor.
- [OSOption](#) * [readOSoL](#) (const std::string &osol) throw (ErrorClass)
parse the OSoL solver options.

6.191.1 Detailed Description

Used to read an OSoL string.

Remarks

This class wraps around the OSoL parser and sends the parser an OSoL string and is returned an [OSOption](#) object.
Definition at line 37 of file OSoLReader.h.

6.191.2 Constructor & Destructor Documentation

6.191.2.1 OSoLReader::OSoLReader ()

Default constructor.

6.191.2.2 OSoLReader::~~OSoLReader ()

Class destructor.

6.191.3 Member Function Documentation

6.191.3.1 OSOption* OSoLReader::readOSoL (const std::string & osol) throw (ErrorClass)

parse the OSoL solver options.

Parameters

<i>osol</i>	is a string that holds the solver options.
-------------	--

Returns

the instance as an [OSOption](#) object.

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

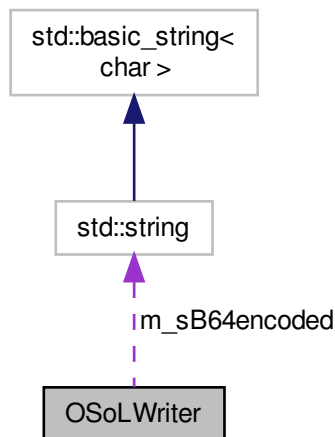
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSoLReader.h](#)

6.192 OSoLWriter Class Reference

Take an [OSOption](#) object and write a string that validates against the OSOL schema.

```
#include "OSoLWriter.h"
```

Collaboration diagram for OSoLWriter:



Public Member Functions

- [OSoLWriter](#) ()
Default constructor.
- [~OSoLWriter](#) ()
Class destructor.
- `std::string` [writeOSoL](#) ([OSOption](#) *theoption)
create an osol string from an [OSOption](#) object

Public Attributes

- `bool` [m_bWriteBase64](#)
m_bWriteBase64 is set to true if we encode the linear constraint coefficients in base64 binary
- `bool` [m_bWhiteSpace](#)
m_bWhiteSpace is set to true if we write white space in the file
- `std::string` [m_sB64encoded](#)
m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

6.192.1 Detailed Description

Take an [OSOption](#) object and write a string that validates against the OSoL schema.

Definition at line 29 of file OSoLWriter.h.

6.192.2 Constructor & Destructor Documentation

6.192.2.1 OSoLWriter::OSoLWriter ()

Default constructor.

6.192.2.2 OSoLWriter::~~OSoLWriter ()

Class destructor.

6.192.3 Member Function Documentation

6.192.3.1 std::string OSoLWriter::writeOSoL (OSOption * *theosoption*)

create an osol string from an [OSOption](#) object

Parameters

<i>theosoption</i>	is a pointer to an OSOption object
--------------------	--

Returns

a string with the [OSOption](#) data that validates against the OSoL schema.

6.192.4 Member Data Documentation

6.192.4.1 bool OSoLWriter::m_bWriteBase64

m_bWriteBase64 is set to true if we encode the linear constraint coefficients in base64 binary

Definition at line 62 of file OSoLWriter.h.

6.192.4.2 bool OSoLWriter::m_bWhiteSpace

m_bWhiteSpace is set to true if we write white space in the file

Definition at line 66 of file OSoLWriter.h.

6.192.4.3 std::string OSoLWriter::m_sB64encoded

m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

Definition at line 71 of file OSoLWriter.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSoLWriter.h](#)

A function to make a deep copy of an *OSOption* object.

- `std::string getFileName ()`
Get the name of the file.
- `std::string getFileSource ()`
Get the source of the file or problem.
- `std::string getFileDescription ()`
Get a description for the file or problem.
- `std::string getFileCreator ()`
Get the name of the person who created the file.
- `std::string getFileLicence ()`
Get licence information associated with the file.
- `std::string getServiceURI ()`
Get the service URI.
- `std::string getServiceName ()`
Get the service name.
- `std::string getInstanceName ()`
Get the instance name.
- `std::string getInstanceLocation ()`
Get the instance location.
- `std::string getInstanceLocationType ()`
Get the location type.
- `std::string getJobID ()`
Get the job ID.
- `std::string getSolverToInvoke ()`
Get the solver name.
- `std::string getLicense ()`
Get the license string.
- `std::string getUsername ()`
Get the user name.
- `std::string getPassword ()`
Get the password.
- `std::string getContact ()`
Get the contact information.
- `std::string getContactTransportType ()`
Get the transport type.
- `std::string getMinDiskSpaceUnit ()`
Get the disk space unit.
- `std::string getMinDiskSpaceDescription ()`
get the disk space description
- `std::string getMinMemoryUnit ()`
Get the memory unit.
- `std::string getMinMemoryDescription ()`
get the memory description
- `std::string getMinCPUSpeedUnit ()`
Get the CPU speed unit.
- `std::string getMinCPUSpeedDescription ()`
Get the CPU speed description.

- `std::string getMinCPUNumberDescription ()`
Get the CPU description.
- `std::string getServiceType ()`
Get the service type.
- `std::string getMaxTimeUnit ()`
Get the time unit.
- `std::string getRequestedStartTime ()`
Get the requested starting time.
- `std::string getOptionStr (std::string optionName)`
Get any of the string-valued options.
- `double getMinDiskSpace ()`
Get the minimum required disk space.
- `double getMinMemorySize ()`
Get the minimum required memory.
- `double getMinCPUSpeed ()`
Get the minimum required CPU speed.
- `double getMaxTime ()`
Get the maximum allowed time.
- `double getOptionDbl (std::string optionName)`
Get any of the double-valued options.
- `int getMinCPUNumber ()`
Get the minimum required number of CPUs.
- `int getNumberOfOtherGeneralOptions ()`
Get the number of <other> options in the <general> element.
- `int getNumberOfOtherSystemOptions ()`
Get the number of <other> options in the <system> element.
- `int getNumberOfOtherServiceOptions ()`
Get the number of <other> options in the <service> element.
- `int getNumberOfOtherJobOptions ()`
Get the number of <other> options in the <job> element.
- `int getNumberOfJobDependencies ()`
Get the number of job dependencies.
- `int getNumberOfRequiredDirectories ()`
Get the number of required directories.
- `int getNumberOfRequiredFiles ()`
Get the number of required files.
- `int getNumberOfDirectoriesToMake ()`
Get the number of directories to make.
- `int getNumberOfFilesToMake ()`
Get the number of files to make.
- `int getNumberOfInputDirectoriesToMove ()`
Get the number of input directories to move.
- `int getNumberOfInputFilesToMove ()`
Get the number of input files to move.
- `int getNumberOfOutputDirectoriesToMove ()`
Get the number of output directories to move.
- `int getNumberOfOutputFilesToMove ()`

- Get the number of output files to move.*

 - `int` [getNumberOfFilesToDelete](#) ()
- Get the number of files to delete.*

 - `int` [getNumberOfDirectoriesToDelete](#) ()
- Get the number of directories to delete.*

 - `int` [getNumberOfProcessesToKill](#) ()
- Get the number of processes to kill.*

 - `int` [getNumberOfVariables](#) ()
- Get the number of variables in the instance.*

 - `int` [getNumberOfObjectives](#) ()
- Get the number of objectives in the instance.*

 - `int` [getNumberOfConstraints](#) ()
- Get the number of constraints in the instance.*

 - `int` [getNumberOfInitVarValues](#) ()
- Get the number of initial variable values.*

 - `int` [getNumberOfInitVarValuesString](#) ()
- Get the number of initial variable strings.*

 - `int` [getNumberOfIntegerVariableBranchingWeights](#) ()
- Get the number of variables for which integer branching weights are provided.*

 - `int` [getNumberOfSOS](#) ()
- Get the number of special ordered sets for which branching weights are provided.*

 - `int` [getNumberOfSOSVarBranchingWeights](#) (int iSOS)
- Get the number of variables for which branching weights are provided in a particular SOS.*

 - `int` [getNumberOfOtherVariableOptions](#) ()
- Get the number of other variable options.*

 - `int` [getNumberOfInitObjValues](#) ()
- Get the number of initial objective values.*

 - `int` [getNumberOfInitObjBounds](#) ()
- Get the number of initial objective bounds.*

 - `int` [getNumberOfOtherObjectiveOptions](#) ()
- Get the number of other objective options.*

 - `int` [getNumberOfInitConValues](#) ()
- Get the number of initial constraint values.*

 - `int` [getNumberOfInitDualVarValues](#) ()
- Get the number of initial dual variable values.*

 - `int` [getNumberOfOtherConstraintOptions](#) ()
- Get the number of other constraint options.*

 - `int` [getNumberOfSolverOptions](#) ()
- Get the number of solver options.*

 - `int` [getOptionInt](#) (std::string optionName)
- Get any of the integer-valued options.*

 - `OtherOption **` [getOtherGeneralOptions](#) ()
- Get the array of other options associated with the <general> element.*

 - `OtherOption **` [getOtherSystemOptions](#) ()
- Get the array of other options associated with the <system> element.*

 - `OtherOption **` [getOtherServiceOptions](#) ()
- Get the array of other options associated with the <service> element.*

- `OtherOption ** getOtherJobOptions ()`
Get the array of other options associated with the <job> element.
- `OtherOption ** getOtherOptions (std::string elementName)`
Get the array of other options associated with any element.
- `OtherOption ** getAllOtherOptions ()`
Get the array of all other options associated with the <general>, <system>, <service> and <job> elements.
- `std::string * getJobDependencies ()`
Get the array of job dependencies.
- `std::string * getRequiredDirectories ()`
Get the array of required directories.
- `std::string * getRequiredFiles ()`
Get the array of required files.
- `std::string * getDirectoriesToMake ()`
Get the array of directories to make.
- `std::string * getFilesToMake ()`
Get the array of files to make.
- `PathPair ** getInputDirectoriesToMove ()`
Get the array of input directories to move.
- `PathPair ** getInputFilesToMove ()`
Get the array of input files to move.
- `PathPair ** getOutputDirectoriesToMove ()`
Get the array of output directories to move.
- `PathPair ** getOutputFilesToMove ()`
Get the array of output files to move.
- `std::string * getDirectoriesToDelete ()`
Get the array of directories to delete.
- `std::string * getFilesToDelete ()`
Get the array of files to delete.
- `std::string * getProcessesToKill ()`
Get the array of processes to kill.
- `InitVarValue ** getInitVarValuesSparse ()`
Get the initial values associated with the variables in sparse form.
- `double * getInitVarValuesDense ()`
Get the initial values associated with the variables in dense form.
- `double * getInitVarValuesDense (int numberOfVariables)`
Get the initial values associated with the variables in dense form.
- `InitVarValueString ** getInitVarValuesStringSparse ()`
Get the initial value strings associated with the variables in sparse form.
- `std::string * getInitVarValuesStringDense ()`
Get the initial value strings associated with the variables in dense form.
- `std::string * getInitVarValuesStringDense (int numberOfVariables)`
Get the initial value strings associated with the variables in dense form.
- `InitBasStatus ** getInitBasisStatusSparse ()`
Get the initial basis status in sparse form.
- `std::string * getInitBasisStatusDense ()`
Get the initial basis information in dense form.
- `int * getVariableInitialBasisStatusDense (int numberOfVariables)`

- Get the initial basis status for all variables in dense form.*

 - int [getNumberOfInitialBasisElements](#) (int type, int status)

Get the number of initial basis elements for a particular variable type and basis status.

 - bool [getInitialBasisElements](#) (int type, int status, int *elem)

Get the initial basis elements for a particular variable type and basis status.

 - [BranchingWeight](#) ** [getIntegerVariableBranchingWeightsSparse](#) ()

Get the integer branching weights in sparse form.

 - double * [getIntegerVariableBranchingWeightsDense](#) ()

Get the integer branching weights in dense form.

 - double * [getIntegerVariableBranchingWeightsDense](#) (int numberOfVariables)

Get the integer branching weights in dense form.

 - [SOSWeights](#) ** [getSOSVariableBranchingWeightsSparse](#) ()

Get the SOS branching weights in sparse form.

 - std::vector
 - < [OtherVariableOption](#) * > [getOtherVariableOptions](#) (std::string solver_name)*Get the <other> variable options associated with a particular solver.*
 - [OtherVariableOption](#) * [getOtherVariableOption](#) (int optionNumber)

Get one particular <other> variable option from the array of options.

 - [OtherVariableOption](#) ** [getAllOtherVariableOptions](#) ()

Get all <other> variable options.

 - [InitObjValue](#) ** [getInitObjValuesSparse](#) ()

Get the initial values associated with the objectives in sparse form.

 - double * [getInitObjValuesDense](#) ()

Get the initial values associated with the objectives in dense form.

 - double * [getInitObjValuesDense](#) (int numberOfObjectives)

Get the initial values associated with the objectives in dense form.

 - [InitObjBound](#) ** [getInitObjBoundsSparse](#) ()

Get the initial bounds associated with the objectives in sparse form.

 - double * [getInitObjLowerBoundsDense](#) ()

Get the initial lower bounds associated with the objectives in dense form.

 - double * [getInitObjLowerBoundsDense](#) (int numberOfObjectives)

Get the initial lower bounds associated with the objectives in dense form.

 - double * [getInitObjUpperBoundsDense](#) ()

Get the initial upper bounds associated with the objectives in dense form.

 - double * [getInitObjUpperBoundsDense](#) (int numberOfObjectives)

Get the initial upper bounds associated with the objectives in dense form.

 - int * [getObjectiveInitialBasisStatusDense](#) (int numberOfObjectives)

Get the initial basis status for all objectives in dense form.

 - std::vector
 - < [OtherObjectiveOption](#) * > [getOtherObjectiveOptions](#) (std::string solver_name)*Get the array of other objective options.*
 - [OtherObjectiveOption](#) * [getOtherObjectiveOption](#) (int optionNumber)

Get one particular <other> objective option from the array of options.

 - [OtherObjectiveOption](#) ** [getAllOtherObjectiveOptions](#) ()

Get all <other> objective options.

 - [InitConValue](#) ** [getInitConValuesSparse](#) ()

Get the initial values associated with the constraints in sparse form.

- double * [getInitConValuesDense](#) ()
Get the initial values associated with the constraints in dense form.
- double * [getInitConValuesDense](#) (int numberOfConstraints)
Get the initial values associated with the constraints in dense form.
- [InitDualVarValue](#) ** [getInitDualVarValuesSparse](#) ()
Get the initial bounds associated with the dual variables in sparse form.
- double * [getInitDualVarLowerBoundsDense](#) ()
Get the initial dual variables associated with the lower bounds in dense form.
- double * [getInitDualVarLowerBoundsDense](#) (int numberOfConstraints)
Get the initial dual variables associated with the lower bounds in dense form.
- double * [getInitDualVarUpperBoundsDense](#) ()
Get the initial dual variables associated with the upper bounds in dense form.
- double * [getInitDualVarUpperBoundsDense](#) (int numberOfConstraints)
Get the initial dual variables associated with the upper bounds in dense form.
- int * [getSlackVariableInitialBasisStatusDense](#) (int numberOfConstraints)
Get the initial basis status for all slack variables in dense form.
- std::vector
 < [OtherConstraintOption](#) * > [getOtherConstraintOptions](#) (std::string solver_name)
 Get the array of other constraint options.
- [OtherConstraintOption](#) * [getOtherConstraintOption](#) (int optionNumber)
 Get one particular <other> constraint option from the array of options.
- [OtherConstraintOption](#) ** [getAllOtherConstraintOptions](#) ()
 Get all <other> constraint options.
- std::vector< [SolverOption](#) * > [getSolverOptions](#) (std::string solver_name)
 Get the options associated with a given solver.
- std::vector< [SolverOption](#) * > [getSolverOptions](#) (std::string solver_name, bool getFreeOptions)
 Get the options associated with a given solver AND options not associated with any solver (if desired)
- [SolverOption](#) ** [getAllSolverOptions](#) ()
 Get all solver options.
- bool [setServiceURI](#) (std::string serviceURI)
 Set the serviceURI.
- bool [setServiceName](#) (std::string serviceName)
 Set the service name.
- bool [setInstanceName](#) (std::string instanceName)
 Set the instance name.
- bool [setInstanceLocation](#) (std::string instanceLocation)
 Set the instance location.
- bool [setInstanceLocation](#) (std::string instanceLocation, std::string locationType)
 Alternative signature to set the instance location and location type simultaneously.
- bool [setInstanceLocationType](#) (std::string locationType)
 Set the instance location type.
- bool [setJobID](#) (std::string jobID)
 Set the job ID.
- bool [setSolverToInvoke](#) (std::string solverToInvoke)
 Set the solver to be invoked.
- bool [setLicense](#) (std::string license)
 Set the license information.

- bool [setUserName](#) (std::string userName)
Set the username.
- bool [setPassword](#) (std::string password)
Set the password.
- bool [setContact](#) (std::string contact)
Set the contact information.
- bool [setContact](#) (std::string contact, std::string transportType)
Alternative signature to set the contact information and transport type simultaneously.
- bool [setContactTransportType](#) (std::string transportType)
Set the transport type for contact.
- bool [setOtherGeneralOptions](#) (int numberOfOptions, [OtherOption](#) **other)
Set the other general options as an entire array.
- bool [setAnOtherGeneralOption](#) (std::string name, std::string value, std::string description)
Add another general option to the <other> option array.
- bool [setMinDiskSpace](#) (std::string unit, std::string description, double value)
Set the minimum disk space required for the current job.
- bool [setMinDiskSpace](#) (double value)
Alternate signature to set only the number of units.
- bool [setMinDiskSpaceUnit](#) (std::string unit)
- bool [setMinMemorySize](#) (std::string unit, std::string description, double value)
Set the minimum memory size required for the current job.
- bool [setMinMemorySize](#) (double value)
Alternate signature to set only the number of units.
- bool [setMinMemoryUnit](#) (std::string unit)
- bool [setMinCPUSpeed](#) (std::string unit, std::string description, double value)
Set the minimum CPU speed required for the current job.
- bool [setMinCPUSpeed](#) (double value)
Alternate signature to set only the number of units.
- bool [setMinCPUSpeedUnit](#) (std::string unit)
- bool [setMinCPUNumber](#) (int number, std::string description)
Set the minimum number of CPU cores required for the current job.
- bool [setMinCPUNumber](#) (int number)
Alternate signature to set only the number of cores.
- bool [setOtherSystemOptions](#) (int numberOfOptions, [OtherOption](#) **other)
- bool [setAnOtherSystemOption](#) (std::string name, std::string value, std::string description)
- bool [setServiceType](#) (std::string serviceType)
- bool [setOtherServiceOptions](#) (int numberOfOptions, [OtherOption](#) **other)
- bool [setAnOtherServiceOption](#) (std::string name, std::string value, std::string description)
- bool [setMaxTime](#) (double value, std::string unit)
- bool [setMaxTime](#) (double value)
- bool [setMaxTimeUnit](#) (std::string unit)
- bool [setRequestedStartTime](#) (std::string time)
- bool [setJobDependencies](#) (int numberOfDependencies, std::string *jobDependencies)
- bool [setAnotherJobDependency](#) (std::string jobId)
- bool [setRequiredDirectories](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherRequiredDirectory](#) (std::string path)
- bool [setRequiredFiles](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherRequiredFile](#) (std::string path)

- bool [setDirectoriesToMake](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherDirectoryToMake](#) (std::string path)
- bool [setFilesToMake](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherFileToMake](#) (std::string path)
- bool [setPathPairs](#) (int object, std::string *from, std::string *to, bool *makeCopy, int numberOfPathPairs)
 - setPathPairs set a number of path pairs into the [OSOption](#) object*
- bool [setInputDirectoriesToMove](#) (int numberOfPathPairs, [PathPair](#) **pathPair)
- bool [setAnotherInputDirectoryToMove](#) (std::string fromPath, std::string toPath, bool makeCopy)
- bool [setInputFilesToMove](#) (int numberOfPathPairs, [PathPair](#) **pathPair)
- bool [setAnotherInputFileToMove](#) (std::string fromPath, std::string toPath, bool makeCopy)
- bool [setOutputFilesToMove](#) (int numberOfPathPairs, [PathPair](#) **pathPair)
- bool [setAnotherOutputFileToMove](#) (std::string fromPath, std::string toPath, bool makeCopy)
- bool [setOutputDirectoriesToMove](#) (int numberOfPathPairs, [PathPair](#) **pathPair)
- bool [setAnotherOutputDirectoryToMove](#) (std::string fromPath, std::string toPath, bool makeCopy)
- bool [setFilesToDelete](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherFileToDelete](#) (std::string path)
- bool [setDirectoriesToDelete](#) (int numberOfPaths, std::string *paths)
- bool [setAnotherDirectoryToDelete](#) (std::string path)
- bool [setProcessesToKill](#) (int numberOfProcesses, std::string *processes)
- bool [setAnotherProcessToKill](#) (std::string process)
- bool [setOtherJobOptions](#) (int numberOfOptions, [OtherOption](#) **other)
- bool [setAnOtherJobOption](#) (std::string name, std::string value, std::string description)
- bool [setNumberOfVariables](#) (int numberOfVariables)
- bool [setNumberOfObjectives](#) (int numberOfObjectives)
- bool [setNumberOfConstraints](#) (int numberOfConstraints)
- bool [setInitVarValues](#) (int numberOfVar, int *idx, double *value, std::string *name)
- bool [setInitVarValuesSparse](#) (int numberOfVar, [InitVarValue](#) **var)
- bool [setInitVarValuesSparse](#) (int numberOfVar, [InitVarValue](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitVarValuesDense](#) (int numberOfVar, double *value)
- bool [setAnotherInitVarValue](#) (int idx, double value)
- bool [setInitVarValuesString](#) (int numberOfVar, int *idx, std::string *value, std::string *name)
- bool [setInitVarValuesStringSparse](#) (int numberOfVar, [InitVarValueString](#) **var)
- bool [setInitVarValuesStringSparse](#) (int numberOfVar, [InitVarValueString](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitVarValuesStringDense](#) (int numberOfVar, std::string *value)
- bool [setAnotherInitVarValueString](#) (int idx, std::string value)
- bool [setInitBasisStatus](#) (int object, int status, int *i, int ni)
- bool [setInitBasisStatusSparse](#) (int numberOfVar, [InitBasStatus](#) **var)
- bool [setInitBasisStatusSparse](#) (int numberOfVar, [InitBasStatus](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitBasisStatusDense](#) (int numberOfVar, std::string *var)
- bool [setAnotherInitBasisStatus](#) (int type, int idx, int status)
 - Set the basis status for another variable, objective or constraint/slack.*
- bool [setIntegerVariableBranchingWeights](#) (int numberOfVar, int *idx, double *value, std::string *name)
- bool [setIntegerVariableBranchingWeightsSparse](#) (int numberOfVar, [BranchingWeight](#) **var)
- bool [setIntegerVariableBranchingWeightsSparse](#) (int numberOfVar, [BranchingWeight](#) **var, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setIntegerVariableBranchingWeightsDense](#) (int numberOfVar, double *value)
- bool [setAnotherIntegerVariableBranchingWeight](#) (int idx, double value)
- bool [setSOSVariableBranchingWeights](#) (int numberOfSOS, [SOSWeights](#) **sos)
- bool [setAnotherSOSVariableBranchingWeight](#) (int sosIdx, int nvar, double weight, int *idx, double *value, std::string *name)

- bool [setNumberOfOtherVariableOptions](#) (int numberOfOther)
- bool [setOtherVariableOptions](#) (int numberOfVar, [OtherVariableOption](#) **var)
- bool [setAnOtherVariableOption](#) ([OtherVariableOption](#) *varOption)
- bool [setOtherVariableOptionAttributes](#) (int iOther, int numberOfVar, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string varType, std::string enumType, std::string description)

Set the attributes for one particular <other> <variable> option.
- bool [setOtherOptionEnumeration](#) (int object, int otherOptionNumber, int enumerationNumber, int numberOfEl, std::string value, std::string description, int *idxArray)

Set one enumeration associated with an <other> option in the <variables>, <objectives> or <constraints> element.
- bool [setOtherVariableOptionVar](#) (int otherOptionNumber, int varNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)

Set one element associated with an <other> option in the <variables> element.
- bool [setInitObjValues](#) (int numberOfObj, int *idx, double *value, std::string *name)
- bool [setInitObjValuesSparse](#) (int numberOfObj, [InitObjValue](#) **obj)
- bool [setInitObjValuesSparse](#) (int numberOfObj, [InitObjValue](#) **obj, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitObjValuesDense](#) (int numberOfObj, double *value)
- bool [setAnotherInitObjValue](#) (int idx, double value)
- bool [setInitObjBounds](#) (int numberOfObj, int *idx, double *lbValue, double *ubValue, std::string *name)
- bool [setInitObjBoundsSparse](#) (int numberOfObj, [InitObjBound](#) **obj)
- bool [setInitObjBoundsSparse](#) (int numberOfObj, [InitObjBound](#) **obj, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitObjBoundsDense](#) (int numberOfObj, double *lb, double *ub)
- bool [setAnotherInitObjBound](#) (int idx, double lbValue, double ubValue)
- bool [setNumberOfOtherObjectiveOptions](#) (int numberOfOther)
- bool [setOtherObjectiveOptions](#) (int numberOfObj, [OtherObjectiveOption](#) **obj)
- bool [setAnOtherObjectiveOption](#) ([OtherObjectiveOption](#) *objOption)
- bool [setOtherObjectiveOptionAttributes](#) (int iOther, int numberOfObj, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string objType, std::string enumType, std::string description)

Set the attributes for one particular <other> <objective> option.
- bool [setOtherObjectiveOptionObj](#) (int otherOptionNumber, int objNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)

Set one <obj> element associated with an <other> option in the <objectives> element.
- bool [setInitConValues](#) (int numberOfCon, int *idx, double *value, std::string *name)
- bool [setInitConValuesSparse](#) (int numberOfCon, [InitConValue](#) **con)
- bool [setInitConValuesSparse](#) (int numberOfCon, [InitConValue](#) **con, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitConValuesDense](#) (int numberOfCon, double *value)
- bool [setAnotherInitConValue](#) (int idx, double value)
- bool [setInitDualValues](#) (int numberOfCon, int *idx, double *lbValue, double *ubValue, std::string *name)
- bool [setInitDualVarValuesSparse](#) (int numberOfCon, [InitDualVarValue](#) **con)
- bool [setInitDualVarValuesSparse](#) (int numberOfCon, [InitDualVarValue](#) **con, [ENUM_COMBINE_ARRAYS](#) disp)
- bool [setInitDualVarValuesDense](#) (int numberOfCon, double *lb, double *ub)
- bool [setAnotherInitDualVarValue](#) (int idx, double lbValue, double ubValue)
- bool [setNumberOfOtherConstraintOptions](#) (int numberOfOther)
- bool [setOtherConstraintOptions](#) (int numberOfOptions, [OtherConstraintOption](#) **other)
- bool [setAnOtherConstraintOption](#) ([OtherConstraintOption](#) *optionValue)
- bool [setOtherConstraintOptionAttributes](#) (int iOther, int numberOfCon, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string conType, std::string enumType, std::string description)

Set the attributes for one particular <other> <constraint> option.

- bool [setOtherConstraintOptionCon](#) (int otherOptionNumber, int conNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)
Set one <con> element associated with an <other> option in the <constraints> element.
- bool [setNumberOfSolverOptions](#) (int numberOfOptions)
- bool [setSolverOptionContent](#) (int iOption, int numberOfItems, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string description, std::string *itemList)
Set the attributes for one particular solver option.
- bool [setSolverOptions](#) (int numberOfSolverOptions, [SolverOption](#) **solverOption)
- bool [setAnotherSolverOption](#) (std::string name, std::string value, std::string solver, std::string category, std::string type, std::string description)
- bool [setOptionInt](#) (std::string optionName, int optionValue)
- bool [setOptionStr](#) (std::string optionName, std::string optionValue)
- bool [setOptionDbf](#) (std::string optionName, double value)

Public Attributes

- [GeneralFileHeader](#) * [optionHeader](#)
OSOption has a header and five other children: general, system, service, job, and optimization.
- [GeneralOption](#) * [general](#)
generalOption holds the first child of the OSOption specified by the OSoL Schema.
- [SystemOption](#) * [system](#)
systemOption holds the second child of the OSOption specified by the OSoL Schema.
- [ServiceOption](#) * [service](#)
serviceOption holds the third child of the OSOption specified by the OSoL Schema.
- [JobOption](#) * [job](#)
jobOption holds the fourth child of the OSOption specified by the OSoL Schema.
- [OptimizationOption](#) * [optimization](#)
optimizationOption holds the fifth child of the OSOption specified by the OSoL Schema.

6.193.1 Detailed Description

The Option Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A class for holding all the solver options information.

Definition at line 3564 of file OSOption.h.

6.193.2 Constructor & Destructor Documentation

6.193.2.1 OSOption::OSOption ()

Default constructor.

6.193.2.2 OSOption::~~OSOption ()

Class destructor.

6.193.3 Member Function Documentation

6.193.3.1 bool OSOption::setHeader (std::string *name*, std::string *source*, std::string *description*, std::string *fileCreator*, std::string *licence*)

A function to populate an instance of the option header element.

Parameters

<i>name</i> ,:	the name of this file or instance
<i>source</i> ,:	the source (e.g., in BiBTeX format)
<i>fileCreator</i> ,:	the creator of this file
<i>description</i> ,:	further description about this file and/or its contents
<i>licence</i> ,:	licence information if applicable

6.193.3.2 bool OSOption::isEqual (OSOption * *that*)

A function to check for the equality of two objects.

6.193.3.3 bool OSOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.193.3.4 bool OSOption::deepCopyFrom (OSOption * *that*)

A function to make a deep copy of an [OSOption](#) object.

Parameters

<i>that</i> ,:	the OSOption object from which information is to be copied
----------------	--

Returns

whether the copy was created successfully

6.193.3.5 std::string OSOption::getFileName ()

Get the name of the file.

6.193.3.6 `std::string OSOption::getFileSource ()`

Get the source of the file or problem.

6.193.3.7 `std::string OSOption::getFileDescription ()`

Get a description for the file or problem.

6.193.3.8 `std::string OSOption::getFileCreator ()`

Get the name of the person who created the file.

6.193.3.9 `std::string OSOption::getFileLicence ()`

Get licence information associated with the file.

6.193.3.10 `std::string OSOption::getServiceURI ()`

Get the service URI.

6.193.3.11 `std::string OSOption::getServiceName ()`

Get the service name.

6.193.3.12 `std::string OSOption::getInstanceName ()`

Get the instance name.

6.193.3.13 `std::string OSOption::getInstanceLocation ()`

Get the instance location.

6.193.3.14 `std::string OSOption::getInstanceLocationType ()`

Get the location type.

6.193.3.15 `std::string OSOption::getJobID ()`

Get the job ID.

6.193.3.16 `std::string OSOption::getSolverToInvoke ()`

Get the solver name.

6.193.3.17 `std::string OSOption::getLicense ()`

Get the license string.

6.193.3.18 `std::string OSOption::getUserName ()`

Get the user name.

6.193.3.19 `std::string OSOption::getPassword ()`

Get the password.

6.193.3.20 `std::string OSOption::getContact ()`

Get the contact information.

6.193.3.21 `std::string OSOption::getContactTransportType ()`

Get the transport type.

6.193.3.22 `std::string OSOption::getMinDiskSpaceUnit ()`

Get the disk space unit.

6.193.3.23 `std::string OSOption::getMinDiskSpaceDescription ()`

get the disk space description

6.193.3.24 `std::string OSOption::getMinMemoryUnit ()`

Get the memory unit.

6.193.3.25 `std::string OSOption::getMinMemoryDescription ()`

get the memory description

6.193.3.26 `std::string OSOption::getMinCPUSpeedUnit ()`

Get the CPU speed unit.

6.193.3.27 `std::string OSOption::getMinCPUSpeedDescription ()`

Get the CPU speed description.

6.193.3.28 `std::string OSOption::getMinCPUNumberDescription ()`

Get the CPU description.

6.193.3.29 `std::string OSOption::getServiceType ()`

Get the service type.

6.193.3.30 `std::string OSOption::getMaxTimeUnit ()`

Get the time unit.

6.193.3.31 `std::string OSOption::getRequestedStartTime ()`

Get the requested starting time.

6.193.3.32 `std::string OSOption::getOptionStr (std::string optionName)`

Get any of the string-valued options.

6.193.3.33 `double OSOption::getMinDiskSpace ()`

Get the minimum required disk space.

6.193.3.34 double OOption::getMinMemorySize ()

Get the minimum required memory.

6.193.3.35 double OOption::getMinCPUSpeed ()

Get the minimum required CPU speed.

6.193.3.36 double OOption::getMaxTime ()

Get the maximum allowed time.

6.193.3.37 double OOption::getOptionDbl (std::string *optionName*)

Get any of the double-valued options.

6.193.3.38 int OOption::getMinCPUNumber ()

Get the minimum required number of CPUs.

6.193.3.39 int OOption::getNumberOfOtherGeneralOptions ()

Get the number of <other> options in the <general> element.

6.193.3.40 int OOption::getNumberOfOtherSystemOptions ()

Get the number of <other> options in the <system> element.

6.193.3.41 int OOption::getNumberOfOtherServiceOptions ()

Get the number of <other> options in the <service> element.

6.193.3.42 int OOption::getNumberOfOtherJobOptions ()

Get the number of <other> options in the <job> element.

6.193.3.43 int OOption::getNumberOfJobDependencies ()

Get the number of job dependencies.

6.193.3.44 int OOption::getNumberOfRequiredDirectories ()

Get the number of required directories.

6.193.3.45 int OOption::getNumberOfRequiredFiles ()

Get the number of required files.

6.193.3.46 int OOption::getNumberOfDirectoriesToMake ()

Get the number of directories to make.

6.193.3.47 int OOption::getNumberOfFilesToMake ()

Get the number of files to make.

6.193.3.48 `int OSOption::getNumberOfInputDirectoriesToMove ()`

Get the number of input directories to move.

6.193.3.49 `int OSOption::getNumberOfInputFilesToMove ()`

Get the number of input files to move.

6.193.3.50 `int OSOption::getNumberOfOutputDirectoriesToMove ()`

Get the number of output directories to move.

6.193.3.51 `int OSOption::getNumberOfOutputFilesToMove ()`

Get the number of output files to move.

6.193.3.52 `int OSOption::getNumberOfFilesToDelete ()`

Get the number of files to delete.

6.193.3.53 `int OSOption::getNumberOfDirectoriesToDelete ()`

Get the number of directories to delete.

6.193.3.54 `int OSOption::getNumberOfProcessesToKill ()`

Get the number of processes to kill.

6.193.3.55 `int OSOption::getNumberOfVariables ()`

Get the number of variables in the instance.

6.193.3.56 `int OSOption::getNumberOfObjectives ()`

Get the number of objectives in the instance.

6.193.3.57 `int OSOption::getNumberOfConstraints ()`

Get the number of constraints in the instance.

6.193.3.58 `int OSOption::getNumberOfInitVarValues ()`

Get the number of initial variable values.

Returns

the number of initial variable values.

6.193.3.59 `int OSOption::getNumberOfInitVarValuesString ()`

Get the number of initial variable strings.

Returns

the number of initial variable strings.

6.193.3.60 int OSOption::getNumberOfIntegerVariableBranchingWeights ()

Get the number of variables for which integer branching weights are provided.

Returns

the number of variables.

6.193.3.61 int OSOption::getNumberOfSOS ()

Get the number of special ordered sets for which branching weights are provided.

Returns

the number of variables.

6.193.3.62 int OSOption::getNumberOfSOSVarBranchingWeights (int iSOS)

Get the number of variables for which branching weights are provided in a particular SOS.

Parameters

<i>iSOS</i>	the number of the SOS
-------------	-----------------------

Returns

the number of variables.

6.193.3.63 int OSOption::getNumberOfOtherVariableOptions ()

Get the number of other variable options.

Returns

the number of other variable options.

6.193.3.64 int OSOption::getNumberOfInitObjValues ()

Get the number of initial objective values.

Returns

the number of initial objective values.

6.193.3.65 int OSOption::getNumberOfInitObjBounds ()

Get the number of initial objective bounds.

Returns

the number of initial objective bound values.

6.193.3.66 int OSOption::getNumberOfOtherObjectiveOptions ()

Get the number of other objective options.

Returns

the number of other objective options.

6.193.3.67 int OSOption::getNumberOfInitConValues ()

Get the number of initial constraint values.

Returns

the number of initial constraint values.

6.193.3.68 int OSOption::getNumberOfInitDualVarValues ()

Get the number of initial dual variable values.

Returns

the number of initial dual variable values.

6.193.3.69 int OSOption::getNumberOfOtherConstraintOptions ()

Get the number of other constraint options.

Returns

the number of other constraint options.

6.193.3.70 int OSOption::getNumberOfSolverOptions ()

Get the number of solver options.

Returns

the number of solver options.

6.193.3.71 int OSOption::getOptionInt (std::string *optionName*)

Get any of the integer-valued options.

6.193.3.72 OtherOption OSOption::getOtherGeneralOptions ()**

Get the array of other options associated with the <general> element.

Returns

a vector of pointers to otherOptions objects associated with the <general> element

6.193.3.73 OtherOption OSOption::getOtherSystemOptions ()**

Get the array of other options associated with the <system> element.

Returns

a vector of pointers to otherOptions objects associated with the <system> element

6.193.3.74 OtherOption OOption::getOtherServiceOptions ()**

Get the array of other options associated with the <service> element.

Returns

a vector of pointers to otherOptions objects associated with the <service> element

6.193.3.75 OtherOption OOption::getOtherJobOptions ()**

Get the array of other options associated with the <job> element.

Returns

a vector of pointers to otherOptions objects associated with the <job> element

6.193.3.76 OtherOption OOption::getOtherOptions (std::string *elementName*)**

Get the array of other options associated with any element.

Returns

a vector of pointers to otherOptions objects associated with the element whose name matches elementName

6.193.3.77 OtherOption OOption::getAllOtherOptions ()**

Get the array of all other options associated with the <general>, <system>, <service> and <job> elements.

Returns

a vector of pointers to all otherOptions objects

6.193.3.78 std::string* OOption::getJobDependencies ()

Get the array of job dependencies.

Returns

a vector of pointers to [JobDependencies](#) objects

6.193.3.79 std::string* OOption::getRequiredDirectories ()

Get the array of required directories.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the directories that are required by the current job

6.193.3.80 std::string* OOption::getRequiredFiles ()

Get the array of required files.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the files that are required by the current job

6.193.3.81 `std::string* OSOption::getDirectoriesToMake ()`

Get the array of directories to make.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the directories that must be created

6.193.3.82 `std::string* OSOption::getFilesToMake ()`

Get the array of files to make.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the files that must be created

6.193.3.83 `PathPair** OSOption::getInputDirectoriesToMove ()`

Get the array of input directories to move.

Returns

a vector of pointers to [PathPair](#) objects giving the input directories that must be moved

6.193.3.84 `PathPair** OSOption::getInputFilesToMove ()`

Get the array of input files to move.

Returns

a vector of pointers to [PathPair](#) objects giving the input files that must be moved

6.193.3.85 `PathPair** OSOption::getOutputDirectoriesToMove ()`

Get the array of output directories to move.

Returns

a vector of pointers to [PathPair](#) objects giving the output directories that must be moved

6.193.3.86 `PathPair** OSOption::getOutputFilesToMove ()`

Get the array of output files to move.

Returns

a vector of pointers to [PathPair](#) objects giving the output files that must be moved

6.193.3.87 `std::string* OSOption::getDirectoriesToDelete ()`

Get the array of directories to delete.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the directories that must be deleted

6.193.3.88 `std::string* OSOption::getFilesToDelete ()`

Get the array of files to delete.

Returns

a vector of pointers to [DirectoriesAndFiles](#) objects giving the files that must be deleted

6.193.3.89 `std::string* OSOption::getProcessesToKill ()`

Get the array of processes to kill.

Returns

a vector of pointers to [Processes](#) objects giving the processes that must be killed

6.193.3.90 `InitVarValue** OSOption::getInitVarValuesSparse ()`

Get the initial values associated with the variables in sparse form.

Returns

a vector of pointers to [InitVarValue](#) objects that hold initial values for (some of) the variables

6.193.3.91 `double* OSOption::getInitVarValuesDense ()`

Get the initial values associated with the variables in dense form.

Returns

a vector of double that holds initial values (or [OSNaN\(\)](#)) for all of the variables

6.193.3.92 `double* OSOption::getInitVarValuesDense (int numberOfVariables)`

Get the initial values associated with the variables in dense form.

Parameters

<i>numberOfVariables</i>	holds the dimension of the vector
--------------------------	-----------------------------------

Returns

a vector of double that holds initial values (or [OSNaN\(\)](#)) for all of the variables

6.193.3.93 `InitVarValueString** OSOption::getInitVarValuesStringSparse ()`

Get the initial value strings associated with the variables in sparse form.

Returns

a vector of pointers to [InitVarValueString](#) objects that hold initial value strings for (some of) the variables

6.193.3.94 `std::string* OSOption::getInitVarValuesStringDense ()`

Get the initial value strings associated with the variables in dense form.

Returns

a vector of strings that holds initial value strings (or "") for all of the variables

6.193.3.95 `std::string* OSOption::getInitVarValuesStringDense (int numberOfVariables)`

Get the initial value strings associated with the variables in dense form.

Parameters

<i>numberOfVariables</i>	holds the dimension of the vector
--------------------------	-----------------------------------

Returns

a vector of strings that holds initial value strings (or "") for all of the variables

6.193.3.96 `InitBasStatus** OSOption::getInitBasisStatusSparse ()`

Get the initial basis status in sparse form.

Returns

a vector of pointers to [InitBasStatus](#) objects that hold initial basis status for (some of) the variables

6.193.3.97 `std::string* OSOption::getInitBasisStatusDense ()`

Get the initial basis information in dense form.

Returns

a vector of strings that holds initial basis status (or "unknown") for all of the variables

6.193.3.98 `int* OSOption::getVariableInitialBasisStatusDense (int numberOfVariables)`

Get the initial basis status for all variables in dense form.

Returns

an array of int, with values corresponding to ENUM_BASIS_STATUS – see [OSGeneral.h](#))

Note

returns ENUM_BASIS_STATUS_unknown for variables that are not initialed

Parameters

<i>numberOfVariables</i>	is the dimension of the array
--------------------------	-------------------------------

6.193.3.99 `int OSOption::getNumberOfInitialBasisElements (int type, int status)`

Get the number of initial basis elements for a particular variable type and basis status.

Returns

the number of elements

Parameters

<i>type</i> ,:	the type of variable or problem component (contained in ENUM_PROBLEM_COMPONENT — see OSGeneral.h)
<i>status</i> ,:	the basis status (contained in ENUM_BASIS_STATUS — see OSGeneral.h)

6.193.3.100 `bool OSOption::getInitialBasisElements (int type, int status, int * elem)`

Get the initial basis elements for a particular variable type and basis status.

Returns

whether the operation was successful or not

Parameters

<i>type</i> ,:	the type of variable or problem component (contained in ENUM_PROBLEM_COMPONENT — see OSGeneral.h)
<i>status</i> ,:	the basis status (contained in ENUM_BASIS_STATUS — see OSGeneral.h)
<i>elem</i> ,:	pointer to the memory location where the user wants to store the returned values

6.193.3.101 `BranchingWeight** OSOption::getIntegerVariableBranchingWeightsSparse ()`

Get the integer branching weights in sparse form.

Returns

a vector of pointers to [BranchingWeight](#) objects that hold branching weights for (some of) the variables

6.193.3.102 `double* OSOption::getIntegerVariableBranchingWeightsDense ()`

Get the integer branching weights in dense form.

Returns

a vector of double that holds branching weights (or [OSNaN\(\)](#)) for all the variables

6.193.3.103 `double* OSOption::getIntegerVariableBranchingWeightsDense (int numberOfVariables)`

Get the integer branching weights in dense form.

Parameters

<i>numberOfVariables</i>	holds the dimension of the vector
--------------------------	-----------------------------------

Returns

a vector of double that holds branching weights (or [OSNaN\(\)](#)) for all the variables

6.193.3.104 SOSWeights OOption::getSOSVariableBranchingWeightsSparse ()**

Get the SOS branching weights in sparse form.

Returns

a vector of pointers to [SOSWeights](#) objects that hold branching weights for (some of) the variables contained in special ordered sets

6.193.3.105 std::vector<OtherVariableOption*> OOption::getOtherVariableOptions (std::string solver_name)

Get the <other> variable options associated with a particular solver.

Parameters

<i>solver_name</i>	is the name of the solver whose options we want
--------------------	---

Returns

a vector of pointers to [OtherVariableOption](#) objects that correspond to the solver named.

6.193.3.106 OtherVariableOption* OOption::getOtherVariableOption (int optionNumber)

Get one particular <other> variable option from the array of options.

Parameters

<i>optionNumber</i>	is the index of the option in the array
---------------------	---

Returns

a pointer to one [OtherVariableOption](#) object

6.193.3.107 OtherVariableOption OOption::getAllOtherVariableOptions ()**

Get all <other> variable options.

Returns

a pointer to an array of [OtherVariableOption](#) objects

6.193.3.108 InitObjValue OOption::getInitObjValuesSparse ()**

Get the initial values associated with the objectives in sparse form.

Returns

a vector of pointers to [InitObjValue](#) objects that hold initial values for (some of) the objectives

6.193.3.109 `double* OSOption::getInitObjValuesDense ()`

Get the initial values associated with the objectives in dense form.

Returns

a vector of double that hold initial values (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.110 `double* OSOption::getInitObjValuesDense (int numberOfObjectives)`

Get the initial values associated with the objectives in dense form.

Parameters

<i>numberOfObjectives</i>	holds the dimension of the vector
---------------------------	-----------------------------------

Returns

a vector of double that hold initial values (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.111 `InitObjBound** OSOption::getInitObjBoundsSparse ()`

Get the initial bounds associated with the objectives in sparse form.

Returns

a vector of pointers to [InitObjBound](#) objects that hold initial bounds for (some of) the objectives

6.193.3.112 `double* OSOption::getInitObjLowerBoundsDense ()`

Get the initial lower bounds associated with the objectives in dense form.

Returns

a vector of double that hold initial lower bounds (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.113 `double* OSOption::getInitObjLowerBoundsDense (int numberOfObjectives)`

Get the initial lower bounds associated with the objectives in dense form.

Parameters

<i>numberOfObjectives</i>	holds the dimension of the vector
---------------------------	-----------------------------------

Returns

a vector of double that hold initial lower bounds (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.114 `double* OSOption::getInitObjUpperBoundsDense ()`

Get the initial upper bounds associated with the objectives in dense form.

Returns

a vector of double that hold initial upper bounds (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.115 `double* OSOption::getInitObjUpperBoundsDense (int numberOfObjectives)`

Get the initial upper bounds associated with the objectives in dense form.

Parameters

<i>numberOfObjectives</i>	holds the dimension of the vector
---------------------------	-----------------------------------

Returns

a vector of double that hold initial upper bounds (or [OSNaN\(\)](#)) for all of the objectives

6.193.3.116 `int* OSOption::getObjectiveInitialBasisStatusDense (int numberOfObjectives)`

Get the initial basis status for all objectives in dense form.

Returns

an array of int, with values corresponding to ENUM_BASIS_STATUS – see [OSGeneral.h](#))

Note

returns ENUM_BASIS_STATUS_unknown for objectives that are not initialed

Parameters

<i>numberOfObjectives</i>	is the dimension of the array
---------------------------	-------------------------------

6.193.3.117 `std::vector<OtherObjectiveOption*> OSOption::getOtherObjectiveOptions (std::string solver_name)`

Get the array of other objective options.

Parameters

<i>solver_name</i>	is the name of the solver whose options we want
--------------------	---

Returns

a vector of pointers to [OtherConstraintOption](#) objects

6.193.3.118 `OtherObjectiveOption* OSOption::getOtherObjectiveOption (int optionNumber)`

Get one particular <other> objective option from the array of options.

Parameters

<i>optionNumber</i>	is the index of the option in the array
---------------------	---

Returns

a pointer to one [OtherObjectiveOption](#) object

6.193.3.119 OtherObjectiveOption OOption::getAllOtherObjectiveOptions ()**

Get all <other> objective options.

Returns

a pointer to an array of [OtherObjectiveOption](#) objects

6.193.3.120 InitConValue OOption::getInitConValuesSparse ()**

Get the initial values associated with the constraints in sparse form.

Returns

a vector of pointers to [InitConValue](#) objects that hold initial values for (some of) the constraints

6.193.3.121 double* OOption::getInitConValuesDense ()

Get the initial values associated with the constraints in dense form.

Returns

a vector of double that hold initial values for all of the constraints

6.193.3.122 double* OOption::getInitConValuesDense (int *numberOfConstraints*)

Get the initial values associated with the constraints in dense form.

Parameters

<i>numberOfConstraints</i>	holds the dimension of the vector
----------------------------	-----------------------------------

Returns

a vector of double that hold initial values for all of the constraints

6.193.3.123 InitDualVarValue OOption::getInitDualVarValuesSparse ()**

Get the initial bounds associated with the dual variables in sparse form.

Returns

a vector of pointers to [InitDualVarValue](#) objects that hold initial bounds for (some of) the dual variables

6.193.3.124 double* OOption::getInitDualVarLowerBoundsDense ()

Get the initial dual variables associated with the lower bounds in dense form.

Returns

a vector of double that hold initial lower bounds for all of the dual variables

6.193.3.125 `double* OSOption::getInitDualVarLowerBoundsDense (int numberOfConstraints)`

Get the initial dual variables associated with the lower bounds in dense form.

Parameters

<i>numberOfConstraints</i>	holds the dimension of the vector
----------------------------	-----------------------------------

Returns

a vector of double that hold initial lower bounds for all of the dual variables

6.193.3.126 `double* OSOption::getInitDualVarUpperBoundsDense ()`

Get the initial dual variables associated with the upper bounds in dense form.

Returns

a vector of double that hold initial upper bounds for all of the dual variables

6.193.3.127 `double* OSOption::getInitDualVarUpperBoundsDense (int numberOfConstraints)`

Get the initial dual variables associated with the upper bounds in dense form.

Parameters

<i>numberOfConstraints</i>	holds the dimension of the vector
----------------------------	-----------------------------------

Returns

a vector of double that hold initial upper bounds for all of the dual variables

6.193.3.128 `int* OSOption::getSlackVariableInitialBasisStatusDense (int numberOfConstraints)`

Get the initial basis status for all slack variables in dense form.

Returns

an array of int, with values corresponding to ENUM_BASIS_STATUS – see [OSGeneral.h](#))

Note

returns ENUM_BASIS_STATUS_unknown for slack variables that are not initialed

Parameters

<i>numberOfConstraints</i>	is the dimension of the array
----------------------------	-------------------------------

6.193.3.129 `std::vector<OtherConstraintOption*> OSOption::getOtherConstraintOptions (std::string solver_name)`

Get the array of other constraint options.

Parameters

<i>solver_name</i>	is the name of the solver whose options we want
--------------------	---

Returns

a vector of pointers to [OtherConstraintOption](#) objects

6.193.3.130 `OtherConstraintOption* OSOption::getOtherConstraintOption (int optionNumber)`

Get one particular <other> constraint option from the array of options.

Parameters

<i>optionNumber</i>	is the index of the option in the array
---------------------	---

Returns

a pointer to one [OtherConstraintOption](#) object

6.193.3.131 `OtherConstraintOption** OSOption::getAllOtherConstraintOptions ()`

Get all <other> constraint options.

Returns

a pointer to an array of [OtherConstraintOption](#) objects

6.193.3.132 `std::vector<SolverOption*> OSOption::getSolverOptions (std::string solver_name)`

Get the options associated with a given solver.

Parameters

<i>solver_name</i>	is the name of the solver whose options we want
--------------------	---

Returns

a vector of pointers to [SolverOption](#) objects that correspond to the solver named.

6.193.3.133 `std::vector<SolverOption*> OSOption::getSolverOptions (std::string solver_name, bool getFreeOptions)`

Get the options associated with a given solver AND options not associated with any solver (if desired)

Parameters

<i>solver_name</i>	is the name of the solver whose options we want
<i>getFreeOptions</i>	is a boolean set to true if the free options (not associated with a solver name) should be returned

Returns

a vector of pointers to [SolverOption](#) objects that correspond to the solver named.

6.193.3.134 SolverOption OSOption::getAllSolverOptions ()**

Get all solver options.

Returns

a pointer to an array [SolverOption](#) objects

6.193.3.135 bool OSOption::setServiceURI (std::string *serviceURI*)

Set the serviceURI.

6.193.3.136 bool OSOption::setServiceName (std::string *serviceName*)

Set the service name.

6.193.3.137 bool OSOption::setInstanceName (std::string *instanceName*)

Set the instance name.

6.193.3.138 bool OSOption::setInstanceLocation (std::string *instanceLocation*)

Set the instance location.

6.193.3.139 bool OSOption::setInstanceLocation (std::string *instanceLocation*, std::string *locationType*)

Alternative signature to set the instance location and location type simultaneously.

6.193.3.140 bool OSOption::setInstanceLocationType (std::string *locationType*)

Set the instance location type.

6.193.3.141 bool OSOption::setJobID (std::string *jobID*)

Set the job ID.

6.193.3.142 bool OSOption::setSolverToInvoke (std::string *solverToInvoke*)

Set the solver to be invoked.

6.193.3.143 bool OSOption::setLicense (std::string *license*)

Set the license information.

6.193.3.144 bool OSOption::setUserName (std::string *userName*)

Set the username.

6.193.3.145 bool OSOption::setPassword (std::string *password*)

Set the password.

6.193.3.146 `bool OSOption::setContact (std::string contact)`

Set the contact information.

6.193.3.147 `bool OSOption::setContact (std::string contact, std::string transportType)`

Alternative signature to set the contact information and transport type simultaneously.

6.193.3.148 `bool OSOption::setContactTransportType (std::string transportType)`

Set the transport type for contact.

6.193.3.149 `bool OSOption::setOtherGeneralOptions (int numberOfOptions, OtherOption ** other)`

Set the other general options as an entire array.

Parameters

<i>numberOfOptions</i>	contains the number of other options to be set
<i>other</i>	is a pointer to an array of OtherOption objects

6.193.3.150 `bool OSOption::setAnOtherGeneralOption (std::string name, std::string value, std::string description)`

Add another general option to the <other> option array.

Parameters

<i>name</i>	- the identifying anme of the option. This string cannot be empty
<i>value</i>	- optional value associated with this option
<i>description</i>	- further information (can be used for documentation)

6.193.3.151 `bool OSOption::setMinDiskSpace (std::string unit, std::string description, double value)`

Set the minimum disk space required for the current job.

Parameters

<i>unit</i>	- select the unit (Kb, Mb, etc.)
<i>description</i>	- further description (can be used for documentation)
<i>value</i>	- number of units of disk space required

6.193.3.152 `bool OSOption::setMinDiskSpace (double value)`

Alternate signature to set only the number of units.

6.193.3.153 `bool OSOption::setMinDiskSpaceUnit (std::string unit)`

6.193.3.154 `bool OSOption::setMinMemorySize (std::string unit, std::string description, double value)`

Set the minimum memory size required for the current job.

Parameters

<i>unit</i>	- select the unit (Kb, Mb, etc.)
<i>description</i>	- further description (can be used for documentation)

<i>value</i>	- number of units of memory size required
--------------	---

6.193.3.155 `bool OSOption::setMinMemorySize (double value)`

Alternate signature to set only the number of units.

6.193.3.156 `bool OSOption::setMinMemoryUnit (std::string unit)`

6.193.3.157 `bool OSOption::setMinCPUSpeed (std::string unit, std::string description, double value)`

Set the minimum CPU speed required for the current job.

Parameters

<i>unit</i>	- select the unit (MHz, GHz, TFlops etc.)
<i>description</i>	- further description (can be used for documentation)
<i>value</i>	- number of units of CPU speed required

6.193.3.158 `bool OSOption::setMinCPUSpeed (double value)`

Alternate signature to set only the number of units.

6.193.3.159 `bool OSOption::setMinCPUSpeedUnit (std::string unit)`

6.193.3.160 `bool OSOption::setMinCPUNumber (int number, std::string description)`

Set the minimum number of CPU cores required for the current job.

Parameters

<i>number</i>	- number of CPU cores required
<i>description</i>	- further description (can be used for documentation)

6.193.3.161 `bool OSOption::setMinCPUNumber (int number)`

Alternate signature to set only the number of cores.

6.193.3.162 `bool OSOption::setOtherSystemOptions (int numberOfOptions, OtherOption ** other)`

6.193.3.163 `bool OSOption::setAnOtherSystemOption (std::string name, std::string value, std::string description)`

6.193.3.164 `bool OSOption::setServiceType (std::string serviceType)`

6.193.3.165 `bool OSOption::setOtherServiceOptions (int numberOfOptions, OtherOption ** other)`

6.193.3.166 `bool OSOption::setAnOtherServiceOption (std::string name, std::string value, std::string description)`

6.193.3.167 `bool OSOption::setMaxTime (double value, std::string unit)`

6.193.3.168 `bool OSOption::setMaxTime (double value)`

6.193.3.169 `bool OSOption::setMaxTimeUnit (std::string unit)`

6.193.3.170 `bool OSOption::setRequestedStartTime (std::string time)`

- 6.193.3.171 `bool OSOption::setJobDependencies (int numberOfDependencies, std::string * jobDependencies)`
- 6.193.3.172 `bool OSOption::setAnotherJobDependency (std::string jobID)`
- 6.193.3.173 `bool OSOption::setRequiredDirectories (int numberOfPaths, std::string * paths)`
- 6.193.3.174 `bool OSOption::setAnotherRequiredDirectory (std::string path)`
- 6.193.3.175 `bool OSOption::setRequiredFiles (int numberOfPaths, std::string * paths)`
- 6.193.3.176 `bool OSOption::setAnotherRequiredFile (std::string path)`
- 6.193.3.177 `bool OSOption::setDirectoriesToMake (int numberOfPaths, std::string * paths)`
- 6.193.3.178 `bool OSOption::setAnotherDirectoryToMake (std::string path)`
- 6.193.3.179 `bool OSOption::setFilesToMake (int numberOfPaths, std::string * paths)`
- 6.193.3.180 `bool OSOption::setAnotherFileToMake (std::string path)`
- 6.193.3.181 `bool OSOption::setPathPairs (int object, std::string * from, std::string * to, bool * makeCopy, int numberOfPathPairs)`

setPathPairs set a number of path pairs into the [OSOption](#) object

Parameters

<i>object</i>	describes the type of pathpairs legal values are ENUM_PATHPAIR_input_dir, ENUM_PATHPAIR_input_file, ENUM_PATHPAIR_output_file, ENUM_PATHPAIR_output_dir
<i>from</i>	is a pointer to an array of strings containing the location of the original object
<i>to</i>	is a pointer to an array of strings containing the location of the destination object
<i>makeCopy</i>	is a pointer to an array of boolean, describing for each object whether it is to be copied or moved
<i>numberOfPathPairs</i>	is an integer giving the number of PathPairs this must equal the number of entries in the from, to and makeCopy arrays

- 6.193.3.182 `bool OSOption::setInputDirectoriesToMove (int numberOfPathPairs, PathPair ** pathPair)`
- 6.193.3.183 `bool OSOption::setAnotherInputDirectoryToMove (std::string fromPath, std::string toPath, bool makeCopy)`
- 6.193.3.184 `bool OSOption::setInputFilesToMove (int numberOfPathPairs, PathPair ** pathPair)`
- 6.193.3.185 `bool OSOption::setAnotherInputFileToMove (std::string fromPath, std::string toPath, bool makeCopy)`
- 6.193.3.186 `bool OSOption::setOutputFilesToMove (int numberOfPathPairs, PathPair ** pathPair)`
- 6.193.3.187 `bool OSOption::setAnotherOutputFileToMove (std::string fromPath, std::string toPath, bool makeCopy)`
- 6.193.3.188 `bool OSOption::setOutputDirectoriesToMove (int numberOfPathPairs, PathPair ** pathPair)`
- 6.193.3.189 `bool OSOption::setAnotherOutputDirectoryToMove (std::string fromPath, std::string toPath, bool makeCopy)`
- 6.193.3.190 `bool OSOption::setFilesToDelete (int numberOfPaths, std::string * paths)`
- 6.193.3.191 `bool OSOption::setAnotherFileToDelete (std::string path)`
- 6.193.3.192 `bool OSOption::setDirectoriesToDelete (int numberOfPaths, std::string * paths)`

- 6.193.3.193 `bool OSOption::setAnotherDirectoryToDelete (std::string path)`
- 6.193.3.194 `bool OSOption::setProcessesToKill (int numberOfProcesses, std::string * processes)`
- 6.193.3.195 `bool OSOption::setAnotherProcessToKill (std::string process)`
- 6.193.3.196 `bool OSOption::setOtherJobOptions (int numberOfOptions, OtherOption ** other)`
- 6.193.3.197 `bool OSOption::setAnOtherJobOption (std::string name, std::string value, std::string description)`
- 6.193.3.198 `bool OSOption::setNumberOfVariables (int numberOfVariables)`
- 6.193.3.199 `bool OSOption::setNumberOfObjectives (int numberOfObjectives)`
- 6.193.3.200 `bool OSOption::setNumberOfConstraints (int numberOfConstraints)`
- 6.193.3.201 `bool OSOption::setInitVarValues (int numberOfVar, int * idx, double * value, std::string * name)`
- 6.193.3.202 `bool OSOption::setInitVarValuesSparse (int numberOfVar, InitVarValue ** var)`
- 6.193.3.203 `bool OSOption::setInitVarValuesSparse (int numberOfVar, InitVarValue ** var, ENUM_COMBINE_ARRAYS disp)`
- 6.193.3.204 `bool OSOption::setInitVarValuesDense (int numberOfVar, double * value)`
- 6.193.3.205 `bool OSOption::setAnotherInitVarValue (int idx, double value)`
- 6.193.3.206 `bool OSOption::setInitVarValuesString (int numberOfVar, int * idx, std::string * value, std::string * name)`
- 6.193.3.207 `bool OSOption::setInitVarValuesStringSparse (int numberOfVar, InitVarValueString ** var)`
- 6.193.3.208 `bool OSOption::setInitVarValuesStringSparse (int numberOfVar, InitVarValueString ** var, ENUM_COMBINE_ARRAYS disp)`
- 6.193.3.209 `bool OSOption::setInitVarValuesStringDense (int numberOfVar, std::string * value)`
- 6.193.3.210 `bool OSOption::setAnotherInitVarValueString (int idx, std::string value)`
- 6.193.3.211 `bool OSOption::setInitBasisStatus (int object, int status, int * i, int ni)`
- 6.193.3.212 `bool OSOption::setInitBasisStatusSparse (int numberOfVar, InitBasStatus ** var)`
- 6.193.3.213 `bool OSOption::setInitBasisStatusSparse (int numberOfVar, InitBasStatus ** var, ENUM_COMBINE_ARRAYS disp)`
- 6.193.3.214 `bool OSOption::setInitBasisStatusDense (int numberOfVar, std::string * var)`
- 6.193.3.215 `bool OSOption::setAnotherInitBasisStatus (int type, int idx, int status)`

Set the basis status for another variable, objective or constraint/slack.

Parameters

<i>type</i> ,:	type of this element (see ENUM_PROBLEM_COMPONENT - OSGeneral.h)
<i>idx</i> ,:	index of this element (nonnegative for variable or constraint, negative for objective)
<i>status</i> ,:	basis status (see ENUM_BASIS_STATUS - OSGeneral.h)

- 6.193.3.216 `bool OSOption::setIntegerVariableBranchingWeights (int numberOfVar, int * idx, double * value, std::string * name)`
- 6.193.3.217 `bool OSOption::setIntegerVariableBranchingWeightsSparse (int numberOfVar, BranchingWeight ** var)`
- 6.193.3.218 `bool OSOption::setIntegerVariableBranchingWeightsSparse (int numberOfVar, BranchingWeight ** var, ENUM_COMBINE_ARRAYS disp)`
- 6.193.3.219 `bool OSOption::setIntegerVariableBranchingWeightsDense (int numberOfVar, double * value)`
- 6.193.3.220 `bool OSOption::setAnotherIntegerVariableBranchingWeight (int idx, double value)`
- 6.193.3.221 `bool OSOption::setSOSVariableBranchingWeights (int numberOfSOS, SOSWeights ** sos)`
- 6.193.3.222 `bool OSOption::setAnotherSOSVariableBranchingWeight (int sosIdx, int nvar, double weight, int * idx, double * value, std::string * name)`
- 6.193.3.223 `bool OSOption::setNumberOfOtherVariableOptions (int numberOfOther)`
- 6.193.3.224 `bool OSOption::setOtherVariableOptions (int numberOfVar, OtherVariableOption ** var)`
- 6.193.3.225 `bool OSOption::setAnOtherVariableOption (OtherVariableOption * varOption)`
- 6.193.3.226 `bool OSOption::setOtherVariableOptionAttributes (int iOther, int numberOfVar, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string varType, std::string enumType, std::string description)`

Set the attributes for one particular <other> <variable> option.

Parameters

<i>iOther</i> ,:	position of this element in the array of <other>
<i>numberOfVar</i> ,:	number of <i>children contained in this <other> element</i>
<i>numberOfEnumerations</i> ,:	<i>number of <enumeration> children</i>
<i>name</i> ,:	<i>name of this <other> element</i>
<i>value</i> ,:	<i>a value associated with this <other> element</i>
<i>solver</i> ,:	<i>the solver associated with this <other> element</i>
<i>category</i> ,:	<i>the category of this <other> element</i>
<i>type</i> ,:	<i>type of this <other> element</i>
<i>varType</i> ,:	<i>type of the data in the array</i>
<i>enumType</i> ,:	<i>type of the data in the <enumeration> array</i>
<i>description</i> ,:	<i>further description of this <other> element</i>

- 6.193.3.227 `bool OSOption::setOtherOptionEnumeration (int object, int otherOptionNumber, int enumerationNumber, int numberOfEI, std::string value, std::string description, int * idxArray)`

Set one enumeration associated with an <other> option in the <variables>, <objectives> or <constraints> element.

Parameters

<i>object</i> ,:	the object into which the enumeration is to be stored (legal values see ENUM_PROBLEM_COMPONENT in OSGeneral.h)
<i>otherOptionNumber</i> ,:	number of the <other> option in the list of <other> options (zero-based)
<i>enumerationNumber</i> ,:	number of the <enumeration> in the list of enumerations (zero-based)

<i>numberOfEl,:</i>	number of objects sharing the value of this enumeration
<i>value,:</i>	value of the enumeration (as a string)
<i>description,:</i>	further information about the enumeration and its value
<i>idxArray,:</i>	the array of indices for the objects sharing this enumeration

6.193.3.228 `bool OSOption::setOtherVariableOptionVar (int otherOptionNumber, int varNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)`

Set one *element* associated with an *<other>* option in the *<variables>* element.

Parameters

<i>otherOption-Number,:</i>	number of the <i><other></i> option in the list of <i><other></i> options (zero-based)
<i>varNumber,:</i>	number of the <i>in the array</i> (zero-based)
<i>idx,:</i>	<i>index of the variable to which this value belongs</i>
<i>value,:</i>	<i>value of the option (as a string)</i>
<i>lbValue,:</i>	<i>value associated with the lower bound of the variable (as a string)</i>
<i>ubValue,:</i>	<i>value associated with the upper bound of the variable (as a string)</i>

6.193.3.229 `bool OSOption::setInitObjValues (int numberOfObj, int * idx, double * value, std::string * name)`

6.193.3.230 `bool OSOption::setInitObjValuesSparse (int numberOfObj, InitObjValue ** obj)`

6.193.3.231 `bool OSOption::setInitObjValuesSparse (int numberOfObj, InitObjValue ** obj, ENUM_COMBINE_ARRAYS disp)`

6.193.3.232 `bool OSOption::setInitObjValuesDense (int numberOfObj, double * value)`

6.193.3.233 `bool OSOption::setAnotherInitObjValue (int idx, double value)`

6.193.3.234 `bool OSOption::setInitObjBounds (int numberOfObj, int * idx, double * lbValue, double * ubValue, std::string * name)`

6.193.3.235 `bool OSOption::setInitObjBoundsSparse (int numberOfObj, InitObjBound ** obj)`

6.193.3.236 `bool OSOption::setInitObjBoundsSparse (int numberOfObj, InitObjBound ** obj, ENUM_COMBINE_ARRAYS disp)`

6.193.3.237 `bool OSOption::setInitObjBoundsDense (int numberOfObj, double * lb, double * ub)`

6.193.3.238 `bool OSOption::setAnotherInitObjBound (int idx, double lbValue, double ubValue)`

6.193.3.239 `bool OSOption::setNumberOfOtherObjectiveOptions (int numberOfOther)`

6.193.3.240 `bool OSOption::setOtherObjectiveOptions (int numberOfObj, OtherObjectiveOption ** obj)`

6.193.3.241 `bool OSOption::setAnOtherObjectiveOption (OtherObjectiveOption * objOption)`

6.193.3.242 `bool OSOption::setOtherObjectiveOptionAttributes (int iOther, int numberOfObj, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string objType, std::string enumType, std::string description)`

Set the attributes for one particular *<other>* *<objective>* option.

Parameters

<i>iOther,:</i>	position of this element in the array of <other>
<i>numberOfObj,:</i>	number of <obj> children contained in this <other> element
<i>numberOfEnumerations,:</i>	number of <enumeration> children
<i>name,:</i>	name of this <other> element
<i>value,:</i>	a value associated with this <other> element
<i>solver,:</i>	the solver associated with this <other> element
<i>category,:</i>	the category of this <other> element
<i>type,:</i>	type of this <other> element
<i>objType,:</i>	type of the data in the array
<i>enumType,:</i>	type of the data in the <enumeration> array
<i>description,:</i>	further description of this <other> element

6.193.3.243 `bool OSOption::setOtherObjectiveOptionObj (int otherOptionNumber, int objNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)`

Set one <obj> element associated with an <other> option in the <objectives> element.

Parameters

<i>otherOption-Number,:</i>	number of the <other> option in the list of <other> options (zero-based)
<i>objNumber,:</i>	number of the <obj> in the array (zero-based)
<i>idx,:</i>	index of the objective to which this value belongs
<i>name,:</i>	name of the objective
<i>value,:</i>	value of the option (as a string)
<i>lbValue,:</i>	value associated with the lower bound of the objective (as a string)
<i>ubValue,:</i>	value associated with the upper bound of the objective (as a string)

6.193.3.244 `bool OSOption::setInitConValues (int numberOfCon, int * idx, double * value, std::string * name)`

6.193.3.245 `bool OSOption::setInitConValuesSparse (int numberOfCon, InitConValue ** con)`

6.193.3.246 `bool OSOption::setInitConValuesSparse (int numberOfCon, InitConValue ** con, ENUM_COMBINE_ARRAYS disp)`

6.193.3.247 `bool OSOption::setInitConValuesDense (int numberOfCon, double * value)`

6.193.3.248 `bool OSOption::setAnotherInitConValue (int idx, double value)`

6.193.3.249 `bool OSOption::setInitDualValues (int numberOfCon, int * idx, double * lbValue, double * ubValue, std::string * name)`

6.193.3.250 `bool OSOption::setInitDualVarValuesSparse (int numberOfCon, InitDualVarValue ** con)`

6.193.3.251 `bool OSOption::setInitDualVarValuesSparse (int numberOfCon, InitDualVarValue ** con, ENUM_COMBINE_ARRAYS disp)`

6.193.3.252 `bool OSOption::setInitDualVarValuesDense (int numberOfCon, double * lb, double * ub)`

6.193.3.253 `bool OSOption::setAnotherInitDualVarValue (int idx, double lbValue, double ubValue)`

6.193.3.254 `bool OSOption::setNumberOfOtherConstraintOptions (int numberOfOther)`

6.193.3.255 `bool OSOption::setOtherConstraintOptions (int numberOfOptions, OtherConstraintOption ** other)`

6.193.3.256 `bool OSOption::setAnOtherConstraintOption (OtherConstraintOption * optionValue)`

6.193.3.257 `bool OSOption::setOtherConstraintOptionAttributes (int iOther, int numberOfCon, int numberOfEnumerations, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string conType, std::string enumType, std::string description)`

Set the attributes for one particular <other> <constraint> option.

Parameters

<i>iOther</i> ,:	position of this element in the array of <other>
<i>numberOfCon</i> ,:	number of <con> children contained in this <other> element
<i>numberOfEnumerations</i> ,:	number of <enumeration> children
<i>name</i> ,:	name of this <other> element
<i>value</i> ,:	a value associated with this <other> element
<i>solver</i> ,:	the solver associated with this <other> element
<i>category</i> ,:	the category of this <other> element
<i>type</i> ,:	type of this <other> element
<i>conType</i> ,:	type of the data in the array
<i>enumType</i> ,:	type of the data in the <enumeration> array
<i>description</i> ,:	further description of this <other> element

6.193.3.258 `bool OSOption::setOtherConstraintOptionCon (int otherOptionNumber, int conNumber, int idx, std::string name, std::string value, std::string lbValue, std::string ubValue)`

Set one <con> element associated with an <other> option in the <constraints> element.

Parameters

<i>otherOptionNumber</i> ,:	number of the <other> option in the list of <other> options (zero-based)
<i>conNumber</i> ,:	number of the <obj> in the array (zero-based)
<i>idx</i> ,:	index of the constraint to which this value belongs
<i>name</i> ,:	name of the constraint
<i>value</i> ,:	value of the option (as a string)
<i>lbValue</i> ,:	value associated with the lower bound of the constraint (as a string)
<i>ubValue</i> ,:	value associated with the upper bound of the constraint (as a string)

6.193.3.259 `bool OSOption::setNumberOfSolverOptions (int numberOfOptions)`

6.193.3.260 `bool OSOption::setSolverOptionContent (int iOption, int numberOfItems, std::string name, std::string value, std::string solver, std::string category, std::string type, std::string description, std::string * itemList)`

Set the attributes for one particular solver option.

Parameters

<i>iOption</i> ,:	position of this element in the array of options
<i>numberOfVar</i> ,:	number of children contained in this <other> element
<i>name</i> ,:	name of this solver option
<i>value</i> ,:	a value associated with this option
<i>solver</i> ,:	the solver to which this option applies

<i>category,:</i>	<i>the category of this option (solver specific)</i>
<i>type,:</i>	<i>type of this option (e.g., numeric or string)</i>
<i>description,:</i>	<i>further description of this option "param itemList: the list of items associated with this option (could be NULL)</i>

6.193.3.261 `bool OOption::setSolverOptions (int numberOfSolverOptions, SolverOption ** solverOption)`

6.193.3.262 `bool OOption::setAnotherSolverOption (std::string name, std::string value, std::string solver, std::string category, std::string type, std::string description)`

6.193.3.263 `bool OOption::setOptionInt (std::string optionName, int optionValue)`

6.193.3.264 `bool OOption::setOptionStr (std::string optionName, std::string optionValue)`

6.193.3.265 `bool OOption::setOptionDbf (std::string optionName, double value)`

6.193.4 Member Data Documentation

6.193.4.1 `GeneralFileHeader*` `OOption::optionHeader`

[OOption](#) has a header and five other children: general, system, service, job, and optimization.

header information

Definition at line 3576 of file `OOption.h`.

6.193.4.2 `GeneralOption*` `OOption::general`

`generalOption` holds the first child of the [OOption](#) specified by the OSoL Schema.

Definition at line 3580 of file `OOption.h`.

6.193.4.3 `SystemOption*` `OOption::system`

`systemOption` holds the second child of the [OOption](#) specified by the OSoL Schema.

Definition at line 3584 of file `OOption.h`.

6.193.4.4 `ServiceOption*` `OOption::service`

`serviceOption` holds the third child of the [OOption](#) specified by the OSoL Schema.

Definition at line 3588 of file `OOption.h`.

6.193.4.5 `JobOption*` `OOption::job`

`jobOption` holds the fourth child of the [OOption](#) specified by the OSoL Schema.

Definition at line 3592 of file `OOption.h`.

6.193.4.6 `OptimizationOption*` `OOption::optimization`

`optimizationOption` holds the fifth child of the [OOption](#) specified by the OSoL Schema.

Definition at line 3596 of file `OOption.h`.

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

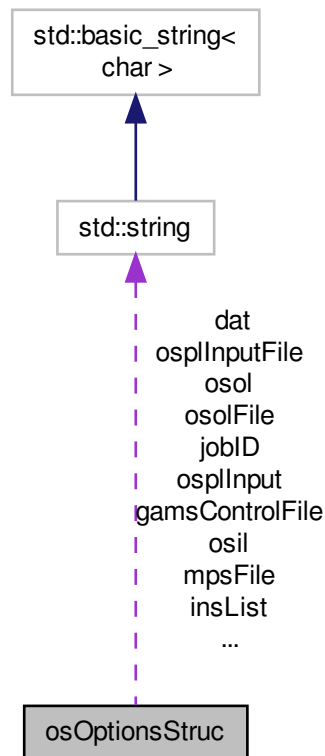
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OOption.h`

6.194 osOptionsStruc Struct Reference

This structure is used to store options for the OSSolverService executable.

```
#include <OSOptionsStruc.h>
```

Collaboration diagram for osOptionsStruc:



Public Member Functions

- `osOptionsStruc ()`
constructor
- `void resetOptions ()`
a method to reset the options to their default values

Public Attributes

- `std::string configFile`
configFile is the name of the file that holds the configuration options if the OSSolverService reads its options from a file rather than command line inputs
- `std::string osilFile`

- osilFile* is the name of the file that holds the model instance in OSiL format

 - std::string **osil**

osil is the content of the *osilFile*
- osolFile* is the name of the file that holds the solver options in OSoL format

 - std::string **osol**

osol is the content of the *osolFile*
- osrlFile* is the name of the file where the solver should write the result in OSrL format

 - std::string **osrl**

osrl is the content of the *osrlFile*
- name of the file containing the instance in LINDO instruction list format

 - std::string **insList**

insList is the content of the *insListFile* – THIS IS NOT IMPLEMENTED
- serviceLocation* is the URL of the remote solver when a local solver is not used

 - std::string **serviceMethod**

the service method the OSSolverService should execute, i.e.
- name of an input file with xml in OS Process language format, used for example to knock on a server, for example
-osplInput ../data/osplFiles/demo.ospl

 - std::string **osplInputFile**

osplInput is the content of the *osplInputFile*
- name of an output file where the solver should write the result of a knock or kill service request

 - std::string **osplOutput**

osplOutput is the content of the *osplOutputFile*
- the name of the file that holds an instance in MPS format

 - std::string **mps**

the string that holds an instance in MPS format
- the name of the file that holds an instance in AMPL nl format

 - std::string **nlFile**

the string that holds an instance in AMPL nl format
- the name of the file that holds an instance in GAMS dat format

 - std::string **datFile**

the string that holds an instance in GAMS dat format
- the name of the .dat that holds the GAMS control file

 - std::string **gamsControlFile**

the name of the solver to be invoked, e.g.
- this parameter is a path to the browser on the local machine.

 - std::string **browser**
- int **printLevel**

this parameter controls the amount of output to print the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

- std::string [logFile](#)

this optional parameter contains the path to a logfile that can be used as an alternate output stream in addition to the normal output to stdout

- int [filePrintLevel](#)

this parameter controls the amount of output to send to the log file (if used) the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

- std::string [jobID](#)

the JobID

- bool [invokeHelp](#)

if this parameter is true we print the contents of the file help.txt and return

- bool [writeVersion](#)

if this parameter is true, we print the current version of the OS project

- bool [printModel](#)

if this parameter is true, we print the current instance as read from an osil, nl or mps file

- std::string [printRowNumberAsString](#)

this parameter contains a string representation (!) of the row number if only a single row (constraint or objective) of the current instance is to be printed String representations are easier to parse in [OSParseosss.l](#) and are easier to recognize as being present or absent

- bool [quit](#)

if this parameter is true, we quit/exit the program – only used in the interactive shell

6.194.1 Detailed Description

This structure is used to store options for the OSSolverService executable.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Remarks

the OSSolverService requires numerous options and these options are stored in the [osOptionsStruc](#)

Definition at line 32 of file [OSOptionsStruc.h](#).

6.194.2 Constructor & Destructor Documentation

6.194.2.1 osOptionsStruc::osOptionsStruc ()

constructor

6.194.3 Member Function Documentation

6.194.3.1 void osOptionsStruc::resetOptions ()

a method to reset the options to their default values

6.194.4 Member Data Documentation

6.194.4.1 `std::string osOptionsStruc::configFile`

`configFile` is the name of the file that holds the configuration options if the `OSSolverService` reads its options from a file rather than command line inputs

Definition at line 38 of file `OSOptionsStruc.h`.

6.194.4.2 `std::string osOptionsStruc::osilFile`

`osilFile` is the name of the file that holds the model instance in OSiL format

Definition at line 43 of file `OSOptionsStruc.h`.

6.194.4.3 `std::string osOptionsStruc::osil`

`osil` is the content of the `osilFile`

Definition at line 47 of file `OSOptionsStruc.h`.

6.194.4.4 `std::string osOptionsStruc::osolFile`

`osolFile` is the name of the file that holds the solver options in OSoL format

Definition at line 52 of file `OSOptionsStruc.h`.

6.194.4.5 `std::string osOptionsStruc::osol`

`osol` is the content of the `osolFile`

Definition at line 56 of file `OSOptionsStruc.h`.

6.194.4.6 `std::string osOptionsStruc::osrlFile`

`osrlFile` is the name of the file where the solver should write the result in OSrL format

Definition at line 61 of file `OSOptionsStruc.h`.

6.194.4.7 `std::string osOptionsStruc::osrl`

`osrl` is the content of the `osrlFile`

Definition at line 65 of file `OSOptionsStruc.h`.

6.194.4.8 `std::string osOptionsStruc::insListFile`

name of the file containing the instance in LINDO instruction list format

Definition at line 70 of file `OSOptionsStruc.h`.

6.194.4.9 `std::string osOptionsStruc::insList`

`insList` is the content of the `insListFile` – THIS IS NOT IMPLEMENTED

Definition at line 74 of file `OSOptionsStruc.h`.

6.194.4.10 `std::string osOptionsStruc::serviceLocation`

`serviceLocation` is the URL of the remote solver when a local solver is not used

Definition at line 79 of file `OSOptionsStruc.h`.

6.194.4.11 std::string osOptionsStruc::serviceMethod

the service method the OSSolverService should execute, i.e.

solve, send, getJobID, kill, knock, or retrieve

Definition at line 84 of file OSOptionsStruc.h.

6.194.4.12 std::string osOptionsStruc::osplInputFile

name of an input file with xml in OS Process language format, used for example to knock on a server, for example
-osplInput ../data/osplFiles/demo.ospl

Definition at line 90 of file OSOptionsStruc.h.

6.194.4.13 std::string osOptionsStruc::osplInput

osplInput is the content of the osplInputFile

Definition at line 94 of file OSOptionsStruc.h.

6.194.4.14 std::string osOptionsStruc::osplOutputFile

name of an output file where the solver should write the result of a knock or kill service request

Definition at line 99 of file OSOptionsStruc.h.

6.194.4.15 std::string osOptionsStruc::osplOutput

osplOutput is the content of the osplOutputFile

Definition at line 103 of file OSOptionsStruc.h.

6.194.4.16 std::string osOptionsStruc::mpsFile

the name of the file that holds an instance in MPS format

Definition at line 106 of file OSOptionsStruc.h.

6.194.4.17 std::string osOptionsStruc::mps

the string that holds an instance in MPS format

Definition at line 109 of file OSOptionsStruc.h.

6.194.4.18 std::string osOptionsStruc::nlFile

the name of the file that holds an instance in AMPL nl format

Definition at line 112 of file OSOptionsStruc.h.

6.194.4.19 std::string osOptionsStruc::nl

the string that holds an instance in AMPL nl format

Definition at line 115 of file OSOptionsStruc.h.

6.194.4.20 std::string osOptionsStruc::datFile

the name of the file that holds an instance in GAMS dat format

Definition at line 118 of file OSOptionsStruc.h.

6.194.4.21 std::string osOptionsStruc::dat

the string that holds an instance in GAMS dat format

Definition at line 121 of file OSOptionsStruc.h.

6.194.4.22 std::string osOptionsStruc::gamsControlFile

the name of the .dat that holds the GAMS control file

Definition at line 124 of file OSOptionsStruc.h.

6.194.4.23 std::string osOptionsStruc::solverName

the name of the solver to be invoked, e.g.

-solver lpopt

Definition at line 129 of file OSOptionsStruc.h.

6.194.4.24 std::string osOptionsStruc::browser

this parameter is a path to the browser on the local machine.

If this optional parameter is specified then the solver result in OSrL format is transformed using XSLT into HTML and displayed in the browser, e.g. -browser /Applications/Firefox.app/Contents/MacOS/firefox

Definition at line 137 of file OSOptionsStruc.h.

6.194.4.25 int osOptionsStruc::printLevel

this parameter controls the amount of output to print the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

Definition at line 143 of file OSOptionsStruc.h.

6.194.4.26 std::string osOptionsStruc::logFile

this optional parameter contains the path to a logfile that can be used as an alternate output stream in addition to the normal output to stdout

Definition at line 149 of file OSOptionsStruc.h.

6.194.4.27 int osOptionsStruc::filePrintLevel

this parameter controls the amount of output to send to the log file (if used) the higher the number, the more output is generated details about print levels can be found in [OSOutput.h](#)

Definition at line 156 of file OSOptionsStruc.h.

6.194.4.28 std::string osOptionsStruc::jobID

the JobID

Definition at line 159 of file OSOptionsStruc.h.

6.194.4.29 bool osOptionsStruc::invokeHelp

if this parameter is true we print the contents of the file help.txt and return

Definition at line 164 of file OSOptionsStruc.h.

6.194.4.30 bool osOptionsStruc::writeVersion

if this parameter is true, we print the current version of the OS project

Definition at line 169 of file OSOptionsStruc.h.

6.194.4.31 bool osOptionsStruc::printModel

if this parameter is true, we print the current instance as read from an osil, nl or mps file

Definition at line 174 of file OSOptionsStruc.h.

6.194.4.32 std::string osOptionsStruc::printRowNumberAsString

this parameter contains a string representation (!) of the row number if only a single row (constraint or objective) of the current instance is to be printed String representations are easier to parse in OSParseosss.l and are easier to recognize as being present or absent

Definition at line 182 of file OSOptionsStruc.h.

6.194.4.33 bool osOptionsStruc::quit

if this parameter is true, we quit/exit the program – only used in the interactive shell

Definition at line 187 of file OSOptionsStruc.h.

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

- [/home/ted/COIN/trunk/OS/src/OSParsers/OSOptionsStruc.h](#)

6.195 OSosrl2ampl Class Reference

The [OSosrl2ampl](#) Class.

```
#include <OSosrl2ampl.h>
```

Public Member Functions

- [OSosrl2ampl](#) ()
the OSosrl2ampl class constructor
- [~OSosrl2ampl](#) ()
the OSosrl2ampl class destructor
- bool [writeSolFile](#) (std::string osrl, ASL *asl, std::string filename)
Convert the solution to AMPL .sol format.

6.195.1 Detailed Description

The [OSosrl2ampl](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

The [OSosrl2AMPL](#) class is used for writing an AMPL sol file, including any results indexed over variables, constraints, objectives.

Definition at line 44 of file OSosrl2AMPL.h.

6.195.2 Constructor & Destructor Documentation**6.195.2.1 OSosrl2AMPL::OSosrl2AMPL ()**

the [OSosrl2AMPL](#) class constructor

6.195.2.2 OSosrl2AMPL::~~OSosrl2AMPL ()

the [OSosrl2AMPL](#) class destructor

6.195.3 Member Function Documentation**6.195.3.1 bool OSosrl2AMPL::writeSolFile (std::string *osrl*, ASL * *asl*, std::string *filename*)**

Convert the solution to AMPL .sol format.

Parameters

<i>osrl</i>	is a string containing the result information
<i>asl</i>	is a pointer to an ASL data structure
<i>filename</i>	is the name of the output file (e.g., as returned from the solver).

Returns

whether the .sol file was created successfully.

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

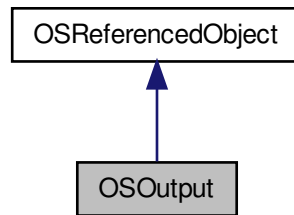
- [/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosrl2AMPL.h](#)

6.196 OSOutput Class Reference

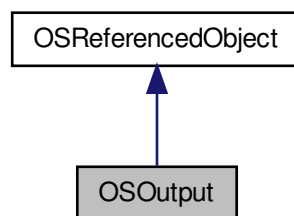
This class handles all the output from OSSolverService, OSAMPLClient and other executables derived from them.

```
#include <OSOutput.h>
```

Inheritance diagram for OSOutput:



Collaboration diagram for OSOutput:



Public Member Functions

- [OSOutput \(\)](#)
Constructor.
- [~OSOutput \(\)](#)
Destructor.
- [bool OSPrint \(ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level, std::string outStr\)](#)
This is the main method to output a string All output generated by the program should ultimately use this method.
- [void FlushAllBuffers \(\)](#)
Flush all buffers.
- [bool SetPrintLevel \(std::string name, ENUM_OUTPUT_LEVEL *level, int dim\)](#)
Modify all print levels associated with a channel.
- [bool SetPrintLevel \(std::string name, ENUM_OUTPUT_LEVEL level\)](#)
set the print level associated with a channel
- [int AddChannel \(std::string name\)](#)
Add a channel to the array outputChannel.

- bool [DeleteChannel](#) (std::string name)
Delete a channel from the array outputChannel.
- int [FindChannel](#) (std::string name)
*Find the position of a channel within the array *outputChannel.*

6.196.1 Detailed Description

This class handles all the output from OSSolverService, OSAmplClient and other executables derived from them.

Every output statement in the code uses methods in this class, passing information about the area that originated the request as well as the print, verbosity or severity level of the message. The message creates output only if the print level matches the user specifications. The main advantage of doing things this way is that multiple output streams can be maintained, each tailored to a specific need and containing only output that the user really wants to see. There can be as many output streams as needed; each one has an identifying name ("stdout" and "stderr" are reserved names) and an array of print levels, one for each area. The class is implemented as a Singleton, which means that two private methods must be defined in the header but must *never* be implemented: a copy constructor, and an equality operator.

Definition at line 146 of file OSOutput.h.

6.196.2 Constructor & Destructor Documentation

6.196.2.1 OSOutput::OSOutput ()

Constructor.

6.196.2.2 OSOutput::~~OSOutput ()

Destructor.

6.196.3 Member Function Documentation

6.196.3.1 bool OSOutput::OSPrint (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level, std::string outStr)

This is the main method to output a string All output generated by the program should ultimately use this method.

Parameters

<i>level</i> ,:	the print level associated with the string
<i>area</i> ,:	the area of the code in which the output was generated
<i>outStr</i> ,:	the string to be output

Returns

whether the output operation was successful

6.196.3.2 void OSOutput::FlushAllBuffers ()

Flush all buffers.

6.196.3.3 bool OSOutput::SetPrintLevel (std::string name, ENUM_OUTPUT_LEVEL * level, int dim)

Modify all print levels associated with a channel.

Parameters

<i>name,:</i>	The name of the channel ("stdout" and "stderr" are reserved names)
<i>level,:</i>	The array of print levels used for the output to this channel
<i>dim,:</i>	The number of entries in this array

Returns

whether the operation was successful

6.196.3.4 bool OSOutput::SetPrintLevel (std::string *name*, ENUM_OUTPUT_LEVEL *level*)

set the print level associated with a channel

Parameters

<i>name,:</i>	The name of the channel ("stdout" and "stderr" are reserved names)
<i>level,:</i>	The print level used for the output to this channel if < ENUM_OUTPUT_LEVEL_NUMBER_OF_ - LEVELS, set the (same) print level in all areas otherwise set the print level only in one particular area

Returns

whether the operation was successful

6.196.3.5 int OSOutput::AddChannel (std::string *name*)

Add a channel to the array outputChannel.

Parameters

<i>name,:</i>	The name of the channel ("stdout" and "stderr" are reserved names)
---------------	--

Returns

the status of the operation: 0: completed successfully 1: channel previously defined 2: out of memory 3: other error condition

6.196.3.6 bool OSOutput::DeleteChannel (std::string *name*)

Delete a channel from the array outputChannel.

Parameters

<i>name,:</i>	The name of the channel
---------------	-------------------------

Returns

whether the operation was completed successfully

6.196.3.7 int OSOutput::FindChannel (std::string *name*)

Find the position of a channel within the array *outputChannel.

Parameters

<i>name</i> ,:	The name of the channel
----------------	-------------------------

Returns

the position if found; -1 otherwise

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

- /home/ted/COIN/trunk/OS/src/OSUtils/[OSOutput.h](#)

6.197 OSOutputChannel Class Reference

a class that holds information about one output channel (file, device, stream, peripheral, etc.)

```
#include <OSOutput.h>
```

Public Member Functions

- [OSOutputChannel](#) (std::string name)
Constructor.
- [~OSOutputChannel](#) ()
Destructor.
- std::string [Name](#) ()
Get the name of the output channel.
- bool [setPrintLevel](#) (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level)
Set the print level for a particular area.
- bool [setAllPrintLevels](#) (ENUM_OUTPUT_LEVEL level)
Set the print level for all areas.
- bool [setAllPrintLevels](#) (ENUM_OUTPUT_LEVEL *level, int dim)
Set different print levels for all areas.
- bool [isAccepted](#) (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level)
Test if the device accepts a particular combination of print level and area (i.e., if the output should be printed)
- void [OSPrintf](#) (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level, std::string str)
Send one string to the output device provided that the output device "accepts" the output (i.e., the print level applicable to the area that originated the output exceeds the level of the print statement.
- void [flushBuffer](#) ()
Flush output buffer.

Open: open the channel

Returns

true if successfully opened; false otherwise

- bool [Open](#) ()

6.197.1 Detailed Description

a class that holds information about one output channel (file, device, stream, peripheral, etc.)

Definition at line 41 of file OSOutput.h.

6.197.2 Constructor & Destructor Documentation

6.197.2.1 OSOutputChannel::OSOutputChannel (std::string *name*)

Constructor.

Parameters

<i>name</i>	holds the name of the file or device that applies to this output device in all code areas
-------------	---

6.197.2.2 OSOutputChannel::~~OSOutputChannel ()

Destructor.

6.197.3 Member Function Documentation

6.197.3.1 std::string OSOutputChannel::Name ()

Get the name of the output channel.

6.197.3.2 bool OSOutputChannel::setPrintLevel (ENUM_OUTPUT_AREA *area*, ENUM_OUTPUT_LEVEL *level*)

Set the print level for a particular area.

Parameters

<i>area</i>	holds the area of the code to which this option is to be applied
<i>level</i>	holds a valid print level

Returns

whether the set() was successful

6.197.3.3 bool OSOutputChannel::setAllPrintLevels (ENUM_OUTPUT_LEVEL *level*)

Set the print level for all areas.

Parameters

<i>level</i>	holds a valid print level
--------------	---------------------------

Returns

whether the set() was successful

6.197.3.4 bool OSOutputChannel::setAllPrintLevels (ENUM_OUTPUT_LEVEL * *level*, int *dim*)

Set different print levels for all areas.

Parameters

<i>level</i>	holds an array of valid print levels
<i>dim</i>	holds the number of entries in the array level

Returns

whether the set() was successful

6.197.3.5 `bool OSOutputChannel::isAccepted (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level)`

Test if the device accepts a particular combination of print level and area (i.e., if the output should be printed)

6.197.3.6 `void OSOutputChannel::OSPrintf (ENUM_OUTPUT_AREA area, ENUM_OUTPUT_LEVEL level, std::string str)`

Send one string to the output device provided that the output device "accepts" the output (i.e., the print level applicable to the area that originated the output exceeds the level of the print statement.

Parameters

<i>area</i> ,:	the area in which the output string originated
<i>level</i> ,:	the print level associated with the string
<i>str</i> ,:	the string that is to be printed

6.197.3.7 `void OSOutputChannel::flushBuffer ()`

Flush output buffer.

6.197.3.8 `bool OSOutputChannel::Open ()`

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

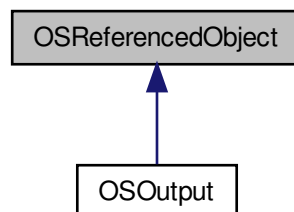
- </home/ted/COIN/trunk/OS/src/OSUtils/OSOutput.h>

6.198 OSReferencedObject Class Reference

ReferencedObject class.

```
#include <OSReferenced.hpp>
```

Inheritance diagram for OSReferencedObject:

**Public Member Functions**

- [OSReferencedObject](#) ()

- virtual `~OSReferencedObject()`
- int `ReferenceCount()` const
- void `AddRef(const OSReferencer *referencer)` const
- void `ReleaseRef(const OSReferencer *referencer)` const

6.198.1 Detailed Description

ReferencedObject class.

This is part of the implementation of an intrusive smart pointer design. This class stores the reference count of all the smart pointers that currently reference it. See the documentation for the [OSSmartPtr](#) class for more details.

A [OSSmartPtr](#) behaves much like a raw pointer, but manages the lifetime of an object, deleting the object automatically. This class implements a reference-counting, intrusive smart pointer design, where all objects pointed to must inherit off of `ReferencedObject`, which stores the reference count. Although this is intrusive (native types and externally authored classes require wrappers to be referenced by smart pointers), it is a safer design. A more detailed discussion of these issues follows after the usage information.

Usage Example: Note: to use the [OSSmartPtr](#), all objects to which you point MUST inherit from `ReferencedObject`.

```
*
* In MyClass.hpp...
*
* #include "OSReferenced.hpp"
*
* class MyClass : public ReferencedObject // must derive from ReferencedObject
* {
*     ...
* }
*
* In my_usage.cpp...
*
* #include "OSSmartPtr.hpp"
* #include "MyClass.hpp"
*
* void func(AnyObject& obj)
* {
*     OSSmartPtr<MyClass> ptr_to_myclass = new MyClass(...);
*     // ptr_to_myclass now points to a new MyClass,
*     // and the reference count is 1
*     ...
*     obj.SetMyClass(ptr_to_myclass);
*     // Here, let's assume that AnyObject uses a
*     // OSSmartPtr<MyClass> internally here.
*     // Now, both ptr_to_myclass and the internal
*     // OSSmartPtr in obj point to the same MyClass object
*     // and its reference count is 2.
*     ...
*     // No need to delete ptr_to_myclass, this
*     // will be done automatically when the
*     // reference count drops to zero.
* }
*
*
```

Other Notes: The [OSSmartPtr](#) implements both dereference operators `->` & `*`. The [OSSmartPtr](#) does NOT implement a conversion operator to the raw pointer. Use the [GetRawPtr\(\)](#) method when this is necessary. Make sure that the raw pointer is NOT deleted. The [OSSmartPtr](#) implements the comparison operators `==` & `!=` for a variety of types. Use these instead of


```
* if (GetRawPtr(smrt_ptr) == ptr) // Don't use this
*
```

OSSmartPtr's, as currently implemented, do NOT handle circular references. For example: consider a higher level object using OSSmartPtrs to point to A and B, but A and B also point to each other (i.e. A has an **OSSmartPtr** to B and B has an **OSSmartPtr** to A). In this scenario, when the higher level object is finished with A and B, their reference counts will never drop to zero (since they reference each other) and they will not be deleted. This can be detected by memory leak tools like valgrind. If the circular reference is necessary, the problem can be overcome by a number of techniques:

1) A and B can have a method that "releases" each other, that is they set their internal OSSmartPtrs to NULL.

```
* void AClass::ReleaseCircularReferences()
* {
*     smart_ptr_to_B = NULL;
* }
*
```

Then, the higher level class can call these methods before it is done using A & B.

2) Raw pointers can be used in A and B to reference each other. Here, an implicit assumption is made that the lifetime is controlled by the higher level object and that A and B will both exist in a controlled manner. Although this seems dangerous, in many situations, this type of referencing is very controlled and this is reasonably safe.

3) This **OSSmartPtr** class could be redesigned with the Weak/Strong design concept. Here, the **OSSmartPtr** is identified as being Strong (controls lifetime of the object) or Weak (merely referencing the object). The Strong **OSSmartPtr** increments (and decrements) the reference count in ReferencedObject but the Weak **OSSmartPtr** does not. In the example above, the higher level object would have Strong OSSmartPtrs to A and B, but A and B would have Weak OSSmartPtrs to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak OSSmartPtrs in A and B would not decrement the reference count and would, of course, not delete the object. This idea is very similar to item (2), where it is implied that the sequence of events is controlled such that A and B will not call anything using their pointers following the higher level delete (i.e. in their destructors!). This is somehow safer, however, because code can be written (however expensive) to perform run-time detection of this situation. For example, the ReferencedObject could store pointers to all Weak OSSmartPtrs that are referencing it and, in its destructor, tell these pointers that it is dying. They could then set themselves to NULL, or set an internal flag to detect usage past this point.

Comments on Non-Intrusive Design: In a non-intrusive design, the reference count is stored somewhere other than the object being referenced. This means, unless the reference counting pointer is the first referencer, it must get a pointer to the referenced object from another smart pointer (so it has access to the reference count location). In this non-intrusive design, if we are pointing to an object with a smart pointer (or a number of smart pointers), and we then give another smart pointer the address through a RAW pointer, we will have two independent, AND INCORRECT, reference counts. To avoid this pitfall, we use an intrusive reference counting technique where the reference count is stored in the object being referenced.

Definition at line 160 of file OSReferenced.hpp.

6.198.2 Constructor & Destructor Documentation

6.198.2.1 OSReferencedObject::OSReferencedObject () [inline]

Definition at line 163 of file OSReferenced.hpp.

6.198.2.2 virtual OSReferencedObject::~OSReferencedObject () [inline], [virtual]

Definition at line 168 of file OSReferenced.hpp.

6.198.3 Member Function Documentation

6.198.3.1 `int OSReferencedObject::ReferenceCount () const [inline]`

Definition at line 185 of file OSReferenced.hpp.

6.198.3.2 `void OSReferencedObject::AddRef (const OSReferencer * referencer) const [inline]`

Definition at line 191 of file OSReferenced.hpp.

6.198.3.3 `void OSReferencedObject::ReleaseRef (const OSReferencer * referencer) const [inline]`

Definition at line 197 of file OSReferenced.hpp.

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

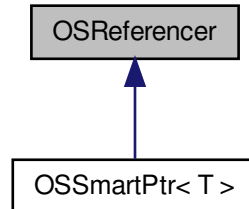
- [/home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.hpp](#)

6.199 OSReferencer Class Reference

Pseudo-class, from which everything has to inherit that wants to use be registered as a Referencer for a Referenced-Object.

```
#include <OSReferenced.hpp>
```

Inheritance diagram for OSReferencer:



6.199.1 Detailed Description

Pseudo-class, from which everything has to inherit that wants to use be registered as a Referencer for a Referenced-Object.

Definition at line 21 of file OSReferenced.hpp.

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

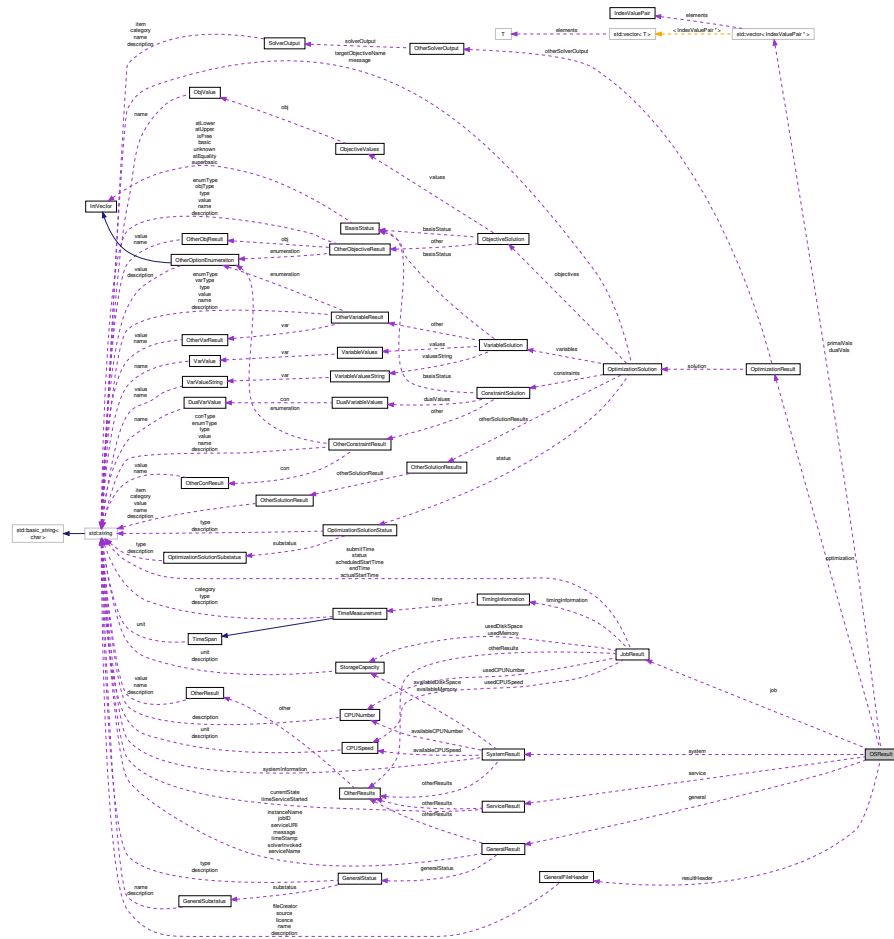
- [/home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.hpp](#)

6.200 OSResult Class Reference

The Result Class.

```
#include <OSResult.h>
```

Collaboration diagram for OSResult:



Public Member Functions

- [OSResult](#) ()
Default constructor.
- [~OSResult](#) ()
Class destructor.
- bool [setHeader](#) (std::string name, std::string source, std::string fileCreator, std::string description, std::string licence)
A function to populate an instance of the result header element.
- bool [isEqual](#) (OSResult *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- [GeneralStatus](#) * [getGeneralStatus](#) ()
Get the general status.
- std::string [getGeneralStatusType](#) ()

Get the general status type, which can be: success, error, warning.

- std::string [getGeneralStatusDescription](#) ()
Get the general status description.
- int [getNumberOfGeneralSubstatuses](#) ()
Get the number of substatuses.
- std::string [getGeneralSubstatusName](#) (int i)
Get the i_th general substatus name.
- std::string [getGeneralSubstatusDescription](#) (int i)
Get the i_th general substatus description.
- std::string [getGeneralMessage](#) ()
Get the general message.
- std::string [getServiceName](#) ()
Get service name.
- std::string [getServiceURI](#) ()
Get service uri.
- std::string [getInstanceName](#) ()
Get instance name.
- std::string [getJobID](#) ()
Get the job id.
- std::string [getSolverInvoked](#) ()
Get the solver invoked.
- std::string [getTimeStamp](#) ()
Get the time stamp.
- int [getNumberOfOtherGeneralResults](#) ()
Get the number of other results in the <general> element.
- std::string [getOtherGeneralResultName](#) (int idx)
Get the name of the i-th other result in the <general> element.
- std::string [getOtherGeneralResultValue](#) (int idx)
- std::string [getOtherGeneralResultDescription](#) (int idx)
- std::string [getSystemInformation](#) ()
- std::string [getAvailableDiskSpaceUnit](#) ()
- std::string [getAvailableDiskSpaceDescription](#) ()
- double [getAvailableDiskSpaceValue](#) ()
- std::string [getAvailableMemoryUnit](#) ()
- std::string [getAvailableMemoryDescription](#) ()
- double [getAvailableMemoryValue](#) ()
- std::string [getAvailableCPUSpeedUnit](#) ()
- std::string [getAvailableCPUSpeedDescription](#) ()
- double [getAvailableCPUSpeedValue](#) ()
- std::string [getAvailableCPUNumberDescription](#) ()
- int [getAvailableCPUNumberValue](#) ()
- int [getNumberOfOtherSystemResults](#) ()
- std::string [getOtherSystemResultName](#) (int idx)
- std::string [getOtherSystemResultValue](#) (int idx)
- std::string [getOtherSystemResultDescription](#) (int idx)
- std::string [getCurrentState](#) ()
- int [getCurrentJobCount](#) ()
- int [getTotalJobsSoFar](#) ()
- std::string [getTimeServiceStarted](#) ()

- double [getServiceUtilization](#) ()
- int [getNumberOfOtherServiceResults](#) ()
- std::string [getOtherServiceResultName](#) (int idx)
- std::string [getOtherServiceResultValue](#) (int idx)
- std::string [getOtherServiceResultDescription](#) (int idx)
- std::string [getJobStatus](#) ()
- std::string [getJobSubmitTime](#) ()
- std::string [getScheduledStartTime](#) ()
- std::string [getActualStartTime](#) ()
- std::string [getJobEndTime](#) ()
- int [getTimeNumber](#) ()
 - Get the number of time measurements.*
- double [getTimeValue](#) ()
 - Get the time measurement.*
- int [getNumberOfTimes](#) ()
- std::string [getTimingInfoUnit](#) (int idx)
- std::string [getTimingInfoType](#) (int idx)
- std::string [getTimingInfoCategory](#) (int idx)
- std::string [getTimingInfoDescription](#) (int idx)
- double [getTimingInfoValue](#) (int idx)
- std::string [getUsedDiskSpaceUnit](#) ()
- std::string [getUsedDiskSpaceDescription](#) ()
- double [getUsedDiskSpaceValue](#) ()
- std::string [getUsedMemoryUnit](#) ()
- std::string [getUsedMemoryDescription](#) ()
- double [getUsedMemoryValue](#) ()
- std::string [getUsedCPUSpeedUnit](#) ()
- std::string [getUsedCPUSpeedDescription](#) ()
- double [getUsedCPUSpeedValue](#) ()
- std::string [getUsedCPUNumberDescription](#) ()
- int [getUsedCPUNumberValue](#) ()
- int [getNumberOfOtherJobResults](#) ()
- std::string [getOtherJobResultName](#) (int idx)
- std::string [getOtherJobResultValue](#) (int idx)
- std::string [getOtherJobResultDescription](#) (int idx)
- int [getVariableNumber](#) ()
 - Get variable number.*
- int [getObjectiveNumber](#) ()
 - Get objective number.*
- int [getConstraintNumber](#) ()
 - Get constraint number.*
- int [getSolutionNumber](#) ()
 - get the number of solutions.*
- [OptimizationSolutionStatus](#) * [getSolutionStatus](#) (int solldx)
 - Get the [i]th optimization solution status, where i equals the given solution index.*
- std::string [getSolutionStatusType](#) (int solldx)
 - Get the [i]th optimization solution status type, where i equals the given solution index.*
- std::string [getSolutionStatusDescription](#) (int solldx)
 - Get the [i]th optimization solution status description, where i equals the given solution index.*

- int [getNumberOfSolutionSubstatuses](#) (int solIdx)
- std::string [getSolutionSubstatusType](#) (int solIdx, int substatusIdx)
- std::string [getSolutionSubstatusDescription](#) (int solIdx, int substatusIdx)
- int [getSolutionTargetObjectiveIdx](#) (int solIdx)
- std::string [getSolutionTargetObjectiveName](#) (int solIdx)
- bool [getSolutionWeightedObjectives](#) (int solIdx)
 - Get the [i]th optimization solution form of the objective.*
- std::string [getSolutionMessage](#) (int solIdx)
 - Get the [i]th optimization solution message, where i equals the given solution index.*
- int [getNumberOfPrimalVariableValues](#) (int solIdx)
- int [getNumberOfVarValues](#) (int solIdx)
- int [getVarValueIdx](#) (int solIdx, int varIdx)
- std::string [getVarValueName](#) (int solIdx, int varIdx)
- double [getVarValue](#) (int solIdx, int varIdx)
- std::vector< [IndexValuePair](#) * > [getOptimalPrimalVariableValues](#) (int solIdx)
 - Get one solution of optimal primal variable values.*
- int [getNumberOfVarValuesString](#) (int solIdx)
- int [getVarValueStringIdx](#) (int solIdx, int varIdx)
- std::string [getVarValueStringName](#) (int solIdx, int varIdx)
- std::string [getVarValueString](#) (int solIdx, int varIdx)
- int [getBasisStatusNumberOfEI](#) (int solIdx, int object, int status)
 - Get the number of indices that belong to a particular basis status.*
- int [getBasisStatusEI](#) (int solIdx, int object, int status, int j)
 - Get an entry in the array of indices that belong to a particular basis status.*
- int [getBasisInformationDense](#) (int solIdx, int object, int *resultArray, int dim)
 - Get the basis information associated with the variables, objectives or constraints for some solution.*
- int [getNumberOfOtherVariableResults](#) (int solIdx)
 - Get numberOfOtherVariableResult.*
- int [getAnOtherVariableResultNumberOfVar](#) (int solIdx, int iOther)
 - Get getAnOtherVariableResultNumberOfVar.*
- std::string [getOtherVariableResultName](#) (int solIdx, int otherIdx)
- std::string [getOtherVariableResultType](#) (int solIdx, int otherIdx)
- std::string [getOtherVariableResultValue](#) (int solIdx, int otherIdx)
- std::string [getOtherVariableResultDescription](#) (int solIdx, int otherIdx)
- int [getOtherVariableResultNumberOfVar](#) (int solIdx, int otherIdx)
- int [getOtherVariableResultNumberOfEnumerations](#) (int solIdx, int otherIdx)
- int [getOtherVariableResultVarIdx](#) (int solIdx, int otherIdx, int varIdx)
- std::string [getOtherVariableResultVar](#) (int solIdx, int otherIdx, int varIdx)
- std::string [getOtherVariableResultArrayType](#) (int solIdx, int otherIdx)
 - Get the type of values contained in the or <enumeration> array associated with an <other> result for some solution.*
- std::string [getOtherVariableResultEnumerationValue](#) (int solIdx, int otherIdx, int enumIdx)
 - Get the value of an enum associated with an <other> result for some solution.*
- std::string [getOtherVariableResultEnumerationDescription](#) (int solIdx, int otherIdx, int enumIdx)
 - Get the description of an enum associated with an <other> result for some solution.*
- int [getOtherVariableResultEnumerationNumberOfEI](#) (int solIdx, int otherIdx, int enumIdx)
 - Get the size of an enum associated with an <other> result for some solution.*
- int [getOtherVariableResultEnumerationEI](#) (int solIdx, int otherIdx, int enumIdx, int j)
 - Get one index of an enum associated with an <other> result for some solution.*
- int [getOtherVariableResultArrayDense](#) (int solIdx, int otherIdx, std::string *resultArray, int dim)

Get the values of a array or an <enumeration> associated with an <other> result for some solution.

- int [getNumberOfObjValues](#) (int solIdx)
- int [getObjValueIdx](#) (int solIdx, int objIdx)
- std::string [getObjValueName](#) (int solIdx, int objIdx)
- double [getObjValue](#) (int solIdx, int objIdx)
- double [getOptimalObjValue](#) (int objIdx, int solIdx)

Get one solution's optimal objective value.

- int [getNumberOfOtherObjectiveResults](#) (int solIdx)
- std::string [getOtherObjectiveResultName](#) (int solIdx, int otherIdx)
- std::string [getOtherObjectiveResultType](#) (int solIdx, int otherIdx)
- std::string [getOtherObjectiveResultValue](#) (int solIdx, int otherIdx)
- std::string [getOtherObjectiveResultDescription](#) (int solIdx, int otherIdx)
- int [getOtherObjectiveResultNumberOfObj](#) (int solIdx, int otherIdx)
- int [getOtherObjectiveResultNumberOfEnumerations](#) (int solIdx, int otherIdx)
- int [getOtherObjectiveResultObjIdx](#) (int solIdx, int otherIdx, int objIdx)
- std::string [getOtherObjectiveResultObj](#) (int solIdx, int otherIdx, int objIdx)
- std::string [getOtherObjectiveResultArrayType](#) (int solIdx, int otherIdx)

Get the type of values contained in the <obj> or <enumeration> array associated with an <other> result for some solution.

- std::string [getOtherObjectiveResultEnumerationValue](#) (int solIdx, int otherIdx, int enumIdx)

Get the value of an enum associated with an <other> result for some solution.

- std::string [getOtherObjectiveResultEnumerationDescription](#) (int solIdx, int otherIdx, int enumIdx)

Get the description of an enum associated with an <other> result for some solution.

- int [getOtherObjectiveResultEnumerationNumberOfEI](#) (int solIdx, int otherIdx, int enumIdx)

Get the size of an enum associated with an <other> result for some solution.

- int [getOtherObjectiveResultEnumerationEI](#) (int solIdx, int otherIdx, int enumIdx, int j)

Get one index of an enum associated with an <other> result for some solution.

- int [getOtherObjectiveResultArrayDense](#) (int solIdx, int otherIdx, std::string *resultArray, int dim)

Get the values of an <obj> array or an <enumeration> associated with an <other> result for some solution.

- int [getNumberOfDualValues](#) (int solIdx)
- int [getDualValueIdx](#) (int solIdx, int conIdx)
- std::string [getDualValueName](#) (int solIdx, int objIdx)
- double [getDualValue](#) (int solIdx, int conIdx)
- std::vector< [IndexValuePair](#) * > [getOptimalDualVariableValues](#) (int solIdx)

Get one solution of optimal dual variable values.

- int [getNumberOfOtherConstraintResults](#) (int solIdx)
- std::string [getOtherConstraintResultName](#) (int solIdx, int otherIdx)
- std::string [getOtherConstraintResultType](#) (int solIdx, int otherIdx)
- std::string [getOtherConstraintResultValue](#) (int solIdx, int otherIdx)
- std::string [getOtherConstraintResultDescription](#) (int solIdx, int otherIdx)
- int [getOtherConstraintResultNumberOfCon](#) (int solIdx, int otherIdx)
- int [getOtherConstraintResultNumberOfEnumerations](#) (int solIdx, int otherIdx)
- int [getOtherConstraintResultConIdx](#) (int solIdx, int otherIdx, int conIdx)
- std::string [getOtherConstraintResultCon](#) (int solIdx, int otherIdx, int conIdx)
- std::string [getOtherConstraintResultArrayType](#) (int solIdx, int otherIdx)

Get the type of values contained in the <con> or <enumeration> array associated with an <other> result for some solution.

- std::string [getOtherConstraintResultEnumerationValue](#) (int solIdx, int otherIdx, int enumIdx)

Get the value of an enum associated with an <other> result for some solution.

- `std::string getOtherConstraintResultEnumerationDescription` (int solIdx, int otherIdx, int enumIdx)
Get the description of an enum associated with an <other> result for some solution.
- `int getOtherConstraintResultEnumerationNumberOfEl` (int solIdx, int otherIdx, int enumIdx)
Get the size of an enum associated with an <other> result for some solution.
- `int getOtherConstraintResultEnumerationEl` (int solIdx, int otherIdx, int enumIdx, int j)
Get one index of an enum associated with an <other> result for some solution.
- `int getOtherConstraintResultArrayDense` (int solIdx, int otherIdx, std::string *resultArray, int dim)
Get the values of a <con> array or an <enumeration> associated with an <other> result for some solution.
- `int getNumberOfOtherSolutionResults` (int solIdx)
- `std::string getOtherSolutionResultName` (int solIdx, int otherIdx)
- `std::string getOtherSolutionResultValue` (int solIdx, int otherIdx)
- `std::string getOtherSolutionResultCategory` (int solIdx, int otherIdx)
- `std::string getOtherSolutionResultDescription` (int solIdx, int otherIdx)
- `int getOtherSolutionResultNumberOfItems` (int solIdx, int otherIdx)
- `std::string getOtherSolutionResultItem` (int solIdx, int otherIdx, int itemIdx)
- `int getNumberOfSolverOutputs` ()
- `std::string getSolverOutputName` (int otherIdx)
- `std::string getSolverOutputCategory` (int otherIdx)
- `std::string getSolverOutputDescription` (int otherIdx)
- `int getSolverOutputNumberOfItems` (int otherIdx)
- `std::string getSolverOutputItem` (int otherIdx, int itemIdx)
- `bool setGeneralStatus` (GeneralStatus *status)
Set the general status.
- `bool setGeneralStatusType` (std::string type)
Set the general status type, which can be: success, error, warning.
- `bool setNumberOfGeneralSubstatuses` (int num)
Set the number of substatus elements.
- `bool setGeneralStatusDescription` (std::string description)
Set the general status description.
- `bool setGeneralSubstatusName` (int idx, std::string name)
Set the general substatus name.
- `bool setGeneralSubstatusDescription` (int idx, std::string description)
Set the general substatus description.
- `bool setGeneralMessage` (std::string message)
Set the general message.
- `bool setServiceName` (std::string serviceName)
Set service name.
- `bool setServiceURI` (std::string serviceURI)
Set service uri.
- `bool setInstanceName` (std::string instanceName)
Set instance name.
- `bool setJobID` (std::string jobID)
Set job id.
- `bool setSolverInvoked` (std::string solverInvoked)
Set solver invoked.
- `bool setTimeStamp` (std::string timeStamp)
Set time stamp.
- `bool setNumberOfOtherGeneralResults` (int num)

- Set number of other general results.*
- bool [setOtherGeneralResultName](#) (int idx, std::string name)
Set the general otherResult name.
- bool [setOtherGeneralResultValue](#) (int idx, std::string value)
Set the general otherResult value.
- bool [setOtherGeneralResultDescription](#) (int idx, std::string description)
Set the general otherResult description.
- bool [setSystemInformation](#) (std::string systemInformation)
Set the system information.
- bool [setAvailableDiskSpaceUnit](#) (std::string unit)
Set the unit in which available disk space is measured.
- bool [setAvailableDiskSpaceDescription](#) (std::string description)
Set the description of available disk space.
- bool [setAvailableDiskSpaceValue](#) (double value)
Set the amount of available disk space.
- bool [setAvailableMemoryUnit](#) (std::string unit)
Set the unit in which available memory is measured.
- bool [setAvailableMemoryDescription](#) (std::string description)
Set the description of available memory.
- bool [setAvailableMemoryValue](#) (double value)
Set the amount of available memory.
- bool [setAvailableCPUSpeedUnit](#) (std::string unit)
Set the unit in which available CPU speed is measured.
- bool [setAvailableCPUSpeedDescription](#) (std::string description)
Set the description of available CPU speed.
- bool [setAvailableCPUSpeedValue](#) (double value)
Set the available CPU speed.
- bool [setAvailableCPUNumberDescription](#) (std::string description)
Set the description of available number of CPUs.
- bool [setAvailableCPUNumberValue](#) (int value)
Set the available number of CPUs.
- bool [setNumberOfOtherSystemResults](#) (int num)
Set number of other system results.
- bool [setOtherSystemResultName](#) (int idx, std::string name)
Set the system otherResult name.
- bool [setOtherSystemResultValue](#) (int idx, std::string value)
Set the system otherResult value.
- bool [setOtherSystemResultDescription](#) (int idx, std::string description)
Set the system otherResult description.
- bool [setCurrentState](#) (std::string currentState)
Set the current state of the service.
- bool [setCurrentJobCount](#) (int jobCount)
Set the current job count.
- bool [setTotalJobsSoFar](#) (int number)
Set the total number of jobs so far.
- bool [setTimeServiceStarted](#) (std::string startTime)
Set the time the service was started.

- bool [setServiceUtilization](#) (double value)
Set the service utilization.
- bool [setNumberOfOtherServiceResults](#) (int num)
Set number of other service results.
- bool [setOtherServiceResultName](#) (int idx, std::string name)
Set the service otherResult name.
- bool [setOtherServiceResultValue](#) (int idx, std::string value)
Set the service otherResult value.
- bool [setOtherServiceResultDescription](#) (int idx, std::string description)
Set the service otherResult description.
- bool [setJobStatus](#) (std::string status)
Set the job status.
- bool [setJobSubmitTime](#) (std::string submitTime)
Set the time when the job was submitted.
- bool [setScheduledStartTime](#) (std::string scheduledStartTime)
Set the job's scheduled start time.
- bool [setActualStartTime](#) (std::string actualStartTime)
Set the job's actual start time.
- bool [setJobEndTime](#) (std::string endTime)
Set the time when the job finished.
- bool [setTime](#) (double time)
Set time.
- bool [addTimingInformation](#) (std::string type, std::string category, std::string unit, std::string description, double value)
Add timing information.
- bool [setTimingInformation](#) (int idx, std::string type, std::string category, std::string unit, std::string description, double value)
Set timing information.
- bool [setNumberOfTimes](#) (int numberOfTimes)
Set the number of time measurements and initial the time array.
- bool [setTimeNumber](#) (int timeNumber)
Set the number of time measurements.
- bool [setUsedDiskSpaceUnit](#) (std::string unit)
Set the unit in which used disk space is measured.
- bool [setUsedDiskSpaceDescription](#) (std::string description)
Set the description of used disk space.
- bool [setUsedDiskSpaceValue](#) (double value)
Set the amount of used disk space.
- bool [setUsedMemoryUnit](#) (std::string unit)
Set the unit in which used memory is measured.
- bool [setUsedMemoryDescription](#) (std::string description)
Set the description of used memory.
- bool [setUsedMemoryValue](#) (double value)
Set the amount of used memory.
- bool [setUsedCPUSpeedUnit](#) (std::string unit)
Set the unit in which used CPU speed is measured.
- bool [setUsedCPUSpeedDescription](#) (std::string description)

- Set the description of used CPU speed.*

 - bool [setUsedCPUSpeedValue](#) (double value)
- Set the used CPU speed.*

 - bool [setUsedCPUNumberDescription](#) (std::string description)
- Set the description of used number of CPUs.*

 - bool [setUsedCPUNumberValue](#) (int value)
- Set the used number of CPUs.*

 - bool [setNumberOfOtherJobResults](#) (int num)
- Set number of other job results.*

 - bool [setOtherJobResultName](#) (int idx, std::string name)
- Set the job otherResult name.*

 - bool [setOtherJobResultValue](#) (int idx, std::string value)
- Set the job otherResult value.*

 - bool [setOtherJobResultDescription](#) (int idx, std::string description)
- Set the job otherResult description.*

 - bool [setVariableNumber](#) (int variableNumber)
- Set the variable number.*

 - bool [setObjectiveNumber](#) (int objectiveNumber)
- Set the objective number.*

 - bool [setConstraintNumber](#) (int constraintNumber)
- Set the constraint number.*

 - bool [setSolutionNumber](#) (int number)
- set the number of solutions.*

 - bool [setSolutionStatus](#) (int solldx, std::string type, std::string description)
- Set the [i]th optimization solution status, where i equals the given solution index.*

 - bool [setSolutionStatusType](#) (int solldx, std::string type)
- Set the [i]th optimization solution status type.*

 - bool [setNumberOfSolutionSubstatuses](#) (int solldx, int num)
- Set the [i]th optimization solution's number of substatus elements.*

 - bool [setSolutionStatusDescription](#) (int solldx, std::string description)
- Set the [i]th optimization solution status description.*

 - bool [setSolutionSubstatusType](#) (int solldx, int substatusIdx, std::string type)
- Set the solution substatus type.*

 - bool [setSolutionSubstatusDescription](#) (int solldx, int substatusIdx, std::string description)
- Set the solution substatus description.*

 - bool [setSolutionTargetObjectiveIdx](#) (int solldx, int objectiveIdx)
- Set the [i]th optimization solution's objective index, where i equals the given solution index.*

 - bool [setSolutionTargetObjectiveName](#) (int solldx, std::string objectiveName)
- Set the [i]th optimization solution's objective name, where i equals the given solution index.*

 - bool [setSolutionWeightedObjectives](#) (int solldx, bool weightedObjectives)
- Record whether the [i]th optimization solution uses weighted objectives, where i equals the given solution index.*

 - bool [setSolutionMessage](#) (int solldx, std::string msg)
- Set the [i]th optimization solution's message, where i equals the given solution index.*

 - bool [setNumberOfPrimalVariableValues](#) (int solldx, int n)
- Set the [i]th optimization solution's number of primal variable values, where i equals the given solution index.*

 - bool [setPrimalVariableValuesSparse](#) (int solldx, std::vector< [IndexValuePair](#) * > x)
- Set the [i]th optimization solution's primal variable values, where i equals the given solution index.*

- bool [setPrimalVariableValuesDense](#) (int solIdx, double *x)
Set the [i]th optimization solution's primal variable values, where i equals the given solution index.
- bool [setNumberOfVarValues](#) (int solIdx, int numberOfVar)
Set the number of primal variables to be given a value.
- bool [setVarValue](#) (int solIdx, int number, int idx, std::string name, double val)
Set a primal variable value.
- bool [setNumberOfVarValuesString](#) (int solIdx, int numberOfVar)
Set the number of string-valued primal variables to be given a value.
- bool [setVarValueString](#) (int solIdx, int number, int idx, std::string name, std::string str)
Set a string-valued primal variable value.
- bool [setBasisStatus](#) (int solIdx, int object, int status, int *i, int ni)
Set the basis status of a number of variables/constraints/objectives.
- bool [setNumberOfOtherVariableResults](#) (int solIdx, int numberOfOtherVariableResults)
Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.
- bool [setAnOtherVariableResultSparse](#) (int solIdx, int otherIdx, std::string name, std::string value, std::string description, int *idx, std::string *s, int n)
Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.
- bool [setAnOtherVariableResultSparse](#) (int solIdx, int otherIdx, std::string name, std::string value, std::string description, int *idx, std::string *s, int n, std::string type, std::string varType, std::string enumType)
Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.
- bool [setAnOtherVariableResultDense](#) (int solIdx, int otherIdx, std::string name, std::string value, std::string description, std::string *s)
Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.
- bool [setAnOtherVariableResultDense](#) (int solIdx, int otherIdx, std::string name, std::string value, std::string description, std::string *s, std::string type, std::string varType, std::string enumType)
Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.
- bool [setOtherVariableResultNumberOfVar](#) (int solIdx, int otherIdx, int numberOfVar)
Set the number of children of another (non-standard/solver specific) variable-related result, for the [i]th solution.
- bool [setOtherVariableResultNumberOfEnumerations](#) (int solIdx, int otherIdx, int numberOfEnumerations)
Set the number of <enumeration> children of another (non-standard/solver specific) variable-related result, for the [i]th solution.
- bool [setOtherVariableResultName](#) (int solIdx, int otherIdx, std::string name)
Set the name of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultType](#) (int solIdx, int otherIdx, std::string type)
Set the type of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultVarType](#) (int solIdx, int otherIdx, std::string varType)
Set the varType of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultEnumType](#) (int solIdx, int otherIdx, std::string enumType)
Set the enumType of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultValue](#) (int solIdx, int otherIdx, std::string value)

Set the value of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

- bool [setOtherVariableResultDescription](#) (int solIdx, int otherIdx, std::string description)
Set the description of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultVarIdx](#) (int solIdx, int otherIdx, int varIdx, int idx)
Set the index of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultVarName](#) (int solIdx, int otherIdx, int varIdx, std::string name)
Set the name of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultVar](#) (int solIdx, int otherIdx, int varIdx, std::string value)
Set the value of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherOptionEnumeration](#) (int solIdx, int otherIdx, int object, int enumIdx, std::string value, std::string description, int *i, int ni)
Set the value and corresponding indices of another (non-standard/solver specific) variable-related result, for the [k]th solution, where k equals the given solution index.
- bool [setNumberOfOtherObjectiveResults](#) (int solIdx, int numberOfOtherObjectiveResults)
Set the [i]th optimization solution's other (non-standard/solver specific) objective-related results, where i equals the given solution index.
- bool [setNumberOfObjValues](#) (int solIdx, int numberOfObj)
Set the number of objectives to be given a value.
- bool [setNumberOfObjectiveValues](#) (int solIdx, int n)
Set the [i]th optimization solution's number of objective values, where i equals the given solution index.
- bool [setObjectiveValuesSparse](#) (int solIdx, std::vector< [IndexValuePair](#) * > x)
Set the [i]th optimization solution's objective values, where i equals the given solution index.
- bool [setObjectiveValuesDense](#) (int solIdx, double *objectiveValues)
Set the [i]th optimization solution's objective values, where i equals the given solution index.
- bool [setObjValue](#) (int solIdx, int number, int idx, std::string name, double val)
Set an objective value.
- bool [setOtherObjectiveResultNumberOfObj](#) (int solIdx, int otherIdx, int numberOfObj)
Set the number of <obj> children of another (non-standard/solver specific) objective-related result, for the [i]th solution.
- bool [setOtherObjectiveResultNumberOfEnumerations](#) (int solIdx, int otherIdx, int numberOfObj)
Set the number of <enumeration> children of another (non-standard/solver specific) objective-related result, for the [i]th solution.
- bool [setOtherObjectiveResultName](#) (int solIdx, int otherIdx, std::string name)
Set the name of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultType](#) (int solIdx, int otherIdx, std::string type)
Set the type of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultObjType](#) (int solIdx, int otherIdx, std::string objType)
Set the objType of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultEnumType](#) (int solIdx, int otherIdx, std::string enumType)
Set the enumType of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultValue](#) (int solIdx, int otherIdx, std::string value)

- Set the value of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.*
- bool [setOtherObjectiveResultDescription](#) (int solIdx, int otherIdx, std::string description)
Set the description of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherObjectiveResultObjIdx](#) (int solIdx, int otherIdx, int objIdx, int idx)
Set the index of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherObjectiveResultObjName](#) (int solIdx, int otherIdx, int objIdx, std::string name)
Set the name of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherObjectiveResultObj](#) (int solIdx, int otherIdx, int objIdx, std::string value)
Set the value of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setNumberOfOtherConstraintResults](#) (int solIdx, int numberOfOtherConstraintResults)
Set the [i]th optimization solution's other (non-standard/solver specific) constraint-related results, where i equals the given solution index.
 - bool [setNumberOfDualValues](#) (int solIdx, int numberOfCon)
Set the number of constraints to be given a value.
 - bool [setNumberOfDualVariableValues](#) (int solIdx, int n)
Set the [i]th optimization solution's number of dual variable values, where i equals the given solution index.
 - bool [setDualVariableValuesSparse](#) (int solIdx, std::vector< [IndexValuePair](#) * > x)
Set the [i]th optimization solution's dual variable values, where i equals the given solution index.
 - bool [setDualVariableValuesDense](#) (int solIdx, double *y)
Set the [i]th optimization solution's dual variable values, where i equals the given solution index.
 - bool [setConstraintValuesDense](#) (int solIdx, double *constraintValues)
Set the [i]th optimization solution's constraint values, where i equals the given solution index.
 - bool [setDualValue](#) (int solIdx, int number, int idx, std::string name, double val)
Set a dual value.
 - bool [setOtherConstraintResultNumberOfCon](#) (int solIdx, int otherIdx, int numberOfCon)
Set the number of <con> children of another (non-standard/solver specific) constraint-related result, for the [i]th solution.
 - bool [setOtherConstraintResultNumberOfEnumerations](#) (int solIdx, int otherIdx, int numberOfCon)
Set the number of <enumeration> children of another (non-standard/solver specific) constraint-related result, for the [i]th solution.
 - bool [setOtherConstraintResultName](#) (int solIdx, int otherIdx, std::string name)
Set the name of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherConstraintResultType](#) (int solIdx, int otherIdx, std::string type)
Set the type of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherConstraintResultConType](#) (int solIdx, int otherIdx, std::string conType)
Set the conType of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherConstraintResultEnumType](#) (int solIdx, int otherIdx, std::string enumType)
Set the enumType of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherConstraintResultValue](#) (int solIdx, int otherIdx, std::string value)
Set the value of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
 - bool [setOtherConstraintResultDescription](#) (int solIdx, int otherIdx, std::string description)

Set the description of another (non-standard/solver specific) constraint-related result, for the $[i]$ th solution, where i equals the given solution index.

- bool [setOtherConstraintResultConIdx](#) (int solIdx, int otherIdx, int conIdx, int idx)
Set the index of another (non-standard/solver specific) constraint-related result, for the $[i]$ th solution, where i equals the given solution index.
- bool [setOtherConstraintResultConName](#) (int solIdx, int otherIdx, int conIdx, std::string name)
Set the name of another (non-standard/solver specific) constraint-related result, for the $[i]$ th solution, where i equals the given solution index.
- bool [setOtherConstraintResultCon](#) (int solIdx, int otherIdx, int conIdx, std::string value)
Set the value of another (non-standard/solver specific) constraint-related result, for the $[i]$ th solution, where i equals the given solution index.
- bool [setNumberOfOtherSolutionResults](#) (int solIdx, int numberOfOtherSolutionResults)
Set the $[i]$ th optimization solution's other (non-standard/solver specific) solution-related results, where i equals the given solution index.
- bool [setOtherSolutionResultName](#) (int solIdx, int otherIdx, std::string name)
Set the name associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setOtherSolutionResultValue](#) (int solIdx, int otherIdx, std::string value)
Set the value associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setOtherSolutionResultCategory](#) (int solIdx, int otherIdx, std::string category)
Set the category associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setOtherSolutionResultDescription](#) (int solIdx, int otherIdx, std::string description)
Set the description associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setOtherSolutionResultNumberOfItems](#) (int solIdx, int otherIdx, int numberOfItems)
Set the number of items associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setOtherSolutionResultItem](#) (int solIdx, int otherIdx, int itemIdx, std::string item)
Set one item associated with the $[j]$ th other solution result of solution $[i]$.
- bool [setAnotherSolutionResult](#) (int solIdx, std::string name, std::string value, std::string category, std::string description, int numberOfItems, std::string *item)
Set another solution result of solution $[i]$.
- bool [setNumberOfSolverOutputs](#) (int numberOfSolverOutputs)
Set the number of other solver outputs.
- bool [setSolverOutputName](#) (int otherIdx, std::string name)
Set the name associated with the $[j]$ th solver output.
- bool [setSolverOutputCategory](#) (int otherIdx, std::string category)
Set the category associated with the $[j]$ th solver output.
- bool [setSolverOutputDescription](#) (int otherIdx, std::string description)
Set the description associated with the $[j]$ th solver output.
- bool [setSolverOutputNumberOfItems](#) (int otherIdx, int numberOfItems)
Set the number of items associated with the $[j]$ th solver output.
- bool [setSolverOutputItem](#) (int otherIdx, int itemIdx, std::string item)
Set one item associated with the $[j]$ th solver output.

Public Attributes

- [GeneralFileHeader](#) * [resultHeader](#)
header information
- [GeneralResult](#) * [general](#)
general holds the first child of the [OSResult](#) specified by the OSrL Schema.

- [SystemResult](#) * [system](#)
system holds the second child of the [OSResult](#) specified by the OSrL Schema.
- [ServiceResult](#) * [service](#)
service holds the third child of the [OSResult](#) specified by the OSrL Schema.
- [JobResult](#) * [job](#)
job holds the fourth child of the [OSResult](#) specified by the OSrL Schema.
- [OptimizationResult](#) * [optimization](#)
optimization holds the fifth child of the [OSResult](#) specified by the OSrL Schema.
- int [m_iVariableNumber](#)
m_iVariableNumber holds the variable number.
- int [m_iObjectiveNumber](#)
m_iObjectiveNumber holds the objective number.
- int [m_iConstraintNumber](#)
m_iConstraintNumber holds the constraint number.
- int [m_iNumberOfOtherVariableResults](#)
m_iNumberOfOtherVariableResults holds the number of [OtherVariableResult](#) objects.
- double * [m_mdPrimalValues](#)
m_mdPrimalValues a vector of primal variables.
- double * [m_mdDualValues](#)
m_mdDualValues a vector of dual variables.
- std::vector< [IndexValuePair](#) * > [primalVals](#)
- std::vector< [IndexValuePair](#) * > [dualVals](#)

6.200.1 Detailed Description

The Result Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for holding all the solution information.

Definition at line 2312 of file OSResult.h.

6.200.2 Constructor & Destructor Documentation

6.200.2.1 OSResult::OSResult ()

Default constructor.

6.200.2.2 OSResult::~~OSResult ()

Class destructor.

6.200.3 Member Function Documentation

6.200.3.1 bool OSResult::setHeader (std::string *name*, std::string *source*, std::string *fileCreator*, std::string *description*, std::string *licence*)

A function to populate an instance of the result header element.

Parameters

<i>name</i> ,:	the name of this file or instance
<i>source</i> ,:	the source (e.g., in BiBTeX format)
<i>fileCreator</i> ,:	the creator of this file
<i>description</i> ,:	further description about this file and/or its contents
<i>licence</i> ,:	licence information if applicable

6.200.3.2 bool OSResult::isEqual (OSResult * *that*)

A function to check for the equality of two objects.

6.200.3.3 bool OSResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.200.3.4 GeneralStatus* OSResult::getGeneralStatus ()

Get the general status.

Returns

the general status.

6.200.3.5 std::string OSResult::getGeneralStatusType ()

Get the general status type, which can be: success, error, warning.

Returns

the general status type, null if none.

6.200.3.6 std::string OSResult::getGeneralStatusDescription ()

Get the general status description.

Returns

the general status description, null or empty std::string if none.

6.200.3.7 int OSResult::getNumberOfGeneralSubstatuses ()

Get the number of substatuses.

Returns

the number of substatuses, -1 if general status does not exist.

6.200.3.8 std::string OSResult::getGeneralSubstatusName (int *i*)

Get the *i*_th general substatus name.

Parameters

<i>i</i>	the number of the substatus (must be between 0 and numberOfSubstatuses)
----------	---

Returns

the general substatus name, null if none.

6.200.3.9 std::string OSResult::getGeneralSubstatusDescription (int *i*)

Get the *i*_th general substatus description.

Parameters

<i>i</i>	the number of the substatus (must be between 0 and numberOfSubstatuses)
----------	---

Returns

the general substatus description, null or empty std::string if none.

6.200.3.10 std::string OSResult::getGeneralMessage ()

Get the general message.

Returns

the general message.

6.200.3.11 std::string OSResult::getServiceName ()

Get service name.

Returns

the service name.

6.200.3.12 std::string OSResult::getServiceURI ()

Get service uri.

Returns

the service uri.

6.200.3.13 std::string OSResult::getInstanceName ()

Get instance name.

Returns

the instance name.

6.200.3.14 std::string OSResult::getJobID ()

Get the job id.

Returns

the job id.

6.200.3.15 std::string OSResult::getSolverInvoked ()

Get the solver invoked.

Returns

the solver invoked.

6.200.3.16 std::string OSResult::getTimeStamp ()

Get the time stamp.

Returns

the time stamp.

6.200.3.17 int OSResult::getNumberOfOtherGeneralResults ()

Get the number of other results in the <general> element.

Returns

the number of other <general> results.

6.200.3.18 std::string OSResult::getOtherGeneralResultName (int *idx*)

Get the name of the i-th other result in the <general> element.

Parameters

<i>i</i>	holds the number of the result whose name is sought.
----------	--

Returns

the name of the other <general> result.

- 6.200.3.19 `std::string OSResult::getOtherGeneralResultValue (int idx)`
- 6.200.3.20 `std::string OSResult::getOtherGeneralResultDescription (int idx)`
- 6.200.3.21 `std::string OSResult::getSystemInformation ()`
- 6.200.3.22 `std::string OSResult::getAvailableDiskSpaceUnit ()`
- 6.200.3.23 `std::string OSResult::getAvailableDiskSpaceDescription ()`
- 6.200.3.24 `double OSResult::getAvailableDiskSpaceValue ()`
- 6.200.3.25 `std::string OSResult::getAvailableMemoryUnit ()`
- 6.200.3.26 `std::string OSResult::getAvailableMemoryDescription ()`
- 6.200.3.27 `double OSResult::getAvailableMemoryValue ()`
- 6.200.3.28 `std::string OSResult::getAvailableCPUSpeedUnit ()`
- 6.200.3.29 `std::string OSResult::getAvailableCPUSpeedDescription ()`
- 6.200.3.30 `double OSResult::getAvailableCPUSpeedValue ()`
- 6.200.3.31 `std::string OSResult::getAvailableCPUNumberDescription ()`
- 6.200.3.32 `int OSResult::getAvailableCPUNumberValue ()`
- 6.200.3.33 `int OSResult::getNumberOfOtherSystemResults ()`
- 6.200.3.34 `std::string OSResult::getOtherSystemResultName (int idx)`
- 6.200.3.35 `std::string OSResult::getOtherSystemResultValue (int idx)`
- 6.200.3.36 `std::string OSResult::getOtherSystemResultDescription (int idx)`
- 6.200.3.37 `std::string OSResult::getCurrentState ()`
- 6.200.3.38 `int OSResult::getCurrentJobCount ()`
- 6.200.3.39 `int OSResult::getTotalJobsSoFar ()`
- 6.200.3.40 `std::string OSResult::getTimeServiceStarted ()`
- 6.200.3.41 `double OSResult::getServiceUtilization ()`
- 6.200.3.42 `int OSResult::getNumberOfOtherServiceResults ()`
- 6.200.3.43 `std::string OSResult::getOtherServiceResultName (int idx)`
- 6.200.3.44 `std::string OSResult::getOtherServiceResultValue (int idx)`
- 6.200.3.45 `std::string OSResult::getOtherServiceResultDescription (int idx)`

6.200.3.46 `std::string OSResult::getJobStatus ()`

6.200.3.47 `std::string OSResult::getJobSubmitTime ()`

6.200.3.48 `std::string OSResult::getScheduledStartTime ()`

6.200.3.49 `std::string OSResult::getActualStartTime ()`

6.200.3.50 `std::string OSResult::getJobEndTime ()`

6.200.3.51 `int OSResult::getTimeNumber ()`

Get the number of time measurements.

Returns

the number of time measurements

6.200.3.52 `double OSResult::getTimeValue ()`

Get the time measurement.

In the first instance, assume that there is only a single measure, which is the total elapsed time in seconds

Returns

the time measurement

6.200.3.53 `int OSResult::getNumberOfTimes ()`

6.200.3.54 `std::string OSResult::getTimingInfoUnit (int idx)`

6.200.3.55 `std::string OSResult::getTimingInfoType (int idx)`

6.200.3.56 `std::string OSResult::getTimingInfoCategory (int idx)`

6.200.3.57 `std::string OSResult::getTimingInfoDescription (int idx)`

6.200.3.58 `double OSResult::getTimingInfoValue (int idx)`

6.200.3.59 `std::string OSResult::getUsedDiskSpaceUnit ()`

6.200.3.60 `std::string OSResult::getUsedDiskSpaceDescription ()`

6.200.3.61 `double OSResult::getUsedDiskSpaceValue ()`

6.200.3.62 `std::string OSResult::getUsedMemoryUnit ()`

6.200.3.63 `std::string OSResult::getUsedMemoryDescription ()`

6.200.3.64 `double OSResult::getUsedMemoryValue ()`

6.200.3.65 `std::string OSResult::getUsedCPUSpeedUnit ()`

6.200.3.66 `std::string OSResult::getUsedCPUSpeedDescription ()`

6.200.3.67 `double OSResult::getUsedCPUSpeedValue ()`

6.200.3.68 `std::string OSResult::getUsedCPUNumberDescription ()`

6.200.3.69 `int OSResult::getUsedCPUNumberValue ()`

6.200.3.70 `int OSResult::getNumberOfOtherJobResults ()`

6.200.3.71 `std::string OSResult::getOtherJobResultName (int idx)`

6.200.3.72 `std::string OSResult::getOtherJobResultValue (int idx)`

6.200.3.73 `std::string OSResult::getOtherJobResultDescription (int idx)`

6.200.3.74 `int OSResult::getVariableNumber ()`

Get variable number.

Returns

variable number, -1 if no information.

6.200.3.75 `int OSResult::getObjectiveNumber ()`

Get objective number.

Returns

objective number, -1 if no information.

6.200.3.76 `int OSResult::getConstraintNumber ()`

Get constraint number.

Returns

constraint number, -1 if no information.

6.200.3.77 `int OSResult::getSolutionNumber ()`

get the number of solutions.

Returns

the number of solutions, 0 if none.

6.200.3.78 **OptimizationSolutionStatus*** `OSResult::getSolutionStatus (int solIdx)`

Get the [*i*]th optimization solution status, where *i* equals the given solution index.

The solution status includes the status type, optional descriptions and possibly substatus.

Parameters

<i>solIdx</i>	holds the solution index to get the solution status.
---------------	--

Returns

the optimization solution status that corresponds to *solldx*, null if none.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OptimizationSolutionStatus

6.200.3.79 std::string OSResult::getSolutionStatusType (int *solldx*)

Get the [i]th optimization solution status type, where i equals the given solution index.

The solution status type can be: unbounded, globallyOptimal, locallyOptimal, optimal, bestSoFar, feasible, infeasible, stoppedByLimit, unsure, error, other

Parameters

<i>solldx</i>	holds the solution index to get the solution status type.
---------------	---

Returns

the optimization solution status type that corresponds to *solldx*, null or empty std::string if none.

6.200.3.80 std::string OSResult::getSolutionStatusDescription (int *solldx*)

Get the [i]th optimization solution status description, where i equals the given solution index.

Parameters

<i>solldx</i>	holds the solution index to get the solution status description.
---------------	--

Returns

the optimization solution status description that corresponds to *solldx*, null or empty std::string if none.

6.200.3.81 int OSResult::getNumberOfSolutionSubstatuses (int *solldx*)**6.200.3.82 std::string OSResult::getSolutionSubstatusType (int *solldx*, int *substatusidx*)****6.200.3.83 std::string OSResult::getSolutionSubstatusDescription (int *solldx*, int *substatusidx*)****6.200.3.84 int OSResult::getSolutionTargetObjectiveIdx (int *solldx*)****6.200.3.85 std::string OSResult::getSolutionTargetObjectiveName (int *solldx*)****6.200.3.86 bool OSResult::getSolutionWeightedObjectives (int *solldx*)**

Get the [i]th optimization solution form of the objective.

Parameters

<i>solldx</i>	holds the solution index to get the solution status description.
---------------	--

Returns

whether weighting is applied to the objective.

6.200.3.87 `std::string OSResult::getSolutionMessage (int solldx)`

Get the [i]th optimization solution message, where i equals the given solution index.

Parameters

<i>solldx</i>	holds the solution index to get the solution message.
---------------	---

Returns

the optimization solution message that corresponds to *solldx*, null or empty if none.

6.200.3.88 `int OSResult::getNumberOfPrimalVariableValues (int solldx)`

6.200.3.89 `int OSResult::getNumberOfVarValues (int solldx)`

6.200.3.90 `int OSResult::getVarValueIdx (int solldx, int varldx)`

6.200.3.91 `std::string OSResult::getVarValueName (int solldx, int varldx)`

6.200.3.92 `double OSResult::getVarValue (int solldx, int varldx)`

6.200.3.93 `std::vector<IndexValuePair*> OSResult::getOptimalPrimalVariableValues (int solldx)`

Get one solution of optimal primal variable values.

Parameters

<i>solldx</i>	holds the solution index the optimal solution corresponds to.
---------------	---

Returns

a vector of variable indexes and values, an empty vector if no optimal value.

6.200.3.94 `int OSResult::getNumberOfVarValuesString (int solldx)`

6.200.3.95 `int OSResult::getVarValueStringIdx (int solldx, int varldx)`

6.200.3.96 `std::string OSResult::getVarValueStringName (int solldx, int varldx)`

6.200.3.97 `std::string OSResult::getVarValueString (int solldx, int varldx)`

6.200.3.98 `int OSResult::getBasisStatusNumberOfEl (int solldx, int object, int status)`

Get the number of indices that belong to a particular basis status.

Parameters

<i>solldx</i>	holds the solution index for the current solution
<i>object</i>	describes the kind of indices to be retrieved (legal values are described in <code>ENUM_BASIS_STATUS</code> — see OSGeneral.h)
<i>status</i>	gives the basis status type

6.200.3.99 `int OSResult::getBasisStatusEl (int solldx, int object, int status, int j)`

Get an entry in the array of indices that belong to a particular basis status.

Parameters

<i>solldx</i>	holds the solution index for the current solution
<i>object</i>	describes the kind of indices to be retrieved (legal values are described in ENUM_BASIS_STATUS — see OSGeneral.h)
<i>status</i>	gives the basis status (basic, atLower, atUpper, etc.)
<i>j</i>	is the (zero-based) position of the desired entry within the index array

6.200.3.100 `int OSResult::getBasisInformationDense (int solldx, int object, int * resultArray, int dim)`

Get the basis information associated with the variables, objectives or constraints for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>object</i>	describes the kind of indices to be retrieved (legal values are described in ENUM_PROBLEM_COMPONENT — see OSGeneral.h)
<i>resultArray</i>	is the array that returns the basis information
<i>dim</i>	is the dimension of the resultArray

Returns

whether the operation was successful: < 0: error condition = 0: no data encountered
0: success

6.200.3.101 `int OSResult::getNumberOfOtherVariableResults (int solldx)`

Get numberOfOtherVariableResult.

Returns

numberOfOtherVariableResult, -1 if no information.

6.200.3.102 `int OSResult::getAnOtherVariableResultNumberOfVar (int solldx, int iOther)`

Get getAnOtherVariableResultNumberOfVar.

Returns

the number of variables in the i'th other variable result, -1 if no information.

6.200.3.103 `std::string OSResult::getOtherVariableResultName (int solldx, int otherldx)`

6.200.3.104 `std::string OSResult::getOtherVariableResultType (int solldx, int otherldx)`

6.200.3.105 `std::string OSResult::getOtherVariableResultValue (int solldx, int otherldx)`

6.200.3.106 `std::string OSResult::getOtherVariableResultDescription (int solldx, int otherldx)`

6.200.3.107 `int OSResult::getOtherVariableResultNumberOfVar (int solldx, int otherldx)`

6.200.3.108 `int OSResult::getOtherVariableResultNumberOfEnumerations (int solldx, int otherldx)`

6.200.3.109 `int OSResult::getOtherVariableResultVarldx (int solldx, int otherldx, int varldx)`

6.200.3.110 `std::string OSResult::getOtherVariableResultVar (int solldx, int otherldx, int varldx)`

6.200.3.111 `std::string OSResult::getOtherVariableResultArrayType (int solldx, int otherldx)`

Get the type of values contained in the *or* `<enumeration>` array associated with an `<other>` result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherldx</i>	is the index of the current <code><other></code> result

Returns

the array type

6.200.3.112 `std::string OSResult::getOtherVariableResultEnumerationValue (int solldx, int otherldx, int enumldx)`

Get the value of an enum associated with an `<other>` result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherldx</i>	is the index of the current <code><other></code> result is the index of the current enumeration level

Returns

the value

6.200.3.113 `std::string OSResult::getOtherVariableResultEnumerationDescription (int solldx, int otherldx, int enumldx)`

Get the description of an enum associated with an `<other>` result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherldx</i>	is the index of the current <code><other></code> result is the index of the current enumeration level

Returns

the description

6.200.3.114 `int OSResult::getOtherVariableResultEnumerationNumberOfEl (int solldx, int otherldx, int enumldx)`

Get the size of an enum associated with an `<other>` result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherldx</i>	is the index of the current <code><other></code> result is the index of the current enumeration level

Returns

the number of indices that assume this level

6.200.3.115 `int OSResult::getOtherVariableResultEnumerationEl (int solldx, int otherldx, int enumldx, int j)`

Get one index of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result
<i>enumldx</i>	is the index of the current enumeration level
<i>j</i>	is the (zero-based) position of the index within the index array

Returns

the array of indices

6.200.3.116 `int OSResult::getOtherVariableResultArrayDense (int solldx, int otherldx, std::string * resultArray, int dim)`

Get the values of a *array* or an <enumeration> associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result
<i>resultArray</i>	is the array that returns the content of the <i>or</i> <enumeration> array
<i>dim</i>	is the array dimension

Returns

whether the operation was successful: < 0: error condition = 0: no data encountered

0: number of data items set

6.200.3.117 `int OSResult::getNumberOfObjValues (int solldx)`

6.200.3.118 `int OSResult::getObjValueldx (int solldx, int objldx)`

6.200.3.119 `std::string OSResult::getObjValueName (int solldx, int objldx)`

6.200.3.120 `double OSResult::getObjValue (int solldx, int objldx)`

6.200.3.121 `double OSResult::getOptimalObjValue (int objldx, int solldx)`

Get one solution's optimal objective value.

Parameters

<i>objldx</i>	holds the objective index the optimal value corresponds to.
<i>solldx</i>	holds the solution index the optimal value corresponds to.

Returns

a double with the optimal objective function value.

6.200.3.122 `int OSResult::getNumberOfOtherObjectiveResults (int solldx)`

6.200.3.123 `std::string OSResult::getOtherObjectiveResultName (int solldx, int otherldx)`

6.200.3.124 `std::string OSResult::getOtherObjectiveResultType (int solldx, int otherldx)`

6.200.3.125 `std::string OSResult::getOtherObjectiveResultValue (int solldx, int otherldx)`

6.200.3.126 `std::string OSResult::getOtherObjectiveResultDescription (int solldx, int otherldx)`

6.200.3.127 `int OSResult::getOtherObjectiveResultNumberOfObj (int solldx, int otherldx)`

6.200.3.128 `int OSResult::getOtherObjectiveResultNumberOfEnumerations (int solldx, int otherldx)`

6.200.3.129 `int OSResult::getOtherObjectiveResultObjIdx (int solldx, int otherldx, int objldx)`

6.200.3.130 `std::string OSResult::getOtherObjectiveResultObj (int solldx, int otherldx, int objldx)`

6.200.3.131 `std::string OSResult::getOtherObjectiveResultArrayType (int solldx, int otherldx)`

Get the type of values contained in the <obj> or <enumeration> array associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result

Returns

the array type

6.200.3.132 `std::string OSResult::getOtherObjectiveResultEnumerationValue (int solldx, int otherldx, int enumldx)`

Get the value of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the value

6.200.3.133 `std::string OSResult::getOtherObjectiveResultEnumerationDescription (int solldx, int otherldx, int enumldx)`

Get the description of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the description

6.200.3.134 `int OSResult::getOtherObjectiveResultEnumerationNumberOfEI (int solldx, int otherldx, int enumldx)`

Get the size of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the number of indices that assume this level

6.200.3.135 `int OSResult::getOtherObjectiveResultEnumerationEI (int solldx, int otherldx, int enumldx, int j)`

Get one index of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level
<i>j</i>	is the (zero-based) position of the index in the array

Returns

the array of indices

6.200.3.136 `int OSResult::getOtherObjectiveResultArrayDense (int solldx, int otherldx, std::string * resultArray, int dim)`

Get the values of an <obj> array or an <enumeration> associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result
<i>resultArray</i>	is the array that returns the content of the <obj> or <enumeration> array
<i>dim</i>	is the array dimension

Returns

whether the operation was successful: < 0: error condition = 0: no data encountered
0: number of data items set

6.200.3.137 `int OSResult::getNumberOfDualValues (int solldx)`

6.200.3.138 `int OSResult::getDualValueIdx (int solldx, int conldx)`

6.200.3.139 `std::string OSResult::getDualValueName (int solldx, int objldx)`

6.200.3.140 `double OSResult::getDualValue (int solldx, int conldx)`

6.200.3.141 `std::vector<IndexValuePair*> OSResult::getOptimalDualVariableValues (int solldx)`

Get one solution of optimal dual variable values.

Parameters

<i>solldx</i>	holds the solution index the optimal solution corresponds to.
---------------	---

Returns

a vector of variable indexes and values, an empty vector if no optimal value.

6.200.3.142 `int OSResult::getNumberOfOtherConstraintResults (int solldx)`

6.200.3.143 `std::string OSResult::getOtherConstraintResultName (int solldx, int otherldx)`

6.200.3.144 `std::string OSResult::getOtherConstraintResultType (int solldx, int otherldx)`

6.200.3.145 `std::string OSResult::getOtherConstraintResultValue (int solldx, int otherldx)`

6.200.3.146 `std::string OSResult::getOtherConstraintResultDescription (int solldx, int otherldx)`

6.200.3.147 `int OSResult::getOtherConstraintResultNumberOfCon (int solldx, int otherldx)`

6.200.3.148 `int OSResult::getOtherConstraintResultNumberOfEnumerations (int solldx, int otherldx)`

6.200.3.149 `int OSResult::getOtherConstraintResultConldx (int solldx, int otherldx, int conldx)`

6.200.3.150 `std::string OSResult::getOtherConstraintResultCon (int solldx, int otherldx, int conldx)`

6.200.3.151 `std::string OSResult::getOtherConstraintResultArrayType (int solldx, int otherldx)`

Get the type of values contained in the <con> or <enumeration> array associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result

Returns

the array type

6.200.3.152 `std::string OSResult::getOtherConstraintResultEnumerationValue (int solldx, int otherldx, int enumldx)`

Get the value of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result
<i>enumldx</i>	is the index of the current enumeration level

Returns

the value

6.200.3.153 `std::string OSResult::getOtherConstraintResultEnumerationDescription (int solldx, int otherldx, int enumldx)`

Get the description of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the description

6.200.3.154 `int OSResult::getOtherConstraintResultEnumerationNumberOfEl (int solldx, int otherldx, int enumldx)`

Get the size of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the number of indices that assume this level

6.200.3.155 `int OSResult::getOtherConstraintResultEnumerationEl (int solldx, int otherldx, int enumldx, int j)`

Get one index of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result is the index of the current enumeration level
<i>j</i>	is the (zero-based) position of the entry in the array

Returns

the array of indices

6.200.3.156 `int OSResult::getOtherConstraintResultArrayDense (int solldx, int otherldx, std::string * resultArray, int dim)`

Get the values of a <con> array or an <enumeration> associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIndex</i>	is the index of the current <other> result
<i>resultArray</i>	is the array that returns the content of the <con> or <enumeration> array
<i>dim</i>	is the array dimension

Returns

whether the operation was successful: < 0: error condition = 0: no data encountered
0: number of data items set

- 6.200.3.157 `int OSResult::getNumberOfOtherSolutionResults (int solldx)`
- 6.200.3.158 `std::string OSResult::getOtherSolutionResultName (int solldx, int otherldx)`
- 6.200.3.159 `std::string OSResult::getOtherSolutionResultValue (int solldx, int otherldx)`
- 6.200.3.160 `std::string OSResult::getOtherSolutionResultCategory (int solldx, int otherldx)`
- 6.200.3.161 `std::string OSResult::getOtherSolutionResultDescription (int solldx, int otherldx)`
- 6.200.3.162 `int OSResult::getOtherSolutionResultNumberOfItems (int solldx, int otherldx)`
- 6.200.3.163 `std::string OSResult::getOtherSolutionResultItem (int solldx, int otherldx, int itemldx)`
- 6.200.3.164 `int OSResult::getNumberOfSolverOutputs ()`
- 6.200.3.165 `std::string OSResult::getSolverOutputName (int otherldx)`
- 6.200.3.166 `std::string OSResult::getSolverOutputCategory (int otherldx)`
- 6.200.3.167 `std::string OSResult::getSolverOutputDescription (int otherldx)`
- 6.200.3.168 `int OSResult::getSolverOutputNumberOfItems (int otherldx)`
- 6.200.3.169 `std::string OSResult::getSolverOutputItem (int otherldx, int itemldx)`
- 6.200.3.170 `bool OSResult::setGeneralStatus (GeneralStatus * status)`

Set the general status.

Parameters

<i>status</i>	holds the general status.
---------------	---------------------------

Returns

whether the general status is set successfully.

- 6.200.3.171 `bool OSResult::setGeneralStatusType (std::string type)`

Set the general status type, which can be: success, error, warning.

Parameters

<i>type</i>	holds the general status type
-------------	-------------------------------

Returns

whether the general status type is set successfully or not.

6.200.3.172 `bool OSResult::setNumberOfGeneralSubstatuses (int num)`

Set the number of substatus elements.

Parameters

<i>num</i>	holds the number of substatuses (a nonegative integer)
------------	--

Returns

whether the number of substatuses is set successfully or not.

6.200.3.173 `bool OSResult::setGeneralStatusDescription (std::string description)`

Set the general status description.

Parameters

<i>description</i>	holds the general status description.
--------------------	---------------------------------------

Returns

whether the general status description is set successfully or not.

6.200.3.174 `bool OSResult::setGeneralSubstatusName (int idx, std::string name)`

Set the general substatus name.

Parameters

<i>name</i>	holds the general substatus name
<i>idx</i>	holds the index of the substatus in the array

Returns

whether the general substatus name is set successfully or not.

6.200.3.175 `bool OSResult::setGeneralSubstatusDescription (int idx, std::string description)`

Set the general substatus description.

Parameters

<i>description</i>	holds the general substatus description.
<i>idx</i>	holds the index of the substatus in the array

Returns

whether the general status description is set successfully or not.

6.200.3.176 `bool OSResult::setGeneralMessage (std::string message)`

Set the general message.

Parameters

<i>message</i>	holds the general message.
----------------	----------------------------

Returns

whether process message is set successfully.

6.200.3.177 `bool OSResult::setServiceName (std::string serviceName)`

Set service name.

Parameters

<i>serviceName</i>	holds the name of the service.
--------------------	--------------------------------

Returns

whether the service name is set successfully.

6.200.3.178 `bool OSResult::setServiceURI (std::string serviceURI)`

Set service uri.

Parameters

<i>serviceURI</i>	holds the uri of the service.
-------------------	-------------------------------

Returns

whether the service uri is set successfully.

6.200.3.179 `bool OSResult::setInstanceName (std::string instanceName)`

Set instance name.

Parameters

<i>instanceName</i>	holds the name of the instance.
---------------------	---------------------------------

Returns

whether the instance name is set successfully.

6.200.3.180 `bool OSResult::setJobID (std::string jobID)`

Set job id.

Parameters

<i>jobID</i>	holds the job id.
--------------	-------------------

Returns

whether the job id is set successfully.

6.200.3.181 `bool OSResult::setSolverInvoked (std::string solverInvoked)`

Set solver invoked.

Parameters

<i>solverInvoked</i>	holds the solver invoked.
----------------------	---------------------------

Returns

whether the solver invoked is set successfully.

6.200.3.182 `bool OSResult::setTimeStamp (std::string timeStamp)`

Set time stamp.

Parameters

<i>time</i>	holds the time stamp.
-------------	-----------------------

Returns

whether the time stamp is set successfully.

6.200.3.183 `bool OSResult::setNumberOfOtherGeneralResults (int num)`

Set number of other general results.

Parameters

<i>num</i>	holds the number of other general results.
------------	--

Returns

whether the number was set successfully.

6.200.3.184 `bool OSResult::setOtherGeneralResultName (int idx, std::string name)`

Set the general otherResult name.

Parameters

<i>name</i>	holds the general otherResult name
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the general otherResult name is set successfully or not.

6.200.3.185 `bool OSResult::setOtherGeneralResultValue (int idx, std::string value)`

Set the general otherResult value.

Parameters

<i>name</i>	holds the general otherResult value
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the general otherResult value is set successfully or not.

6.200.3.186 `bool OSResult::setOtherGeneralResultDescription (int idx, std::string description)`

Set the general otherResult description.

Parameters

<i>name</i>	holds the general otherResult description
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the general otherResult description is set successfully or not.

6.200.3.187 `bool OSResult::setSystemInformation (std::string systemInformation)`

Set the system information.

Parameters

<i>system- Information</i>	holds the system information
--------------------------------	------------------------------

Returns

whether the system information was set successfully or not.

6.200.3.188 `bool OSResult::setAvailableDiskSpaceUnit (std::string unit)`

Set the unit in which available disk space is measured.

Parameters

<i>unit</i>	holds unit (byte, kilobyte, megabyte, gigabyte, terabyte, petabyte)
-------------	---

Returns

whether the system information was set successfully or not.

6.200.3.189 `bool OSResult::setAvailableDiskSpaceDescription (std::string description)`

Set the description of available disk space.

Parameters

<i>description</i>	holds further information about available disk space
--------------------	--

Returns

whether the system information was set successfully or not.

6.200.3.190 `bool OSResult::setAvailableDiskSpaceValue (double value)`

Set the amount of available disk space.

Parameters

<i>value</i>	holds the number of disk space units
--------------	--------------------------------------

Returns

whether the system information was set successfully or not.

6.200.3.191 `bool OSResult::setAvailableMemoryUnit (std::string unit)`

Set the unit in which available memory is measured.

Parameters

<i>unit</i>	holds unit (byte, kilobyte, megabyte, gigabyte, terabyte)
-------------	---

Returns

whether the system information was set successfully or not.

6.200.3.192 `bool OSResult::setAvailableMemoryDescription (std::string description)`

Set the description of available memory.

Parameters

<i>description</i>	holds further information about available memory
--------------------	--

Returns

whether the system information was set successfully or not.

6.200.3.193 `bool OSResult::setAvailableMemoryValue (double value)`

Set the amount of available memory.

Parameters

<i>value</i>	holds the number of memory units
--------------	----------------------------------

Returns

whether the system information was set successfully or not.

6.200.3.194 bool OSResult::setAvailableCPUSpeedUnit (std::string *unit*)

Set the unit in which available CPU speed is measured.

Parameters

<i>unit</i>	holds unit
-------------	------------

Returns

whether the system information was set successfully or not.

6.200.3.195 bool OSResult::setAvailableCPUSpeedDescription (std::string *description*)

Set the description of available CPU speed.

Parameters

<i>description</i>	holds further information about the CPU speed
--------------------	---

Returns

whether the system information was set successfully or not.

6.200.3.196 bool OSResult::setAvailableCPUSpeedValue (double *value*)

Set the available CPU speed.

Parameters

<i>value</i>	holds the available CPU speed
--------------	-------------------------------

Returns

whether the system information was set successfully or not.

6.200.3.197 bool OSResult::setAvailableCPUNumberDescription (std::string *description*)

Set the description of available number of CPUs.

Parameters

<i>description</i>	is used to impart further info about the CPUs
--------------------	---

Returns

whether the system information was set successfully or not.

6.200.3.198 bool OSResult::setAvailableCPUNumberValue (int *value*)

Set the available number of CPUs.

Parameters

<i>value</i>	holds the available number of CPUs
--------------	------------------------------------

Returns

whether the system information was set successfully or not.

6.200.3.199 bool OSResult::setNumberOfOtherSystemResults (int *num*)

Set number of other system results.

Parameters

<i>num</i>	holds the number of other system results.
------------	---

Returns

whether the number was set successfully.

6.200.3.200 bool OSResult::setOtherSystemResultName (int *idx*, std::string *name*)

Set the system otherResult name.

Parameters

<i>name</i>	holds the system otherResult name
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the system otherResult name is set successfully or not.

6.200.3.201 bool OSResult::setOtherSystemResultValue (int *idx*, std::string *value*)

Set the system otherResult value.

Parameters

<i>name</i>	holds the system otherResult value
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the system otherResult value is set successfully or not.

6.200.3.202 `bool OSResult::setOtherSystemResultDescription (int idx, std::string description)`

Set the system otherResult description.

Parameters

<i>name</i>	holds the system otherResult description
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the system otherResult description is set successfully or not.

6.200.3.203 `bool OSResult::setCurrentState (std::string currentState)`

Set the current state of the service.

Parameters

<i>currentState</i>	holds the current state
---------------------	-------------------------

Returns

whether the service information was set successfully or not.

6.200.3.204 `bool OSResult::setCurrentJobCount (int jobCount)`

Set the current job count.

Parameters

<i>jobCount</i>	holds the current job count
-----------------	-----------------------------

Returns

whether the service information was set successfully or not.

6.200.3.205 `bool OSResult::setTotalJobsSoFar (int number)`

Set the total number of jobs so far.

Parameters

<i>number</i>	holds the total number of jobs
---------------	--------------------------------

Returns

whether the service information was set successfully or not.

6.200.3.206 `bool OSResult::setTimeServiceStarted (std::string startTime)`

Set the time the service was started.

Parameters

<i>startTime</i>	holds the starting time
------------------	-------------------------

Returns

whether the service information was set successfully or not.

6.200.3.207 bool OSResult::setServiceUtilization (double *value*)

Set the service utilization.

Parameters

<i>value</i>	holds the service utilization
--------------	-------------------------------

Returns

whether the service information was set successfully or not.

6.200.3.208 bool OSResult::setNumberOfOtherServiceResults (int *num*)

Set number of other service results.

Parameters

<i>num</i>	holds the number of other service results.
------------	--

Returns

whether the number was set successfully.

6.200.3.209 bool OSResult::setOtherServiceResultName (int *idx*, std::string *name*)

Set the service otherResult name.

Parameters

<i>name</i>	holds the service otherResult name
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the service otherResult name is set successfully or not.

6.200.3.210 bool OSResult::setOtherServiceResultValue (int *idx*, std::string *value*)

Set the service otherResult value.

Parameters

<i>name</i>	holds the service otherResult value
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the service otherResult value is set successfully or not.

6.200.3.211 bool OSResult::setOtherServiceResultDescription (int *idx*, std::string *description*)

Set the service otherResult description.

Parameters

<i>name</i>	holds the service otherResult description
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the service otherResult description is set successfully or not.

6.200.3.212 bool OSResult::setJobStatus (std::string *status*)

Set the job status.

Parameters

<i>status</i>	holds the job status
---------------	----------------------

Returns

whether the job status was set successfully or not.

6.200.3.213 bool OSResult::setJobSubmitTime (std::string *submitTime*)

Set the time when the job was submitted.

Parameters

<i>submitTime</i>	holds the time when the job was submitted
-------------------	---

Returns

whether the information was set successfully or not.

6.200.3.214 bool OSResult::setScheduledStartTime (std::string *scheduledStartTime*)

Set the job's scheduled start time.

Parameters

<i>scheduledStart- Time</i>	holds the scheduled start time
---------------------------------	--------------------------------

Returns

whether the information was set successfully or not.

6.200.3.215 `bool OSResult::setActualStartTime (std::string actualStartTime)`

Set the job's actual start time.

Parameters

<i>actualStartTime</i>	holds the actual start time
------------------------	-----------------------------

Returns

whether the information was set successfully or not.

6.200.3.216 `bool OSResult::setJobEndTime (std::string endTime)`

Set the time when the job finished.

Parameters

<i>endTime</i>	holds the time when the job finished
----------------	--------------------------------------

Returns

whether the information was set successfully or not.

6.200.3.217 `bool OSResult::setTime (double time)`

Set time.

Parameters

<i>time</i>	holds the time.
-------------	-----------------

Returns

whether the time is set successfully.

6.200.3.218 `bool OSResult::addTimingInformation (std::string type, std::string category, std::string unit, std::string description, double value)`

Add timing information.

Parameters

<i>type</i>	holds the timer type (cpuTime/elapsedTime/other).
<i>category</i>	holds the timer category (total/input/preprocessing, etc.)
<i>unit</i>	holds the timer unit (tick/milliscond/second/minute/etc.)
<i>description</i>	holds further information about the timer.
<i>value</i>	holds the time measurement.

Returns

whether the time is set successfully.

6.200.3.219 `bool OSResult::setTimingInformation (int idx, std::string type, std::string category, std::string unit, std::string description, double value)`

Set timing information.

Parameters

<i>idx</i>	holds the index within the time array of the item to be set
<i>type</i>	holds the timer type (cpuTime/elapsedTime/other).
<i>category</i>	holds the timer category (total/input/preprocessing, etc.)
<i>unit</i>	holds the timer unit (tick/milliscond/second/minute/etc.)
<i>description</i>	holds further information about the timer.
<i>value</i>	holds the time measurement.

Returns

whether the time is set successfully.

6.200.3.220 `bool OSResult::setNumberOfTimes (int numberOfTimes)`

Set the number of time measurements and initial the time array.

Parameters

<i>numberOfTimes</i>	holds the number of measurements
----------------------	----------------------------------

Returns

whether the function completed successfully or not.

6.200.3.221 `bool OSResult::setTimeNumber (int timeNumber)`

Set the number of time measurements.

Parameters

<i>timeNumber</i>	holds the number of measurements
-------------------	----------------------------------

Returns

whether the time number is set successfully or not.

6.200.3.222 `bool OSResult::setUsedDiskSpaceUnit (std::string unit)`

Set the unit in which used disk space is measured.

Parameters

<i>unit</i>	holds unit (byte, kilobyte, megabyte, gigabyte, terabyte, petabyte)
-------------	---

Returns

whether the information was set successfully or not.

6.200.3.223 `bool OSResult::setUsedDiskSpaceDescription (std::string description)`

Set the description of used disk space.

Parameters

<i>description</i>	holds further information about used disk space
--------------------	---

Returns

whether the information was set successfully or not.

6.200.3.224 `bool OSResult::setUsedDiskSpaceValue (double value)`

Set the amount of used disk space.

Parameters

<i>value</i>	holds the number of disk space units
--------------	--------------------------------------

Returns

whether the information was set successfully or not.

6.200.3.225 `bool OSResult::setUsedMemoryUnit (std::string unit)`

Set the unit in which used memory is measured.

Parameters

<i>unit</i>	holds unit (byte, kilobyte, megabyte, gigabyte, terabyte)
-------------	---

Returns

whether the information was set successfully or not.

6.200.3.226 `bool OSResult::setUsedMemoryDescription (std::string description)`

Set the description of used memory.

Parameters

<i>description</i>	holds further information about used memory
--------------------	---

Returns

whether the information was set successfully or not.

6.200.3.227 `bool OSResult::setUsedMemoryValue (double value)`

Set the amount of used memory.

Parameters

<i>value</i>	holds the number of memory units
--------------	----------------------------------

Returns

whether the information was set successfully or not.

6.200.3.228 `bool OSResult::setUsedCPUSpeedUnit (std::string unit)`

Set the unit in which used CPU speed is measured.

Parameters

<i>unit</i>	holds unit
-------------	------------

Returns

whether the information was set successfully or not.

6.200.3.229 `bool OSResult::setUsedCPUSpeedDescription (std::string description)`

Set the description of used CPU speed.

Parameters

<i>description</i>	holds further information about the CPU speed
--------------------	---

Returns

whether the information was set successfully or not.

6.200.3.230 `bool OSResult::setUsedCPUSpeedValue (double value)`

Set the used CPU speed.

Parameters

<i>value</i>	holds the used CPU speed
--------------	--------------------------

Returns

whether the information was set successfully or not.

6.200.3.231 `bool OSResult::setUsedCPUNumberDescription (std::string description)`

Set the description of used number of CPUs.

Parameters

<i>description</i>	is used to impart further info about the CPUs
--------------------	---

Returns

whether the system information was set successfully or not.

6.200.3.232 `bool OSResult::setUsedCPUNumberValue (int value)`

Set the used number of CPUs.

Parameters

<i>value</i>	holds the used number of CPUs
--------------	-------------------------------

Returns

whether the information was set successfully or not.

6.200.3.233 `bool OSResult::setNumberOfOtherJobResults (int num)`

Set number of other job results.

Parameters

<i>num</i>	holds the number of other job results.
------------	--

Returns

whether the number was set successfully.

6.200.3.234 `bool OSResult::setOtherJobResultName (int idx, std::string name)`

Set the job otherResult name.

Parameters

<i>name</i>	holds the job otherResult name
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the job otherResult name is set successfully or not.

6.200.3.235 `bool OSResult::setOtherJobResultValue (int idx, std::string value)`

Set the job otherResult value.

Parameters

<i>name</i>	holds the job otherResult value
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the job otherResult value is set successfully or not.

6.200.3.236 `bool OSResult::setOtherJobResultDescription (int idx, std::string description)`

Set the job otherResult description.

Parameters

<i>name</i>	holds the job otherResult description
<i>idx</i>	holds the index of the otherResult in the array

Returns

whether the job otherResult description is set successfully or not.

6.200.3.237 `bool OSResult::setVariableNumber (int variableNumber)`

Set the variable number.

Parameters

<i>variableNumber</i>	holds the number of variables
-----------------------	-------------------------------

Returns

whether the variable number is set successfully or not.

6.200.3.238 `bool OSResult::setObjectiveNumber (int objectiveNumber)`

Set the objective number.

Parameters

<i>objectiveNumber</i>	holds the number of objectives
------------------------	--------------------------------

Returns

whether the objective number is set successfully or not.

6.200.3.239 `bool OSResult::setConstraintNumber (int constraintNumber)`

Set the constraint number.

Parameters

<i>constraint- Number</i>	holds the number of constraints
-------------------------------	---------------------------------

Returns

whether the constraint number is set successfully or not.

6.200.3.240 `bool OSResult::setSolutionNumber (int number)`

set the number of solutions.

This method must be called before setting other optimization solution related results. Before this method is called, the [setVariableNumber\(int\)](#), [setObjectiveNumber\(int\)](#), [setConstraintNumber\(int\)](#) methods have to be called first.

Parameters

<i>number</i>	holds the number of solutions to set.
---------------	---------------------------------------

Returns

whether the solution number is set successfully.

See Also

[setVariableNumber\(int\)](#)
[setObjectiveNumber\(int\)](#)
[setConstraintNumber\(int\)](#)

6.200.3.241 `bool OSResult::setSolutionStatus (int solldx, std::string type, std::string description)`

Set the [i]th optimization solution status, where i equals the given solution index.

The solution status includes the status type, optional descriptions and possibly substatuses. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the solution status.
<i>status</i>	holds the optimization solution status to set.

Returns

whether the optimization solution status is set successfully or not.

See Also

`org.optimizationservices.oscommon.datastructure.osresult.OptimizationSolutionStatus`
[setSolutionNumber\(int\)](#)

6.200.3.242 `bool OSResult::setSolutionStatusType (int solldx, std::string type)`

Set the [i]th optimization solution status type.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>type</i>	holds the solution status type

Returns

whether the solution status type is set successfully or not.

6.200.3.243 `bool OSResult::setNumberOfSolutionSubstatus (int solldx, int num)`

Set the [i]th optimization solution's number of substatus elements.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>num</i>	holds the number of substatuses (a nonegative integer)

Returns

whether the number of substatuses is set successfully or not.

6.200.3.244 `bool OSResult::setSolutionStatusDescription (int solldx, std::string description)`

Set the [i]th optimization solution status description.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>description</i>	holds the solution status description.

Returns

whether the solution status description is set successfully or not.

6.200.3.245 `bool OSResult::setSolutionSubstatusType (int solldx, int substatusidx, std::string type)`

Set the solution substatus type.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>substatusidx</i>	holds the index of the substatus in the array
<i>type</i>	holds the general substatus type

Returns

whether the general substatus type is set successfully or not.

6.200.3.246 `bool OSResult::setSolutionSubstatusDescription (int solldx, int substatusidx, std::string description)`

Set the solution substatus description.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>substatusidx</i>	holds the index of the substatus in the array
<i>description</i>	holds the general substatus description.

Returns

whether the solution status description is set successfully or not.

6.200.3.247 `bool OSResult::setSolutionTargetObjectiveldx (int solldx, int objectiveldx)`

Set the [i]th optimization solution's objective index, where i equals the given solution index.

The first objective's index should be -1, the second -2, and so on. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the objective index.
<i>objectvelidx</i>	holds the objective index to set. All the objective indexes are negative starting from -1 downward.

Returns

whether the optimization objective index is set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.248 `bool OSResult::setSolutionTargetObjectiveName (int solldx, std::string objectiveName)`

Set the [i]th optimization solution's objective name, where i equals the given solution index.

The first objective's index should be -1, the second -2, and so on. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the objective index.
<i>objectiveName</i>	holds the objective indexname to set.

Returns

whether the optimization objective name is set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.249 `bool OSResult::setSolutionWeightedObjectives (int solldx, bool weightedObjectives)`

Record whether the [i]th optimization solution uses weighted objectives, where i equals the given solution index.

Parameters

<i>solldx</i>	holds the solution index to set the objective index.
<i>weighted-Objectives</i>	holds the value "true" or "false".

Returns

whether the information was set successfully or not.

6.200.3.250 `bool OSResult::setSolutionMessage (int solldx, std::string msg)`

Set the [i]th optimization solution's message, where i equals the given solution index.

The first objective's index should be -1, the second -2, and so on. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the objective index.
<i>msg</i>	holds the solution message to set.

Returns

whether the optimization objective index is set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.251 `bool OSResult::setNumberOfPrimalVariableValues (int solIdx, int n)`

Set the [i]th optimization solution's number of primal variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>n</i>	holds the number of elements in the array x

Returns

whether primal variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.252 `bool OSResult::setPrimalVariableValuesSparse (int solIdx, std::vector< IndexValuePair * > x)`

Set the [i]th optimization solution's primal variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>x</i>	holds a vector of type IndexValuePair ; the idx component holds the index of the variable; the value component holds its value. The vector could be null if all variables are 0.

Returns

whether primal variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.253 `bool OSResult::setPrimalVariableValuesDense (int solIdx, double * x)`

Set the [i]th optimization solution's primal variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>x</i>	holds a double dense array of variable values to set; it could be null if all variables are 0.

Returns

whether primal variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.254 `bool OSResult::setNumberOfVarValues (int solIdx, int numberOfVar)`

Set the number of primal variables to be given a value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>numberOfVar</i>	holds the number of primal variables that are to be set

Returns

whether the information was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.255 `bool OSResult::setVarValue (int solIdx, int number, int idx, std::string name, double val)`

Set a primal variable value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>number</i>	holds the location within the sparse array var that is to be used
<i>idx</i>	holds the index of the primal variable that is to be set
<i>name</i>	holds the variable name (or an empty string).
<i>val</i>	holds the variable value to set.

Returns

whether primal variable value was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.256 `bool OSResult::setNumberOfVarValuesString (int solIdx, int numberOfVar)`

Set the number of string-valued primal variables to be given a value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>numberOfVar</i>	holds the number of primal variables that are to be set

Returns

whether the information was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.257 `bool OSResult::setVarValueString (int solIdx, int number, int idx, std::string name, std::string str)`

Set a string-valued primal variable value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the primal variable values.
<i>number</i>	holds the location within the sparse array var that is to be used
<i>idx</i>	holds the index of the primal variable that is to be set
<i>name</i>	holds the variable name (or an empty string).
<i>str</i>	holds the variable value to set.

Returns

whether primal variable value was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.258 `bool OSResult::setBasisStatus (int solIdx, int object, int status, int * i, int ni)`

Set the basis status of a number of variables/constraints/objectives.

Parameters

<i>solldx</i>	holds the index of the solution to which the basis values belong.
<i>object</i>	holds the type of basis object to be used (legal values are taken from the ENUM_PROBLEM_COMPONENT enumeration — see OSGeneral.h))
<i>status</i>	holds the status which is to be used (legal values are taken from the ENUM_BASIS_STATUS enumeration — see OSGeneral.h)
<i>i</i>	holds the integer array whose values are to be transferred. (NOTE WELL: This method does not handle individual variables --- the entire basis must be processed)
<i>ni</i>	holds the number of elements of i

Returns

whether basis status was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.259 `bool OSResult::setNumberOfOtherVariableResults (int solldx, int numberOfOtherVariableResults)`

Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first. This method then allocates the memory for the new [OtherVariableResult](#) objects

Parameters

<i>solldx</i>	is the solution index
<i>numberOfOther-VariableResult</i>	holds the number of OtherVariableResult objects Each other variable result contains the name (required), an optional description (std::string) and an optional value (std::string). Each other variable result can also optionally contain an array OtherVarResult for each variable. The Other-VarResult contains a variable idx (required) and an optional std::string value.

Returns

whether the other variable results are set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.260 `bool OSResult::setAnOtherVariableResultSparse (int solldx, int otherIdx, std::string name, std::string value, std::string description, int * idx, std::string * s, int n)`

Set the [i]th optimization solution's other (non-standard/solver specific) variable-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the new OtherVariableResult object
<i>name</i>	holds the name of the other element
<i>value</i>	holds the value of the other element
<i>idx</i>	holds a pointer to the indexes of the var element
<i>s</i>	holds a pointer to the array of values of the var element
<i>n</i>	holds the number of elements of the array

Returns

whether the other variable results are set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.261 `bool OSResult::setAnOtherVariableResultSparse (int solIdx, int otherIdx, std::string name, std::string value, std::string description, int * idx, std::string * s, int n, std::string type, std::string varType, std::string enumType)`

Set the [i]th optimization solution's other (non-standard/solver specific)variable-related results, where i equals the given solution index.

This alternate signature sets the type of the value attribute and the *and* `<enumeration>` arrays

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	<i>holds the solution index</i>
<i>otherIdx</i>	<i>holds the index of the new OtherVariableResult object</i>
<i>name</i>	<i>holds the name of the other element</i>
<i>value</i>	<i>holds the value of the other element</i>
<i>idx</i>	<i>holds a pointer to the indexes of the var element</i>
<i>s</i>	<i>holds a pointer to the array of values of the var element</i>
<i>n</i>	<i>holds the number of elements of the array</i>
<i>type</i>	<i>holds the type of the <code><other></code> element's value attribute</i>
<i>varType</i>	<i>holds the type of the <code><other></code> element's array</i>
<i>enumType</i>	<i>holds the type of the <code><other></code> element's <code><enumeration></code> array</i>

Returns

whether the other variable results are set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.262 `bool OSResult::setAnOtherVariableResultDense (int solldx, int otherldx, std::string name, std::string value, std::string description, std::string * s)`

Set the [i]th optimization solution's other (non-standard/solver specific)variable-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the new OtherVariableResult object
<i>name</i>	holds the name of the other element
<i>value</i>	holds the value of the other element
<i>s</i>	holds a pointer to the array of values of the var element

Returns

whether the other variable results are set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.263 `bool OSResult::setAnOtherVariableResultDense (int solldx, int otherldx, std::string name, std::string value, std::string description, std::string * s, std::string type, std::string varType, std::string enumType)`

Set the [i]th optimization solution's other (non-standard/solver specific)variable-related results, where i equals the given solution index.

This alternate signature sets the type of the value attribute and the *and* *<enumeration>* arrays

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the new OtherVariableResult object
<i>name</i>	holds the name of the other element
<i>value</i>	holds the value of the other element
<i>s</i>	holds a pointer to the array of values of the var element
<i>type</i>	holds the type of the <i><other></i> element's value attribute
<i>varType</i>	holds the type of the <i><other></i> element's array
<i>enumType</i>	holds the type of the <i><other></i> element's <i><enumeration></i> array

Returns

whether the other variable results are set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.264 `bool OSResult::setOtherVariableResultNumberOfVar (int solIdx, int otherIdx, int numberOfVar)`

Set the number of *children of another (non-standard/solver specific) variable-related result, for the [i]th solution.*

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>numberOfVar</i>	holds the number of <i>children</i>

Returns

whether the other variable result's name was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.265 `bool OSResult::setOtherVariableResultNumberOfEnumerations (int solIdx, int otherIdx, int numberOfEnumerations)`

Set the number of <enumeration> children of another (non-standard/solver specific) variable-related result, for the [i]th solution.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>numberOfEnumerations</i>	holds the number of <enumeration> children

Returns

whether the other variable result's name was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.266 `bool OSResult::setOtherVariableResultName (int solIdx, int otherIdx, std::string name)`

Set the name of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>name</i>	holds the name of the other element

Returns

whether the other variable result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.267 bool OSResult::setOtherVariableResultType (int solldx, int otherIdx, std::string type)

Set the type of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>type</i>	holds the type of the other element

Returns

whether the other variable result's type was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.268 bool OSResult::setOtherVariableResultVarType (int solldx, int otherIdx, std::string varType)

Set the varType of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>varType</i>	holds the data type of the <i>array of the <other> element</i>

Returns

whether the other variable result's varType was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.269 bool OSResult::setOtherVariableResultEnumType (int solIdx, int otherIdx, std::string enumType)

Set the enumType of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>enumType</i>	holds the data type of the <enumeration> array of the <other> element

Returns

whether the other variable result's enumType was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.270 bool OSResult::setOtherVariableResultValue (int solIdx, int otherIdx, std::string value)

Set the value of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherVariableResult object
<i>value</i>	holds the name of the other element

Returns

whether the other variable result's value was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.271 `bool OSResult::setOtherVariableResultDescription (int solldx, int otherldx, std::string description)`

Set the description of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherVariableResult object
<i>description</i>	holds the name of the other element

Returns

whether the other variable result's description was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.272 `bool OSResult::setOtherVariableResultVarldx (int solldx, int otherldx, int varldx, int idx)`

Set the index of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherVariableResult object
<i>varldx</i>	holds the index of the location to which the information is stored
<i>idx</i>	holds the index of the variable to which the information belongs

Returns

whether the other variable result's index was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.273 `bool OSResult::setOtherVariableResultVarName (int solldx, int otherldx, int varldx, std::string name)`

Set the name of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherVariableResult object
<i>varldx</i>	holds the index of the location to which the information is stored
<i>name</i>	holds the name of the variable to which the information belongs

Returns

whether the other variable result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.274 `bool OSResult::setOtherVariableResultVar (int solldx, int otherldx, int varldx, std::string value)`

Set the value of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherVariableResult object
<i>varldx</i>	holds the index of the location to which the information is stored
<i>value</i>	holds the value of the variable to which the information belongs

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.275 `bool OSResult::setOtherOptionEnumeration (int solldx, int otherldx, int object, int enumldx, std::string value, std::string description, int * i, int ni)`

Set the value and corresponding indices of another (non-standard/solver specific) variable-related result, for the [k]th solution, where k equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherVariableResult object
<i>object</i>	holds the object to which this enumeration pertains (legal values are taken from the ENUM_PROBLEM_COMPONENT enumeration — see OSGeneral.h)

<i>enumIdx</i>	holds the index of the OtherOptionEnumeration object
<i>value</i>	holds the value of this result
<i>description</i>	holds a description of this result
<i>i</i>	holds the indices of the variables that take on this value
<i>ni</i>	holds the dimension of the index vector i

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

6.200.3.276 bool OSResult::setNumberOfOtherObjectiveResults (int solIdx, int numberOfOtherObjectiveResults)

Set the [i]th optimization solution's other (non-standard/solver specific) objective-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first. This method then allocates the memory for the new [OtherObjectiveResult](#) objects

Parameters

<i>solIdx</i>	is the solution index
<i>numberOfOther-ObjectiveResult</i>	holds the number of OtherObjectiveResult objects Each other objective result contains the name (required), an optional description (std::string) and an optional value (std::string). Each other objective result can also optionally contain an array OtherObjResult for each objective. The Other-ObjResult contains an objective idx (required) and an optional std::string value.

Returns

whether the other objective results are set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.277 bool OSResult::setNumberOfObjValues (int solIdx, int numberOfObj)

Set the number of objectives to be given a value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the objective values.
<i>numberOfObj</i>	holds the number of objectives that are to be set

Returns

whether the information was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.278 bool OSResult::setNumberOfObjectiveValues (int *solldx*, int *n*)

Set the [i]th optimization solution's number of objective values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>n</i>	holds the number of elements in the array x

Returns

whether objective values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.279 bool OSResult::setObjectiveValuesSparse (int *solldx*, std::vector< [IndexValuePair](#) * > *x*)

Set the [i]th optimization solution's objective values, where i equals the given solution index.

Usually one of the objective is what the solution was solved for (or based on). Its index should be indicated in the solution's `objectiveIdx` attribute. Based on this objective's solution, the rest of the objective values are (optionally) calculated. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the objective values.
<i>x</i>	holds a vector of type IndexValuePair ; the <code>idx</code> component holds the index of the objective; the <code>value</code> component holds its value. The vector could be null if all objectives are 0. Possibly only the objective that the solution is based on has the value, and the rest of the objective values all get a <code>Double.NaN</code> value, meaning that they are not calculated.

Returns

whether objective values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.280 bool OSResult::setObjectiveValuesDense (int *solldx*, double * *objectiveValues*)

Set the [i]th optimization solution's objective values, where i equals the given solution index.

Usually one of the objective is what the solution was solved for (or based on). Its index should be indicated in the solution's `objectiveIdx` attribute. Based on this objective's solution, the rest of the objective values are (optionally) calculated. Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the objective values.
<i>objectiveValues</i>	holds the double sparse array of objective values to set. Possibly only the objective that the solution is based on has the value, and the rest of the objective values all get a <code>Double.NaN</code> value, meaning that they are not calculated.

Returns

whether objective values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.281 `bool OSResult::setObjValue (int solIdx, int number, int idx, std::string name, double val)`

Set an objective value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the objective values.
<i>number</i>	holds the location within the sparse array obj that is to be used
<i>idx</i>	holds the index of the objective that is to be set
<i>name</i>	holds the objective name (or an empty string).
<i>val</i>	holds the objective value to set.

Returns

whether primal variable value was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.282 `bool OSResult::setOtherObjectiveResultNumberOfObj (int solIdx, int otherIdx, int numberOfObj)`

Set the number of <obj> children of another (non-standard/solver specific) objective-related result, for the [i]th solution.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>numberOfObj</i>	holds the number of <obj> children

Returns

whether the other objective result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.283 `bool OSResult::setOtherObjectiveResultNumberOfEnumerations (int solIdx, int otherIdx, int numberOfObj)`

Set the number of <enumeration> children of another (non-standard/solver specific) objective-related result, for the [i]th solution.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>numberOfObj</i>	holds the number of <obj> children

Returns

whether the other objective result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.284 `bool OSResult::setOtherObjectiveResultName (int solIdx, int otherIdx, std::string name)`

Set the name of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>name</i>	holds the name of the other element

Returns

whether the other objective result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.285 `bool OSResult::setOtherObjectiveResultType (int solldx, int otherldx, std::string type)`

Set the type of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>name</i>	holds the type of the <other> element

Returns

whether the other objective result's type was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.286 `bool OSResult::setOtherObjectiveResultObjType (int solldx, int otherldx, std::string objType)`

Set the objType of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>name</i>	holds the data type of the <other> element's array

Returns

whether the other objective result's objType was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.287 `bool OSResult::setOtherObjectiveResultEnumType (int solldx, int otherldx, std::string enumType)`

Set the enumType of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>name</i>	holds the data type of the <other> element's <enumeration> array

Returns

whether the other objective result's enumType was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.288 bool OSResult::setOtherObjectiveResultValue (int *solldx*, int *otherldx*, std::string *value*)

Set the value of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>value</i>	holds the name of the other element

Returns

whether the other objective result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.289 bool OSResult::setOtherObjectiveResultDescription (int *solldx*, int *otherldx*, std::string *description*)

Set the description of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>description</i>	holds the name of the other element

Returns

whether the other objective result's description was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.290 bool OSResult::setOtherObjectiveResultObjIdx (int solIdx, int otherIdx, int objIdx, int idx)

Set the index of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>objIdx</i>	holds the index of the location to which the information is stored
<i>idx</i>	holds the index of the objective to which the information belongs

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.291 bool OSResult::setOtherObjectiveResultObjName (int solIdx, int otherIdx, int objIdx, std::string name)

Set the name of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>objIdx</i>	holds the index of the location to which the information is stored
<i>name</i>	holds the name of the objective to which the information belongs

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.292 bool OSResult::setOtherObjectiveResultObj (int *solldx*, int *otherldx*, int *objldx*, std::string *value*)

Set the value of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherObjectiveResult object
<i>objldx</i>	holds the index of the location to which the information is stored
<i>value</i>	holds the value of the objective to which the information belongs

Returns

whether the other objective result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

6.200.3.293 bool OSResult::setNumberOfOtherConstraintResults (int *solldx*, int *numberOfOtherConstraintResults*)

Set the [i]th optimization solution's other (non-standard/solver specific) constraint-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first. This method then allocates the memory for the new [OtherConstraintResult](#) objects

Parameters

<i>solldx</i>	is the solution index
<i>numberOfOtherConstraintResults</i>	holds the number of OtherConstraintResult objects Each other objective result contains the name (required), an optional description (std::string) and an optional value (std::string). Each other constraint result can also optionally contain an array OtherConResult for each constraint. The OtherConResult contains a constraint idx (required) and an optional std::string value.

Returns

whether the other objective results are set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.294 `bool OSResult::setNumberOfDualValues (int solIdx, int numberOfCon)`

Set the number of constraints to be given a value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the constraint values.
<i>numberOfCon</i>	holds the number of constraint that are to be set

Returns

whether the information was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.295 `bool OSResult::setNumberOfDualVariableValues (int solIdx, int n)`

Set the [i]th optimization solution's number of dual variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the dual variable values.
<i>n</i>	holds the number of elements in the array x

Returns

whether dual variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.296 `bool OSResult::setDualVariableValuesSparse (int solIdx, std::vector< IndexValuePair * > x)`

Set the [i]th optimization solution's dual variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index to set the dual variable values.
<i>x</i>	holds a vector of type IndexValuePair ; the idx component holds the index of the constraint; the value component holds its value. The vector could be null if all dual variables are 0.

Returns

whether dual variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.297 `bool OSResult::setDualVariableValuesDense (int solldx, double * y)`

Set the [i]th optimization solution's dual variable values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the dual variable values.
<i>y</i>	holds a double dense array of variable dual values; it could be NULL if all values are 0.

Returns

whether dual variable values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.298 `bool OSResult::setConstraintValuesDense (int solldx, double * constraintValues)`

Set the [i]th optimization solution's constraint values, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>constraintValues</i>	holds the a double dense array of constraint values to set; it could be null if all constraint values are 0.

Returns

whether constraint values are set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.299 `bool OSResult::setDualValue (int solldx, int number, int idx, std::string name, double val)`

Set a dual value.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>number</i>	holds the location within the sparse array con that is to be used
<i>idx</i>	holds the index of the constraint that is to be set
<i>name</i>	holds the constraint name (or an empty string).
<i>val</i>	holds the constraint value to set.

Returns

whether dual variable value was set successfully or not.

See Also

[setSolutionNumber\(int\)](#)

6.200.3.300 bool OSResult::setOtherConstraintResultNumberOfCon (int *solIdx*, int *otherIdx*, int *numberOfCon*)

Set the number of <con> children of another (non-standard/solver specific) constraint-related result, for the [i]th solution.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>numberOfCon</i>	holds the number of <con> children

Returns

whether the other constraint result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.301 bool OSResult::setOtherConstraintResultNumberOfEnumerations (int *solIdx*, int *otherIdx*, int *numberOfCon*)

Set the number of <enumeration> children of another (non-standard/solver specific) constraint-related result, for the [i]th solution.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>numberOfCon</i>	holds the number of <con> children

Returns

whether the other constraint result's name was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.302 `bool OSResult::setOtherConstraintResultName (int solldx, int otherldx, std::string name)`

Set the name of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherConstraintResult object
<i>name</i>	holds the name of the other element

Returns

whether the other constraint result's name was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherConResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.303 `bool OSResult::setOtherConstraintResultType (int solldx, int otherldx, std::string type)`

Set the type of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherConstraintResult object
<i>name</i>	holds the type of the <other> element

Returns

whether the other constraint result's type was set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.OtherConResult](#)
[setSolutionNumber\(int\)](#)

6.200.3.304 `bool OSResult::setOtherConstraintResultConType (int solldx, int otherldx, std::string conType)`

Set the conType of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>name</i>	holds the type of the <other> element's <con> array

Returns

whether the other constraint result's conType was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.305 bool OSResult::setOtherConstraintResultEnumType (int *solldx*, int *otherIdx*, std::string *enumType*)

Set the enumType of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>name</i>	holds the type of the <other> element's <enumeration> array

Returns

whether the other constraint result's enumType was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.306 bool OSResult::setOtherConstraintResultValue (int *solldx*, int *otherIdx*, std::string *value*)

Set the value of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>value</i>	holds the name of the other element

Returns

whether the other constraint result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.307 bool OSResult::setOtherConstraintResultDescription (int *solIdx*, int *otherIdx*, std::string *description*)

Set the description of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>description</i>	holds the name of the other element

Returns

whether the other constraint result's description was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.308 bool OSResult::setOtherConstraintResultConIdx (int *solIdx*, int *otherIdx*, int *conIdx*, int *idx*)

Set the index of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solIdx</i>	holds the solution index
<i>otherIdx</i>	holds the index of the OtherConstraintResult object
<i>conIdx</i>	holds the index of the location to which the information is stored
<i>idx</i>	holds the index of the onstraint to which the information belongs

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.309 bool OSResult::setOtherConstraintResultConName (int *solldx*, int *otherldx*, int *conldx*, std::string *name*)

Set the name of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherConstraintResult object
<i>conldx</i>	holds the index of the location to which the information is stored
<i>name</i>	holds the name of the constraint to which the information belongs

Returns

whether the other variable result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.310 bool OSResult::setOtherConstraintResultCon (int *solldx*, int *otherldx*, int *conldx*, std::string *value*)

Set the value of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index
<i>otherldx</i>	holds the index of the OtherConstraintResult object
<i>conldx</i>	holds the index of the location to which the information is stored
<i>value</i>	holds the value of the constraint to which the information belongs

Returns

whether the other constraint result's value was set successfully or not.

See Also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

6.200.3.311 `bool OSResult::setNumberOfOtherSolutionResults (int solldx, int numberOfOtherSolutionResults)`

Set the [i]th optimization solution's other (non-standard/solver specific) solution-related results, where i equals the given solution index.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first. This method then allocates the memory for the new [OtherSolutionResult](#) objects

Parameters

<i>solldx</i>	is the solution index
<i>numberOfOtherSolutionResults</i>	holds the number of OtherSolutionResult objects Each other objective result contains the name (required), an optional description (std::string) and an optional category (std::string). Each other solution result can also optionally contain an array Item for each result. The Item content is string-valued.

Returns

whether the other objective results are set successfully or not.

See Also

[org.optimizationservices.oscommon.datastructure.osresult.OtherSolutionResult](#)
[org.optimizationservices.oscommon.datastructure.osresult.Item](#)
[setSolutionNumber\(int\)](#)

6.200.3.312 `bool OSResult::setOtherSolutionResultName (int solldx, int otherldx, std::string name)`

Set the name associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherldx</i>	holds the index of the otherSolutionResult
<i>name</i>	holds the name of the otherSolutionResult

Returns

whether the other solution result was set successfully or not.

6.200.3.313 `bool OSResult::setOtherSolutionResultValue (int solldx, int otherldx, std::string value)`

Set the value associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherldx</i>	holds the index of the otherSolutionResult
<i>value</i>	holds the value of the otherSolutionResult

Returns

whether the other solution result was set successfully or not.

6.200.3.314 `bool OSResult::setOtherSolutionResultCategory (int solldx, int otherldx, std::string category)`

Set the category associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherldx</i>	holds the index of the otherSolutionResult
<i>category</i>	holds the category of the otherSolutionResult

Returns

whether the other solution result was set successfully or not.

6.200.3.315 `bool OSResult::setOtherSolutionResultDescription (int solldx, int otherldx, std::string description)`

Set the description associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherldx</i>	holds the index of the otherSolutionResult
<i>category</i>	holds the description of the otherSolutionResult

Returns

whether the other solution result was set successfully or not.

6.200.3.316 `bool OSResult::setOtherSolutionResultNumberOfItems (int solldx, int otherldx, int numberOfItems)`

Set the number of items associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherldx</i>	holds the index of the otherSolutionResult
<i>numberOfItems</i>	holds the number of items

Returns

whether the other solution result was set successfully or not.

6.200.3.317 `bool OSResult::setOtherSolutionResultItem (int solldx, int otherldx, int itemldx, std::string item)`

Set one item associated with the [j]th other solution result of solution [i].

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the constraint values.
<i>otherIdx</i>	holds the index of the otherSolutionResult
<i>itemIdx</i>	holds the index of the item
<i>item</i>	holds the value of the item

Returns

whether the other solution result item was set successfully or not.

6.200.3.318 `bool OSResult::setAnotherSolutionResult (int solldx, std::string name, std::string value, std::string category, std::string description, int numberOfItems, std::string * item)`

Set another solution result of solution [i].

Parameters

<i>solldx</i>	holds the solution index i.
<i>name</i>	holds the name of the other solution result
<i>value</i>	holds the value of the other solution result
<i>category</i>	holds the category of the result
<i>description</i>	holds a description of the result
<i>numberOfItems</i>	holds the number of items
<i>item</i>	holds a pointer to the array of items (can be NULL if numberOfItems is 0)

Returns

whether the other solution result was set successfully or not.

6.200.3.319 `bool OSResult::setNumberOfSolverOutputs (int numberOfSolverOutputs)`

Set the number of other solver outputs.

Parameters

<i>numberOfOther-SolverOutputs</i>	holds the number of SolverOutput objects Each solver output can also optionally contain an array Item for each result. The Item content is string-valued.
------------------------------------	---

Returns

whether the results were set successfully or not.

6.200.3.320 `bool OSResult::setSolverOutputName (int otherIdx, std::string name)`

Set the name associated with the [j]th solver output.

Parameters

<i>otherIdx</i>	holds the index of the solverOutput object
<i>name</i>	holds the name of the solver output

Returns

whether the solver output was set successfully or not.

6.200.3.321 `bool OSResult::setSolverOutputCategory (int otherIdx, std::string category)`

Set the category associated with the [j]th solver output.

Parameters

<i>otherIdx</i>	holds the index of the solverOutput object
<i>name</i>	holds the category of the solver output

Returns

whether the solver output was set successfully or not.

6.200.3.322 `bool OSResult::setSolverOutputDescription (int otherIdx, std::string description)`

Set the description associated with the [j]th solver output.

Parameters

<i>otherIdx</i>	holds the index of the solverOutput object
<i>name</i>	holds the description of the solver output

Returns

whether the solver output was set successfully or not.

6.200.3.323 `bool OSResult::setSolverOutputNumberOfItems (int otherIdx, int numberOfItems)`

Set the number of items associated with the [j]th solver output.

Before this method is called, the [setSolutionNumber\(int\)](#) method has to be called first.

Parameters

<i>otherIdx</i>	holds the index of the solverOutput object
<i>numberOfItems</i>	holds the number of items

Returns

whether the information was set successfully or not.

6.200.3.324 `bool OSResult::setSolverOutputItem (int otherIdx, int itemIdx, std::string item)`

Set one item associated with the [j]th solver output.

Parameters

<i>otherIdx</i>	holds the index of the otherSolutionResult
<i>itemIdx</i>	holds the index of the item
<i>item</i>	holds the value of the item

Returns

whether the information was set successfully or not.

6.200.4 Member Data Documentation

6.200.4.1 GeneralFileHeader* OSResult::resultHeader

header information

Definition at line 2320 of file OSResult.h.

6.200.4.2 GeneralResult* OSResult::general

general holds the first child of the [OSResult](#) specified by the OSrL Schema.

Definition at line 2325 of file OSResult.h.

6.200.4.3 SystemResult* OSResult::system

system holds the second child of the [OSResult](#) specified by the OSrL Schema.

Definition at line 2330 of file OSResult.h.

6.200.4.4 ServiceResult* OSResult::service

service holds the third child of the [OSResult](#) specified by the OSrL Schema.

Definition at line 2335 of file OSResult.h.

6.200.4.5 JobResult* OSResult::job

job holds the fourth child of the [OSResult](#) specified by the OSrL Schema.

Definition at line 2340 of file OSResult.h.

6.200.4.6 OptimizationResult* OSResult::optimization

optimization holds the fifth child of the [OSResult](#) specified by the OSrL Schema.

Definition at line 2345 of file OSResult.h.

6.200.4.7 int OSResult::m_iVariableNumber

m_iVariableNumber holds the variable number.

Definition at line 2378 of file OSResult.h.

6.200.4.8 int OSResult::m_iObjectiveNumber

m_iObjectiveNumber holds the objective number.

Definition at line 2383 of file OSResult.h.

6.200.4.9 int OSResult::m_iConstraintNumber

m_iConstraintNumber holds the constraint number.

Definition at line 2388 of file OSResult.h.

6.200.4.10 int OSResult::m_iNumberOfOtherVariableResults

m_iNumberOfOtherVariableResults holds the number of [OtherVariableResult](#) objects.

Definition at line 2393 of file OSResult.h.

6.200.4.11 double* OSResult::m_mdPrimalValues

m_mdPrimalValues a vector of primal variables.

Definition at line 2398 of file OSResult.h.

6.200.4.12 double* OSResult::m_mdDualValues

m_mdDualValues a vector of dual variables.

Definition at line 2403 of file OSResult.h.

6.200.4.13 std::vector<IndexValuePair*> OSResult::primalVals

Definition at line 2406 of file OSResult.h.

6.200.4.14 std::vector<IndexValuePair*> OSResult::dualVals

Definition at line 2408 of file OSResult.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSResult.h](#)

6.201 OSrL2Gams Class Reference

Reads an optimization result and stores result and solution in a Gams Modeling Object.

```
#include <OSosrl2gams.hpp>
```

Public Member Functions

- [OSrL2Gams](#) (struct gmoRec *gmo_)
Constructor.
- [~OSrL2Gams](#) ()
Destructor.
- void [writeSolution](#) (OSResult &osresult)
Writes a solution into a GMO with the result given as OSResult object.
- void [writeSolution](#) (std::string &osrl)
Writes a solution into a GMO with the result given as osrl string.

6.201.1 Detailed Description

Reads an optimization result and stores result and solution in a Gams Modeling Object.

Definition at line 20 of file OSosrl2gams.hpp.

6.201.2 Constructor & Destructor Documentation

6.201.2.1 OSrL2Gams::OSrL2Gams (struct gmoRec * *gmo_*)

Constructor.

Parameters

<i>gmo_</i>	GMO handler.
-------------	--------------

6.201.2.2 OSrL2Gams::~~OSrL2Gams () [inline]

Destructor.

Definition at line 34 of file OSosrl2gams.hpp.

6.201.3 Member Function Documentation

6.201.3.1 void OSrL2Gams::writeSolution (OSResult & *osresult*)

Writes a solution into a GMO with the result given as [OSResult](#) object.

Parameters

<i>osresult</i>	Optimization result as object.
-----------------	--------------------------------

6.201.3.2 void OSrL2Gams::writeSolution (std::string & *osrl*)

Writes a solution into a GMO with the result given as *osrl* string.

Parameters

<i>osrl</i>	Optimization result as string.
-------------	--------------------------------

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

- [/home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosrl2gams.hpp](#)

6.202 OSrLParserData Class Reference

The [OSrLParserData](#) Class.

```
#include <OSrLParserData.h>
```


- total number of constraints in the model instance*
- int [numberOfObjectives](#)
total number of [Objectives](#) in the model instance
- int [numberOfIdx](#)
number of indexes in a category of basis elements, may change from category to category and solution to solution
- int [numberOfVar](#)
number of variables in a solution instance, may change from solution to solution
- int [numberOfVarIdx](#)
number of variables indices in other variable result enumeration, may change from solution to solution
- int [numberOfCon](#)
number of constraints in a solution instance, may change from solution to solution
- int [numberOfEnumerations](#)
- int [numberOfObj](#)
number of [Objectives](#) in a solution instance may change from solution to solution
- int [numberOf](#)
a temporary variable to hold the number of entries in a list
- int [kounter](#)
a temporary counter to count variables, number of attributes, etc.
- int [iOther](#)
a temporary counter to count other variable, objective and constraint results
- int [ivar](#)
a temporary counter to count second-level objects
- int [idx](#)
a temporary variable to hold an integer index value
- double [tempVal](#)
a temporary variable to hold an integer or double value
- int [templnt](#)
a temporary variable to hold an integer value
- std::string [tempStr](#)
a temporary variable to hold a string
- std::string [name](#)
a temporary variable to hold a variable, objective or constraint name
- std::ostream [outStr](#)
a temporary variable to hold an output stream value
- int [numberOfOtherVariableResults](#)
the number of types of variable results other than the value of the variable
- int [numberOfOtherObjectiveResults](#)
the number of types of objective results other than the value of the objective
- int [numberOfOtherConstraintResults](#)
the number of types of constraint results other than the value of the constraint
- unsigned int [solutionIdx](#)
an index of which solution we have found
- int [mult](#)
a multiplier or repeat count for compact representation of an array
- int [incr](#)
an increment for compact representation of an array (used with mult)
- bool [numberAttributePresent](#)

a number of boolean vriables to track which of the attributes have been found in the present list.

- bool [incrPresent](#)
- bool [multPresent](#)
- bool [idxAttributePresent](#)
- bool [categoryAttributePresent](#)
- bool [descriptionAttributePresent](#)
- bool [nameAttributePresent](#)
- bool [numberOfVarAttributePresent](#)
- bool [numberOfVarIdxAttributePresent](#)
- bool [numberOfObjAttributePresent](#)
- bool [numberOfObjIdxAttributePresent](#)
- bool [numberOfConAttributePresent](#)
- bool [numberOfConIdxAttributePresent](#)
- bool [numberOfEnumerationsAttributePresent](#)
- bool [typeAttributePresent](#)
- bool [varTypeAttributePresent](#)
- bool [objTypeAttributePresent](#)
- bool [conTypeAttributePresent](#)
- bool [enumTypeAttributePresent](#)
- bool [unitAttributePresent](#)
- bool [valueAttributePresent](#)
- bool [weightedObjAttributePresent](#)
- std::string [categoryAttribute](#)

many attributes, particularly those that return strings, are used multiple times, and the parser uses generic constructs for them.

- std::string [descriptionAttribute](#)
- std::string [nameAttribute](#)
- std::string [typeAttribute](#)
- std::string [varTypeAttribute](#)
- std::string [objTypeAttribute](#)
- std::string [conTypeAttribute](#)
- std::string [enumTypeAttribute](#)
- std::string [unitAttribute](#)
- std::string [valueAttribute](#)
- bool [nVarPresent](#)
- bool [nObjPresent](#)
- bool [nConPresent](#)
- bool [generalStatusPresent](#)

set general...Present to true if the corresponding element (child of the <general> element) has been parsed

- bool [generalMessagePresent](#)
- bool [generalServiceURIPresent](#)
- bool [generalServiceNamePresent](#)
- bool [generalInstanceNamePresent](#)
- bool [generalJobIDPresent](#)
- bool [generalSolverInvokedPresent](#)
- bool [generalTimeStampPresent](#)
- bool [otherGeneralResultsPresent](#)
- bool [systemInformationPresent](#)

set system...Present to true if the corresponding element (child of the <system> element) has been parsed

- bool [systemAvailableDiskSpacePresent](#)

- bool [systemAvailableMemoryPresent](#)
- bool [systemAvailableCPUSpeedPresent](#)
- bool [systemAvailableCPUNumberPresent](#)
- bool [otherSystemResultsPresent](#)
- bool [serviceCurrentStatePresent](#)
 - set service...Present to true if the corresponding element (child of the <service> element) has been parsed*
- bool [serviceCurrentJobCountPresent](#)
- bool [serviceTotalJobsSoFarPresent](#)
- bool [timeServiceStartedPresent](#)
- bool [serviceUtilizationPresent](#)
- bool [otherServiceResultsPresent](#)
- bool [jobStatusPresent](#)
 - set job...Present to true if the corresponding element (child of the <job> element) has been parsed*
- bool [jobSubmitTimePresent](#)
- bool [scheduledStartTimePresent](#)
- bool [actualStartTimePresent](#)
- bool [jobEndTimePresent](#)
- bool [jobTimingInformationPresent](#)
- bool [jobUsedDiskSpacePresent](#)
- bool [jobUsedMemoryPresent](#)
- bool [jobUsedCPUSpeedPresent](#)
- bool [jobUsedCPUNumberPresent](#)
- bool [otherJobResultsPresent](#)
- bool [numberOfItemsPresent](#)
- int [numberOfItems](#)
- struct [IndexValuePair](#) * [primalValPair](#)
 - for each solution we will build a vector of index-value pairs of primal values*
- std::vector< [IndexValuePair](#) * > [primalVals](#)
- struct [IndexValuePair](#) * [objValPair](#)
 - for each solution we will build a vector of index-value pairs of objective function values*
- std::vector< [IndexValuePair](#) * > [objVals](#)
- struct [IndexValuePair](#) * [dualValPair](#)
 - for each solution we will build a vector of index-value pairs of dual values*
- std::vector< [IndexValuePair](#) * > [dualVals](#)
- struct [OtherVariableResultStruct](#) * [otherVarStruct](#)
 - a pointer to an [OtherVariableResultStruct](#) structure*
- std::vector< [OtherVariableResultStruct](#) * > [otherVarVec](#)
 - store a vector of pointers to otherVarVec structures*
- char * [errorText](#)
 - if the parser finds invalid text it is held here and we delete if the file was not valid*
- std::string [parser_errors](#)
 - used to accumulate error message so the parser does not die on the first error encountered*
- bool [ignoreDataAfterErrors](#)
 - two booleans to govern the behavior after an error has been encountered*
- bool [suppressFurtherErrorMessage](#)

6.202.1 Detailed Description

The [OSrLParserData](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

the [OSrLParserData](#) class is used to temporarily hold data found in parsing the OSrL instance we do this so we can have a reentrant parser.

Definition at line 83 of file OSrLParserData.h.

6.202.2 Constructor & Destructor Documentation

6.202.2.1 OSrLParserData::OSrLParserData ()

the [OSrLParserData](#) class constructor

6.202.2.2 OSrLParserData::~OSrLParserData ()

6.202.3 Member Data Documentation

6.202.3.1 std::string OSrLParserData::statusType

the status type of the result

Definition at line 95 of file OSrLParserData.h.

6.202.3.2 std::string OSrLParserData::statusDescription

the status Description of the solution

Definition at line 98 of file OSrLParserData.h.

6.202.3.3 double OSrLParserData::timeValue

the next few variables store a time measurement and associated attribute values

Definition at line 102 of file OSrLParserData.h.

6.202.3.4 std::string OSrLParserData::timeType

Definition at line 103 of file OSrLParserData.h.

6.202.3.5 std::string OSrLParserData::timeCategory

Definition at line 104 of file OSrLParserData.h.

6.202.3.6 std::string OSrLParserData::timeUnit

Definition at line 105 of file OSrLParserData.h.

6.202.3.7 std::string OSrLParserData::timeDescription

Definition at line 106 of file OSrLParserData.h.

6.202.3.8 int OSrLParserData::numberOfTimes

There could be more than one time measurement; numberOfTimes stores the number of them.

Definition at line 110 of file OSrLParserData.h.

6.202.3.9 std::string OSrLParserData::tmpOtherValue

Provide temporary storage for attribute values associated with an [OtherVarResult](#).

Definition at line 113 of file OSrLParserData.h.

6.202.3.10 std::string OSrLParserData::tmpOtherName

Definition at line 114 of file OSrLParserData.h.

6.202.3.11 std::string OSrLParserData::tmpOtherDescription

Definition at line 115 of file OSrLParserData.h.

6.202.3.12 std::string OSrLParserData::itemContent

Provide temporary storage for a single <record> contained in an [OtherSolutionResult](#).

Definition at line 118 of file OSrLParserData.h.

6.202.3.13 void* OSrLParserData::scanner

scanner is used to store data in a reentrant lexer we use this to pass an [OSrLParserData](#) object to the parser

Definition at line 122 of file OSrLParserData.h.

6.202.3.14 unsigned int OSrLParserData::numberOfSolutions

number of result solutions

Definition at line 125 of file OSrLParserData.h.

6.202.3.15 int OSrLParserData::numberOfVariables

total number of variables in the model instance

Definition at line 128 of file OSrLParserData.h.

6.202.3.16 int OSrLParserData::numberOfConstraints

total number of constraints in the model instance

Definition at line 131 of file OSrLParserData.h.

6.202.3.17 int OSrLParserData::numberOfObjectives

total number of [Objectives](#) in the model instance

Definition at line 134 of file OSrLParserData.h.

6.202.3.18 int OSrLParserData::numberOfIdx

number of indexes in a category of basis elements, may change from category to category and solution to solution

Definition at line 139 of file OSrLParserData.h.

6.202.3.19 int OSrLParserData::numberOfVar

number of variables in a solution instance, may change from solution to solution

Definition at line 144 of file OSrLParserData.h.

6.202.3.20 int OSrLParserData::numberOfVarIdx

number of variables indices in other variable result enumeration, may change from solution to solution

Definition at line 149 of file OSrLParserData.h.

6.202.3.21 int OSrLParserData::numberOfCon

number of constraints in a solution instance, may change from solution to solution

Definition at line 154 of file OSrLParserData.h.

6.202.3.22 int OSrLParserData::numberOfEnumerations

Definition at line 156 of file OSrLParserData.h.

6.202.3.23 int OSrLParserData::numberOfObj

number of [Objectives](#) in a solution instance may change from solution to solution

Definition at line 161 of file OSrLParserData.h.

6.202.3.24 int OSrLParserData::numberOf

a temporary variable to hold the number of entries in a list

Definition at line 164 of file OSrLParserData.h.

6.202.3.25 int OSrLParserData::kounter

a temporary counter to count variables, number of attributes, etc.

Definition at line 167 of file OSrLParserData.h.

6.202.3.26 int OSrLParserData::iOther

a temporary counter to count other variable, objective and constraint results

Definition at line 170 of file OSrLParserData.h.

6.202.3.27 int OSrLParserData::ivar

a temporary counter to count second-level objects

Definition at line 173 of file OSrLParserData.h.

6.202.3.28 int OSrLParserData::idx

a temporary variable to hold an integer index value

Definition at line 176 of file OSrLParserData.h.

6.202.3.29 double OSrLParserData::tempVal

a temporary variable to hold an integer or double value

Definition at line 179 of file OSrLParserData.h.

6.202.3.30 int OSrLParserData::templnt

a temporary variable to hold an integer value

Definition at line 182 of file OSrLParserData.h.

6.202.3.31 std::string OSrLParserData::tempStr

a temporary variable to hold a string

Definition at line 185 of file OSrLParserData.h.

6.202.3.32 std::string OSrLParserData::name

a temporary variable to hold a variable, objective or constraint name

Definition at line 188 of file OSrLParserData.h.

6.202.3.33 std::ostream OSrLParserData::outStr

a temporary variable to hold an output stream value

Definition at line 191 of file OSrLParserData.h.

6.202.3.34 int OSrLParserData::numberOfOtherVariableResults

the number of types of variable results other than the value of the variable

Definition at line 196 of file OSrLParserData.h.

6.202.3.35 int OSrLParserData::numberOfOtherObjectiveResults

the number of types of objective results other than the value of the objective

Definition at line 201 of file OSrLParserData.h.

6.202.3.36 int OSrLParserData::numberOfOtherConstraintResults

the number of types of constraint results other than the value of the constraint

Definition at line 206 of file OSrLParserData.h.

6.202.3.37 unsigned int OSrLParserData::solutionIdx

an index of which solution we have found

Definition at line 209 of file OSrLParserData.h.

6.202.3.38 int OSrLParserData::mult

a multiplier or repeat count for compact representation of an array

Definition at line 212 of file OSrLParserData.h.

6.202.3.39 int OSrLParserData::incr

an increment for compact representation of an array (used with mult)

Definition at line 215 of file OSrLParserData.h.

6.202.3.40 bool OSrLParserData::numberAttributePresent

a number of boolean variables to track which of the attributes have been found in the present list.

Attributes have standardized names, and the information about their presence or absence is immaterial once the list has been completely processed, so the boolean variables can be reused in the same way the names can be reused.

Definition at line 223 of file OSrLParserData.h.

6.202.3.41 bool OSrLParserData::incrPresent

Definition at line 224 of file OSrLParserData.h.

6.202.3.42 bool OSrLParserData::multPresent

Definition at line 225 of file OSrLParserData.h.

6.202.3.43 bool OSrLParserData::idxAttributePresent

Definition at line 226 of file OSrLParserData.h.

6.202.3.44 bool OSrLParserData::categoryAttributePresent

Definition at line 227 of file OSrLParserData.h.

6.202.3.45 bool OSrLParserData::descriptionAttributePresent

Definition at line 228 of file OSrLParserData.h.

6.202.3.46 bool OSrLParserData::nameAttributePresent

Definition at line 229 of file OSrLParserData.h.

6.202.3.47 bool OSrLParserData::numberOfVarAttributePresent

Definition at line 230 of file OSrLParserData.h.

6.202.3.48 bool OSrLParserData::numberOfVarIdxAttributePresent

Definition at line 231 of file OSrLParserData.h.

6.202.3.49 bool OSrLParserData::numberOfObjAttributePresent

Definition at line 232 of file OSrLParserData.h.

6.202.3.50 bool OSrLParserData::numberOfObjIdxAttributePresent

Definition at line 233 of file OSrLParserData.h.

6.202.3.51 bool OSrLParserData::numberOfConAttributePresent

Definition at line 234 of file OSrLParserData.h.

6.202.3.52 bool OSrLParserData::numberOfConIdxAttributePresent

Definition at line 235 of file OSrLParserData.h.

6.202.3.53 bool OSrLParserData::numberOfEnumerationsAttributePresent

Definition at line 236 of file OSrLParserData.h.

6.202.3.54 bool OSrLParserData::typeAttributePresent

Definition at line 237 of file OSrLParserData.h.

6.202.3.55 bool OSrLParserData::varTypeAttributePresent

Definition at line 238 of file OSrLParserData.h.

6.202.3.56 bool OSrLParserData::objTypeAttributePresent

Definition at line 239 of file OSrLParserData.h.

6.202.3.57 bool OSrLParserData::conTypeAttributePresent

Definition at line 240 of file OSrLParserData.h.

6.202.3.58 bool OSrLParserData::enumTypeAttributePresent

Definition at line 241 of file OSrLParserData.h.

6.202.3.59 bool OSrLParserData::unitAttributePresent

Definition at line 242 of file OSrLParserData.h.

6.202.3.60 bool OSrLParserData::valueAttributePresent

Definition at line 243 of file OSrLParserData.h.

6.202.3.61 bool OSrLParserData::weightedObjAttributePresent

Definition at line 244 of file OSrLParserData.h.

6.202.3.62 std::string OSrLParserData::categoryAttribute

many attributes, particularly those that return strings, are used multiple times, and the parser uses generic constructs for them.

These temporary variables are used to hold the values returned by the parser.

Definition at line 251 of file OSrLParserData.h.

6.202.3.63 std::string OSrLParserData::descriptionAttribute

Definition at line 252 of file OSrLParserData.h.

6.202.3.64 std::string OSrLParserData::nameAttribute

Definition at line 253 of file OSrLParserData.h.

6.202.3.65 std::string OSrLParserData::typeAttribute

Definition at line 254 of file OSrLParserData.h.

6.202.3.66 std::string OSrLParserData::varTypeAttribute

Definition at line 255 of file OSrLParserData.h.

6.202.3.67 std::string OSrLParserData::objTypeAttribute

Definition at line 256 of file OSrLParserData.h.

6.202.3.68 std::string OSrLParserData::conTypeAttribute

Definition at line 257 of file OSrLParserData.h.

6.202.3.69 std::string OSrLParserData::enumTypeAttribute

Definition at line 258 of file OSrLParserData.h.

6.202.3.70 std::string OSrLParserData::unitAttribute

Definition at line 259 of file OSrLParserData.h.

6.202.3.71 std::string OSrLParserData::valueAttribute

Definition at line 260 of file OSrLParserData.h.

6.202.3.72 bool OSrLParserData::nVarPresent

Definition at line 263 of file OSrLParserData.h.

6.202.3.73 bool OSrLParserData::nObjPresent

Definition at line 264 of file OSrLParserData.h.

6.202.3.74 bool OSrLParserData::nConPresent

Definition at line 265 of file OSrLParserData.h.

6.202.3.75 bool OSrLParserData::generalStatusPresent

set general...Present to true if the corresponding element (child of the <general> element) has been parsed

Definition at line 270 of file OSrLParserData.h.

6.202.3.76 bool OSrLParserData::generalMessagePresent

Definition at line 271 of file OSrLParserData.h.

6.202.3.77 bool OSrLParserData::generalServiceURIPresent

Definition at line 272 of file OSrLParserData.h.

6.202.3.78 bool OSrLParserData::generalServiceNamePresent

Definition at line 273 of file OSrLParserData.h.

6.202.3.79 bool OSrLParserData::generalInstanceNamePresent

Definition at line 274 of file OSrLParserData.h.

6.202.3.80 bool OSrLParserData::generalJobIDPresent

Definition at line 275 of file OSrLParserData.h.

6.202.3.81 bool OSrLParserData::generalSolverInvokedPresent

Definition at line 276 of file OSrLParserData.h.

6.202.3.82 bool OSrLParserData::generalTimeStampPresent

Definition at line 277 of file OSrLParserData.h.

6.202.3.83 bool OSrLParserData::otherGeneralResultsPresent

Definition at line 278 of file OSrLParserData.h.

6.202.3.84 bool OSrLParserData::systemInformationPresent

set system...Present to true if the corresponding element (child of the <system> element) has been parsed

Definition at line 283 of file OSrLParserData.h.

6.202.3.85 bool OSrLParserData::systemAvailableDiskSpacePresent

Definition at line 284 of file OSrLParserData.h.

6.202.3.86 bool OSrLParserData::systemAvailableMemoryPresent

Definition at line 285 of file OSrLParserData.h.

6.202.3.87 bool OSrLParserData::systemAvailableCPUSpeedPresent

Definition at line 286 of file OSrLParserData.h.

6.202.3.88 bool OSrLParserData::systemAvailableCPUNumberPresent

Definition at line 287 of file OSrLParserData.h.

6.202.3.89 bool OSrLParserData::otherSystemResultsPresent

Definition at line 288 of file OSrLParserData.h.

6.202.3.90 bool OSrLParserData::serviceCurrentStatePresent

set service...Present to true if the corresponding element (child of the <service> element) has been parsed

Definition at line 293 of file OSrLParserData.h.

6.202.3.91 bool OSrLParserData::serviceCurrentJobCountPresent

Definition at line 294 of file OSrLParserData.h.

6.202.3.92 bool OSrLParserData::serviceTotalJobsSoFarPresent

Definition at line 295 of file OSrLParserData.h.

6.202.3.93 bool OSrLParserData::timeServiceStartedPresent

Definition at line 296 of file OSrLParserData.h.

6.202.3.94 bool OSrLParserData::serviceUtilizationPresent

Definition at line 297 of file OSrLParserData.h.

6.202.3.95 bool OSrLParserData::otherServiceResultsPresent

Definition at line 298 of file OSrLParserData.h.

6.202.3.96 bool OSrLParserData::jobStatusPresent

set job...Present to true if the corresponding element (child of the <job> element) has been parsed

Definition at line 303 of file OSrLParserData.h.

6.202.3.97 bool OSrLParserData::jobSubmitTimePresent

Definition at line 304 of file OSrLParserData.h.

6.202.3.98 bool OSrLParserData::scheduledStartTimePresent

Definition at line 305 of file OSrLParserData.h.

6.202.3.99 bool OSrLParserData::actualStartTimePresent

Definition at line 306 of file OSrLParserData.h.

6.202.3.100 bool OSrLParserData::jobEndTimePresent

Definition at line 307 of file OSrLParserData.h.

6.202.3.101 bool OSrLParserData::jobTimingInformationPresent

Definition at line 308 of file OSrLParserData.h.

6.202.3.102 bool OSrLParserData::jobUsedDiskSpacePresent

Definition at line 309 of file OSrLParserData.h.

6.202.3.103 bool OSrLParserData::jobUsedMemoryPresent

Definition at line 310 of file OSrLParserData.h.

6.202.3.104 bool OSrLParserData::jobUsedCPUSpeedPresent

Definition at line 311 of file OSrLParserData.h.

6.202.3.105 bool OSrLParserData::jobUsedCPUNumberPresent

Definition at line 312 of file OSrLParserData.h.

6.202.3.106 `bool OSrLParserData::otherJobResultsPresent`

Definition at line 313 of file OSrLParserData.h.

6.202.3.107 `bool OSrLParserData::numberOfItemsPresent`

Definition at line 315 of file OSrLParserData.h.

6.202.3.108 `int OSrLParserData::numberOfItems`

Definition at line 316 of file OSrLParserData.h.

6.202.3.109 `struct IndexValuePair* OSrLParserData::primalValPair`

for each solution we will build a vector of index-value pairs of primal values

Definition at line 321 of file OSrLParserData.h.

6.202.3.110 `std::vector<IndexValuePair*> OSrLParserData::primalVals`

Definition at line 322 of file OSrLParserData.h.

6.202.3.111 `struct IndexValuePair* OSrLParserData::objValPair`

for each solution we will build a vector of index-value pairs of objective function values

Definition at line 327 of file OSrLParserData.h.

6.202.3.112 `std::vector<IndexValuePair*> OSrLParserData::objVals`

Definition at line 328 of file OSrLParserData.h.

6.202.3.113 `struct IndexValuePair* OSrLParserData::dualValPair`

for each solution we will build a vector of index-value pairs of dual values

Definition at line 333 of file OSrLParserData.h.

6.202.3.114 `std::vector<IndexValuePair*> OSrLParserData::dualVals`

Definition at line 334 of file OSrLParserData.h.

6.202.3.115 `struct OtherVariableResultStruct* OSrLParserData::otherVarStruct`

a pointer to an [OtherVariableResultStruct](#) structure

Definition at line 338 of file OSrLParserData.h.

6.202.3.116 `std::vector<OtherVariableResultStruct*> OSrLParserData::otherVarVec`

store a vector of pointers to otherVarVec structures

Definition at line 341 of file OSrLParserData.h.

6.202.3.117 `char* OSrLParserData::errorText`

if the parser finds invalid text it is held here and we delete if the file was not valid

Definition at line 346 of file OSrLParserData.h.

6.202.3.118 `std::string OSrLParserData::parser_errors`

used to accumulate error message so the parser does not die on the first error encountered

Definition at line 351 of file OSrLParserData.h.

6.202.3.119 `bool OSrLParserData::ignoreDataAfterErrors`

two booleans to govern the behavior after an error has been encountered

Definition at line 354 of file OSrLParserData.h.

6.202.3.120 `bool OSrLParserData::suppressFurtherErrorMessages`

Definition at line 355 of file OSrLParserData.h.

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

- [/home/ted/COIN/trunk/OS/src/OSParsers/OSrLParserData.h](#)

6.203 OSrLReader Class Reference

The [OSrLReader](#) Class.

```
#include <OSrLReader.h>
```

Public Member Functions

- [OSrLReader](#) ()
OSrL class constructor.
- [~OSrLReader](#) ()
OSrL class destructor.
- [OSResult * readOSrL](#) (const std::string &posrl) throw (ErrorClass)
Get an [OSResult](#) object from an OSrL string.

6.203.1 Detailed Description

The [OSrLReader](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for parsing an OSrL string and creating an [OSResult](#) object from the string.

Definition at line 42 of file OSrLReader.h.

6.203.2 Constructor & Destructor Documentation

6.203.2.1 OSrLReader::OSrLReader ()

OSrL class constructor.

6.203.2.2 OSrLReader::~~OSrLReader ()

OSrL class destructor.

6.203.3 Member Function Documentation

6.203.3.1 OSResult* OSrLReader::readOSrL (const std::string & *posrl*) throw (ErrorClass)

Get an [OSResult](#) object from an OSrL string.

Parameters

<i>osrl</i>	an OSrL string.
-------------	-----------------

Returns

the [OSResult](#) object corresponding to the OSrL string.

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

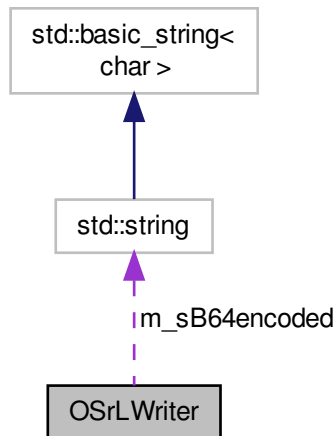
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLReader.h](#)

6.204 OSrLWriter Class Reference

Take an [OSResult](#) object and write a string that validates against OSrL.

```
#include "OSrLWriter.h"
```

Collaboration diagram for OSrLWriter:



Public Member Functions

- [OSrLWriter \(\)](#)
Default constructor.
- [~OSrLWriter \(\)](#)
Class destructor.
- `std::string` [writeOSrL \(OSResult *theosresult\)](#)
create an osrl string from an [OSResult](#) object

Public Attributes

- `bool` [m_bWriteBase64](#)
m_bWriteBase64 is set to true if we encode the linear constraint coefficients in base64 binary
- `bool` [m_bWhiteSpace](#)
m_bWhiteSpace is set to true if we write white space in the file
- `std::string` [m_sB64encoded](#)
m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

6.204.1 Detailed Description

Take an [OSResult](#) object and write a string that validates against OSrL.

Definition at line 30 of file OSrLWriter.h.

6.204.2 Constructor & Destructor Documentation

6.204.2.1 OSrLWriter::OSrLWriter ()

Default constructor.

6.204.2.2 OSrLWriter::~~OSrLWriter ()

Class destructor.

6.204.3 Member Function Documentation

6.204.3.1 std::string OSrLWriter::writeOSrL (OSResult * *theosresult*)

create an osrl string from an [OSResult](#) object

Parameters

<i>theosresult</i>	is a pointer to an OSResult object
--------------------	--

Returns

a string with the [OSResult](#) data that validates against the OSrL schema.

6.204.4 Member Data Documentation

6.204.4.1 bool OSrLWriter::m_bWriteBase64

m_bWriteBase64 is set to true if we encode the linear constraint coefficients in base64 binary

Definition at line 65 of file OSrLWriter.h.

6.204.4.2 bool OSrLWriter::m_bWhiteSpace

m_bWhiteSpace is set to true if we write white space in the file

Definition at line 69 of file OSrLWriter.h.

6.204.4.3 std::string OSrLWriter::m_sB64encoded

m_sB64encoded is a string of data (start, colldx, rowldx, or values) from linear constraints coefficients encoded in base64 binary

Definition at line 74 of file OSrLWriter.h.

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

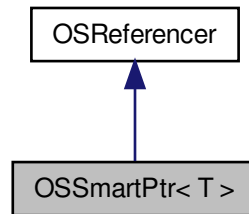
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSrLWriter.h](#)

6.205 OSSmartPtr< T > Class Template Reference

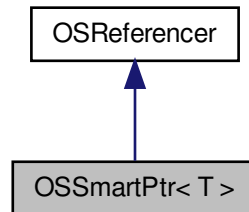
Template class for Smart Pointers.

```
#include <OSSmartPtr.hpp>
```

Inheritance diagram for OSSmartPtr< T >:



Collaboration diagram for OSSmartPtr< T >:



Public Member Functions

Constructors/Destructors

- [OSSmartPtr](#) ()
Default constructor, initialized to NULL.
- [OSSmartPtr](#) (const [OSSmartPtr](#)< T > ©)
Copy constructor, initialized from copy.
- [OSSmartPtr](#) (T *ptr)
Constructor, initialized from T ptr.*
- [~OSSmartPtr](#) ()
Destructor, automatically decrements the reference count, deletes the object if necessary.

Friends

friend method declarations.

- template<class U >
U * [GetRawPtr](#) (const [OSSmartPtr](#)< U > &smart_ptr)

Returns the raw pointer contained.

- `template<class U >`
`OSSmartPtr< const U > ConstPtr (const OSSmartPtr< U > &smart_ptr)`

Returns a const pointer.

- `template<class U >`
`bool IsValid (const OSSmartPtr< U > &smart_ptr)`

Returns true if the OSSmartPtr is NOT NULL.

- `template<class U >`
`bool IsNull (const OSSmartPtr< U > &smart_ptr)`

Returns true if the OSSmartPtr is NULL.

Overloaded operators.

- `T * operator-> () const`
Overloaded arrow operator, allows the user to call methods using the contained pointer.
- `T & operator* () const`
Overloaded dereference operator, allows the user to dereference the contained pointer.
- `OSSmartPtr< T > & operator= (T *rhs)`
Overloaded equals operator, allows the user to set the value of the OSSmartPtr from a raw pointer.
- `OSSmartPtr< T > & operator= (const OSSmartPtr< T > &rhs)`
Overloaded equals operator, allows the user to set the value of the OSSmartPtr from another OSSmartPtr.
- `template<class U1 , class U2 >`
`bool operator== (const OSSmartPtr< U1 > &lhs, const OSSmartPtr< U2 > &rhs)`
Overloaded equality comparison operator, allows the user to compare the value of two OSSmartPtrs.
- `template<class U1 , class U2 >`
`bool operator== (const OSSmartPtr< U1 > &lhs, U2 *raw_rhs)`
Overloaded equality comparison operator, allows the user to compare the value of an OSSmartPtr with a raw pointer.
- `template<class U1 , class U2 >`
`bool operator== (U1 *lhs, const OSSmartPtr< U2 > &raw_rhs)`
Overloaded equality comparison operator, allows the user to compare the value of a raw pointer with an OSSmartPtr.
- `template<class U1 , class U2 >`
`bool operator!= (const OSSmartPtr< U1 > &lhs, const OSSmartPtr< U2 > &rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of two OSSmartPtrs.
- `template<class U1 , class U2 >`
`bool operator!= (const OSSmartPtr< U1 > &lhs, U2 *raw_rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of an OSSmartPtr with a raw pointer.
- `template<class U1 , class U2 >`
`bool operator!= (U1 *lhs, const OSSmartPtr< U2 > &raw_rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of an OSSmartPtr with a raw pointer.

6.205.1 Detailed Description

`template<class T>class OSSmartPtr< T >`

Template class for Smart Pointers.

An `OSSmartPtr` behaves much like a raw pointer, but manages the lifetime of an object, deleting the object automatically. This class implements a reference-counting, intrusive smart pointer design, where all objects pointed to must inherit from `ReferencedObject`, which stores the reference count. Although this is intrusive (native types and externally authored

classes require wrappers to be referenced by smart pointers), it is a safer design. A more detailed discussion of these issues follows after the usage information.

Usage Example: Note: to use the [OSSmartPtr](#), all objects to which you point MUST inherit from ReferencedObject.

```
*
* In MyClass.hpp...
*
* #include "OSReferenced.hpp"
*
* class MyClass : public ReferencedObject // must derive from ReferencedObject
* {
*     ...
* }
*
* In my_usage.cpp...
*
* #include "OSSmartPtr.hpp"
* #include "MyClass.hpp"
*
* void func(AnyObject& obj)
* {
*     OSSmartPtr<MyClass> ptr_to_myclass = new MyClass(...);
*     // ptr_to_myclass now points to a new MyClass,
*     // and the reference count is 1
*
*     ...
*
*     obj.SetMyClass(ptr_to_myclass);
*     // Here, let's assume that AnyObject uses a
*     // OSSmartPtr<MyClass> internally here.
*     // Now, both ptr_to_myclass and the internal
*     // OSSmartPtr in obj point to the same MyClass object
*     // and its reference count is 2.
*
*     ...
*
*     // No need to delete ptr_to_myclass, this
*     // will be done automatically when the
*     // reference count drops to zero.
*
* }
*
*
```

It is not necessary to use [OSSmartPtr](#)'s in all cases where an object is used that has been allocated "into" a [OSSmartPtr](#). It is possible to just pass objects by reference or regular pointers, even if lower down in the stack an [OSSmartPtr](#) is to be held on to. Everything should work fine as long as a pointer created by "new" is immediately passed into an [OSSmartPtr](#), and if [OSSmartPtr](#)'s are used to hold on to objects.

Other Notes: The [OSSmartPtr](#) implements both dereference operators -> & *. The [OSSmartPtr](#) does NOT implement a conversion operator to the raw pointer. Use the [GetRawPtr\(\)](#) method when this is necessary. Make sure that the raw pointer is NOT deleted. The [OSSmartPtr](#) implements the comparison operators == & != for a variety of types. Use these instead of

```
*     if (GetRawPtr(smrt_ptr) == ptr) // Don't use this
*
```

[OSSmartPtr](#)'s, as currently implemented, do NOT handle circular references. For example: consider a higher level object using [OSSmartPtr](#)s to point to A and B, but A and B also point to each other (i.e. A has an [OSSmartPtr](#) to B and B has an [OSSmartPtr](#) to A). In this scenario, when the higher level object is finished with A and B, their reference counts will never drop to zero (since they reference each other) and they will not be deleted. This can be detected by memory leak tools like valgrind. If the circular reference is necessary, the problem can be overcome by a number of techniques:

1) A and B can have a method that "releases" each other, that is they set their internal [OSSmartPtr](#)s to NULL.

```

*         void AClass::ReleaseCircularReferences()
*         {
*             smart_ptr_to_B = NULL;
*         }
*

```

Then, the higher level class can call these methods before it is done using A & B.

2) Raw pointers can be used in A and B to reference each other. Here, an implicit assumption is made that the lifetime is controlled by the higher level object and that A and B will both exist in a controlled manner. Although this seems dangerous, in many situations, this type of referencing is very controlled and this is reasonably safe.

3) This [OSSmartPtr](#) class could be redesigned with the Weak/Strong design concept. Here, the [OSSmartPtr](#) is identified as being Strong (controls lifetime of the object) or Weak (merely referencing the object). The Strong [OSSmartPtr](#) increments (and decrements) the reference count in ReferencedObject but the Weak [OSSmartPtr](#) does not. In the example above, the higher level object would have Strong OSSmartPtrs to A and B, but A and B would have Weak OSSmartPtrs to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak OSSmartPtrs in A and B would not decrement the reference count and would, of course, not delete the object. This idea is very similar to item (2), where it is implied that the sequence of events is controlled such that A and B will not call anything using their pointers following the higher level delete (i.e. in their destructors!). This is somehow safer, however, because code can be written (however expensive) to perform run-time detection of this situation. For example, the ReferencedObject could store pointers to all Weak OSSmartPtrs that are referencing it and, in its destructor, tell these pointers that it is dying. They could then set themselves to NULL, or set an internal flag to detect usage past this point.

Comments on Non-Intrusive Design: In a non-intrusive design, the reference count is stored somewhere other than the object being referenced. This means, unless the reference counting pointer is the first referencer, it must get a pointer to the referenced object from another smart pointer (so it has access to the reference count location). In this non-intrusive design, if we are pointing to an object with a smart pointer (or a number of smart pointers), and we then give another smart pointer the address through a RAW pointer, we will have two independent, AND INCORRECT, reference counts. To avoid this pitfall, we use an intrusive reference counting technique where the reference count is stored in the object being referenced.

Definition at line 156 of file OSSmartPtr.hpp.

6.205.2 Constructor & Destructor Documentation

6.205.2.1 `template<class T> OSSmartPtr< T >::OSSmartPtr ()`

Default constructor, initialized to NULL.

Definition at line 328 of file OSSmartPtr.hpp.

6.205.2.2 `template<class T> OSSmartPtr< T >::OSSmartPtr (const OSSmartPtr< T > & copy)`

Copy constructor, initialized from copy.

Definition at line 344 of file OSSmartPtr.hpp.

6.205.2.3 `template<class T> OSSmartPtr< T >::OSSmartPtr (T* ptr)`

Constructor, initialized from T* ptr.

Definition at line 361 of file OSSmartPtr.hpp.

6.205.2.4 `template<class T> OSSmartPtr< T >::~~OSSmartPtr ()`

Destructor, automatically decrements the reference count, deletes the object if necessary.

Definition at line 376 of file OSSmartPtr.hpp.

6.205.3 Member Function Documentation

6.205.3.1 `template<class T> T * OSSmartPtr< T >::operator-> () const`

Overloaded arrow operator, allows the user to call methods using the contained pointer.

Definition at line 382 of file OSSmartPtr.hpp.

6.205.3.2 `template<class T> T & OSSmartPtr< T >::operator* () const`

Overloaded dereference operator, allows the user to dereference the contained pointer.

Definition at line 389 of file OSSmartPtr.hpp.

6.205.3.3 `template<class T> OSSmartPtr< T > & OSSmartPtr< T >::operator= (T * rhs)`

Overloaded equals operator, allows the user to set the value of the [OSSmartPtr](#) from a raw pointer.

Definition at line 396 of file OSSmartPtr.hpp.

6.205.3.4 `template<class T> OSSmartPtr< T > & OSSmartPtr< T >::operator= (const OSSmartPtr< T > & rhs)`

Overloaded equals operator, allows the user to set the value of the [OSSmartPtr](#) from another [OSSmartPtr](#).

Definition at line 403 of file OSSmartPtr.hpp.

6.205.4 Friends And Related Function Documentation

6.205.4.1 `template<class T> template<class U1 , class U2 > bool operator== (const OSSmartPtr< U1 > & lhs, const OSSmartPtr< U2 > & rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of two OSSmartPtrs.

6.205.4.2 `template<class T> template<class U1 , class U2 > bool operator== (const OSSmartPtr< U1 > & lhs, U2 * raw_rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of an [OSSmartPtr](#) with a raw pointer.

6.205.4.3 `template<class T> template<class U1 , class U2 > bool operator== (U1 * lhs, const OSSmartPtr< U2 > & raw_rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of a raw pointer with an [OSSmartPtr](#).

6.205.4.4 `template<class T> template<class U1 , class U2 > bool operator!= (const OSSmartPtr< U1 > & lhs, const OSSmartPtr< U2 > & rhs) [friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of two OSSmartPtrs.

6.205.4.5 `template<class T> template<class U1 , class U2 > bool operator!= (const OSSmartPtr< U1 > & lhs, U2 * raw_rhs) [friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of an [OSSmartPtr](#) with a raw pointer.

6.205.4.6 `template<class T> template<class U1 , class U2 > bool operator!=(U1 * lhs, const OSSmartPtr< U2 > & raw_rhs)`
`[friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of an [OSSmartPtr](#) with a raw pointer.

6.205.4.7 `template<class T> template<class U > U* GetRawPtr (const OSSmartPtr< U > & smart_ptr)` `[friend]`

Returns the raw pointer contained.

Use to get the value of the raw ptr (i.e. to pass to other methods/functions, etc.) Note: This method does NOT copy, therefore, modifications using this value modify the underlying object contained by the [OSSmartPtr](#), NEVER delete this returned value.

6.205.4.8 `template<class T> template<class U > OSSmartPtr<const U> ConstPtr (const OSSmartPtr< U > & smart_ptr)`
`[friend]`

Returns a const pointer.

6.205.4.9 `template<class T> template<class U > bool IsValid (const OSSmartPtr< U > & smart_ptr)` `[friend]`

Returns true if the [OSSmartPtr](#) is NOT NULL.

Use this to check if the [OSSmartPtr](#) is not null This is preferred to `if(GetRawPtr(sp) != NULL)`

6.205.4.10 `template<class T> template<class U > bool IsNull (const OSSmartPtr< U > & smart_ptr)` `[friend]`

Returns true if the [OSSmartPtr](#) is NULL.

Use this to check if the [OSSmartPtr](#) IsNull. This is preferred to `if(GetRawPtr(sp) == NULL)`

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

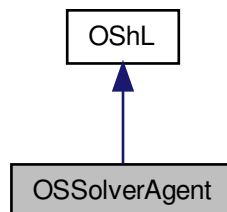
- `/home/ted/COIN/trunk/OS/src/OSUtils/OSSmartPtr.hpp`

6.206 OSSolverAgent Class Reference

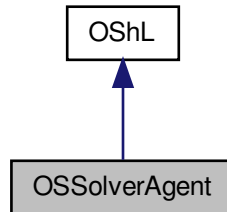
Used by a client to invoke a remote solver.

```
#include "OSSolverAgent.h"
```

Inheritance diagram for OSSolverAgent:



Collaboration diagram for OSSolverAgent:



Public Member Functions

- [OSSolverAgent](#) (std::string solverURI)
Default constructor.
- [~OSSolverAgent](#) ()
Class destructor.
- std::string [solve](#) (std::string osil, std::string osol)
implement the [solve\(\)](#) method which is a virtual function in [OShL](#), this is synchronous
- std::string [getJobID](#) (std::string osol)
implement the [getJobID\(\)](#) method which is a virtual function in [OShL](#)
- bool [send](#) (std::string osil, std::string osol)
implement the [send\(\)](#) method which is a virtual function in [OShL](#)
- std::string [kill](#) (std::string osol)
implement the [kill\(\)](#) method which is a virtual function in [OShL](#)
- std::string [retrieve](#) (std::string osol)
implement the [retrieve\(\)](#) method which is a virtual function in [OShL](#)
- std::string [knock](#) (std::string ospl, std::string osol)
implement the [knock\(\)](#) method which is a virtual function in [OShL](#)
- std::string [fileUpload](#) (std::string osilFileName, std::string osil)
implement the [fileUpload\(\)](#) method which is a virtual function in [OShL](#)

6.206.1 Detailed Description

Used by a client to invoke a remote solver.

Remarks

This is an implementation of the virtual class [OShL](#). We need to implement the following virtual methods.

The following key methods are invoked:

1. [solve](#)

2. kill
3. send
4. retrieve
5. knock
6. getJobID

Definition at line 41 of file OSSolverAgent.h.

6.206.2 Constructor & Destructor Documentation

6.206.2.1 OSSolverAgent::OSSolverAgent (std::string *solverURI*)

Default constructor.

Parameters

<i>solverURI</i>	is the location of remote solver or scheduler
------------------	---

6.206.2.2 OSSolverAgent::~~OSSolverAgent ()

Class destructor.

6.206.3 Member Function Documentation

6.206.3.1 std::string OSSolverAgent::solve (std::string *osil*, std::string *osol*) [virtual]

implement the [solve\(\)](#) method which is a virtual function in [OShL](#), this is synchronous

Parameters

<i>osil</i>	a string that holds the problem instance
<i>osol</i>	is a string of options for the solver

Returns

osrl which is a string with the result.

Implements [OShL](#).

6.206.3.2 std::string OSSolverAgent::getJobID (std::string *osol*) [virtual]

implement the [getJobID\(\)](#) method which is a virtual function in [OShL](#)

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is the jobId

Implements [OShL](#).

6.206.3.3 `bool OSSolverAgent::send (std::string osil, std::string osol)` `[virtual]`

implement the [send\(\)](#) method which is a virtual function in [OShL](#)

Parameters

<i>osil</i>	is the string with the instance in OSiL format
<i>osol</i>	is the string with the options in OSoL format

Returns

a bool which is true if the job is successfully submitted

Implements [OShL](#).

6.206.3.4 `std::string OSSolverAgent::kill (std::string osol)` `[virtual]`

implement the [kill\(\)](#) method which is a virtual function in [OShL](#)

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is in OSpL format

Implements [OShL](#).

6.206.3.5 `std::string OSSolverAgent::retrieve (std::string osol)` `[virtual]`

implement the [retrieve\(\)](#) method which is a virtual function in [OShL](#)

Parameters

<i>osol</i>	is the string with the options in OSoL format
-------------	---

Returns

a string which is in the result of the optimization is OSrL fomrat

Implements [OShL](#).

6.206.3.6 `std::string OSSolverAgent::knock (std::string ospl, std::string osol)` `[virtual]`

implement the [knock\(\)](#) method which is a virtual function in [OShL](#)

Parameters

<i>ospl</i>	is the string with the process information in OSpL format
<i>osol</i>	is the string with the options in OSoL format

Returns

a string which is the knock result in OSpL format.

Implements [OShL](#).

6.206.3.7 `std::string OSSolverAgent::fileUpload (std::string osilFileName, std::string osil)`

implement the `fileUpload()` method which is a virtual function in [OShL](#)

Parameters

<i>osilFileName</i>	is the name of the file with the OSiL instance to be written on the server
<i>osil</i>	is a string with the OSiL problem instance

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

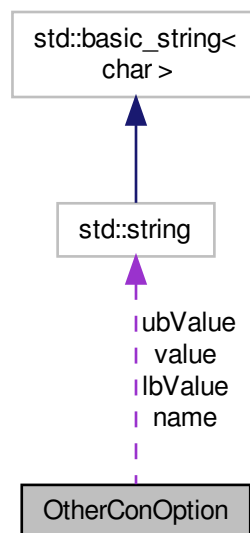
- `/home/ted/COIN/trunk/OS/src/OSAgent/OSSolverAgent.h`

6.207 OtherConOption Class Reference

the [OtherConOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherConOption:

**Public Member Functions**

- [OtherConOption](#) ()

Default constructor.

- [~OtherConOption](#) ()

Class destructor.

- bool [IsEqual](#) ([OtherConOption](#) *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant)

A function to make a random instance of this class.

- bool [deepCopyFrom](#) ([OtherConOption](#) *that)

A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)

variable index

- std::string [name](#)

optional variable name

- std::string [value](#)

value of the option

- std::string [lbValue](#)

lower bound of the option

- std::string [ubValue](#)

upper bound of the option

6.207.1 Detailed Description

the [OtherConOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3091 of file OSOption.h.

6.207.2 Constructor & Destructor Documentation

6.207.2.1 OtherConOption::OtherConOption ()

Default constructor.

6.207.2.2 OtherConOption::~~OtherConOption ()

Class destructor.

6.207.3 Member Function Documentation

6.207.3.1 bool OtherConOption::isEqual (OtherConOption * *that*)

A function to check for the equality of two objects.

6.207.3.2 bool OtherConOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.207.3.3 bool OtherConOption::deepCopyFrom (OtherConOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.207.4 Member Data Documentation

6.207.4.1 int OtherConOption::idx

variable index

Definition at line 3096 of file OSOption.h.

6.207.4.2 std::string OtherConOption::name

optional variable name

Definition at line 3099 of file OSOption.h.

6.207.4.3 std::string OtherConOption::value

value of the option

Definition at line 3102 of file OSOption.h.

6.207.4.4 std::string OtherConOption::lbValue

lower bound of the option

Definition at line 3105 of file OSOption.h.

6.207.4.5 std::string OtherConOption::ubValue

upper bound of the option

Definition at line 3108 of file OSOption.h.

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

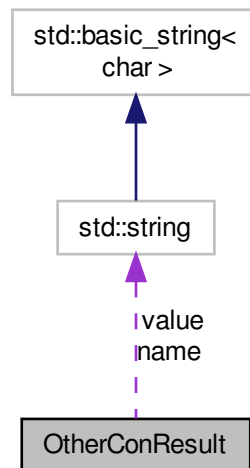
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.208 OtherConResult Class Reference

The [OtherConResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherConResult:



Public Member Functions

- [OtherConResult](#) ()
Default constructor.
- [~OtherConResult](#) ()
Class destructor.
- bool [IsEqual](#) ([OtherConResult](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [idx](#)
idx is the index on the constraint
- std::string [name](#)
optional name
- std::string [value](#)
value is a value associated with the constraint indexed by idx, for example value might be the value of a dual variable or it might be the name of the constraint.

6.208.1 Detailed Description

The [OtherConResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides general result information for constraints.

Definition at line 1695 of file OSResult.h.

6.208.2 Constructor & Destructor Documentation

6.208.2.1 OtherConResult::OtherConResult ()

Default constructor.

6.208.2.2 OtherConResult::~~OtherConResult ()

Class destructor.

6.208.3 Member Function Documentation

6.208.3.1 bool OtherConResult::isEqual (OtherConResult * that)

A function to check for the equality of two objects.

6.208.3.2 bool OtherConResult::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.208.4 Member Data Documentation

6.208.4.1 int OtherConResult::idx

idx is the index on the constraint

Definition at line 1700 of file OSResult.h.

6.208.4.2 std::string OtherConResult::name

optional name

Definition at line 1703 of file OSResult.h.

6.208.4.3 std::string OtherConResult::value

value is a value associated with the constraint indexed by idx, for example value might be the value of a dual variable or it might be the name of the constraint.

Definition at line 1710 of file OSResult.h.

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

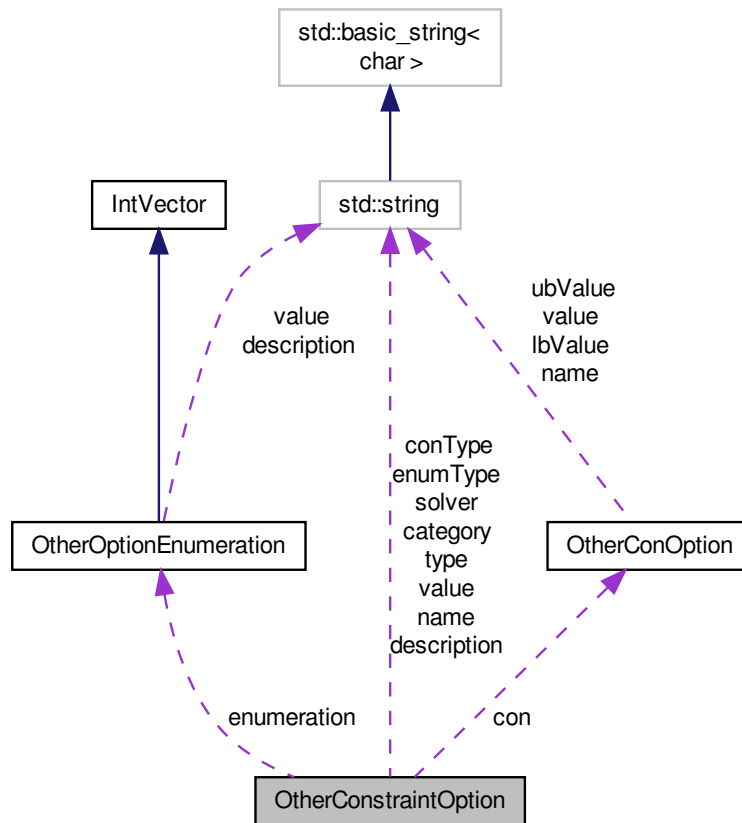
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSResult.h](#)

6.209 OtherConstraintOption Class Reference

the [OtherConstraintOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherConstraintOption:



Public Member Functions

- `OtherConstraintOption ()`
Default constructor.
- `~OtherConstraintOption ()`
Class destructor.
- `bool IsEqual (OtherConstraintOption *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (OtherConstraintOption *that)`
A function to make a deep copy of an instance of this class.
- `bool setCon (int numberOfCon, OtherConOption **con)`
A function to set an array of <con> elements.
- `bool addCon (int idx, std::string value, std::string lbValue, std::string ubValue)`
A function to add a <con> element.

Public Attributes

- int [numberOfCon](#)
number of <con> children
- int [numberOfEnumerations](#)
number of <enumeration> child elements
- std::string [name](#)
name of the option
- std::string [value](#)
value of the option
- std::string [solver](#)
name of the solver to which this option applies
- std::string [category](#)
name of the category into which this option falls
- std::string [type](#)
type of the option value (integer, double, boolean, string)
- std::string [description](#)
description of the option
- [OtherConOption](#) ** [con](#)
array of option values
- std::string [conType](#)
type of the values in the con array
- [OtherOptionEnumeration](#) ** [enumeration](#)
- std::string [enumType](#)
type of the values in the enumeration array

6.209.1 Detailed Description

the [OtherConstraintOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3156 of file OSOption.h.

6.209.2 Constructor & Destructor Documentation

6.209.2.1 OtherConstraintOption::OtherConstraintOption ()

Default constructor.

6.209.2.2 OtherConstraintOption::~~OtherConstraintOption ()

Class destructor.

6.209.3 Member Function Documentation

6.209.3.1 bool OtherConstraintOption::isEqual (OtherConstraintOption * *that*)

A function to check for the equality of two objects.

6.209.3.2 bool OtherConstraintOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.209.3.3 bool OtherConstraintOption::deepCopyFrom (OtherConstraintOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.209.3.4 bool OtherConstraintOption::setCon (int *numberOfCon*, OtherConOption ** *con*)

A function to set an array of <con> elements.

Parameters

<i>numberOfCon</i> ,:	number of <con> elements to be set
<i>obj</i> ,:	the array of <con> elements that are to be set

6.209.3.5 bool OtherConstraintOption::addCon (int *idx*, std::string *value*, std::string *lbValue*, std::string *ubValue*)

A function to add a <con> element.

Parameters

<i>idx</i> ,:	the index of the constraint
<i>value</i> ,:	the value associated with this constraint
<i>lbValue</i> ,:	a lower bound associated with this constraint
<i>ubValue</i> ,:	an upper bound associated with this constraint

6.209.4 Member Data Documentation

6.209.4.1 int OtherConstraintOption::numberOfCon

number of <con> children

Definition at line 3161 of file OSOption.h.

6.209.4.2 int OtherConstraintOption::numberOfEnumerations

number of <enumeration> child elements

Definition at line 3164 of file OSOption.h.

6.209.4.3 std::string OtherConstraintOption::name

name of the option

Definition at line 3167 of file OSOption.h.

6.209.4.4 std::string OtherConstraintOption::value

value of the option

Definition at line 3170 of file OSOption.h.

6.209.4.5 std::string OtherConstraintOption::solver

name of the solver to which this option applies

Definition at line 3173 of file OSOption.h.

6.209.4.6 std::string OtherConstraintOption::category

name of the category into which this option falls

Definition at line 3176 of file OSOption.h.

6.209.4.7 std::string OtherConstraintOption::type

type of the option value (integer, double, boolean, string)

Definition at line 3179 of file OSOption.h.

6.209.4.8 std::string OtherConstraintOption::description

description of the option

Definition at line 3182 of file OSOption.h.

6.209.4.9 OtherConOption** OtherConstraintOption::con

array of option values

Definition at line 3185 of file OOption.h.

6.209.4.10 `std::string OtherConstraintOption::conType`

type of the values in the con array

Definition at line 3188 of file OOption.h.

6.209.4.11 `OtherOptionEnumeration** OtherConstraintOption::enumeration`

Definition at line 3194 of file OOption.h.

6.209.4.12 `std::string OtherConstraintOption::enumType`

type of the values in the enumeration array

Definition at line 3197 of file OOption.h.

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

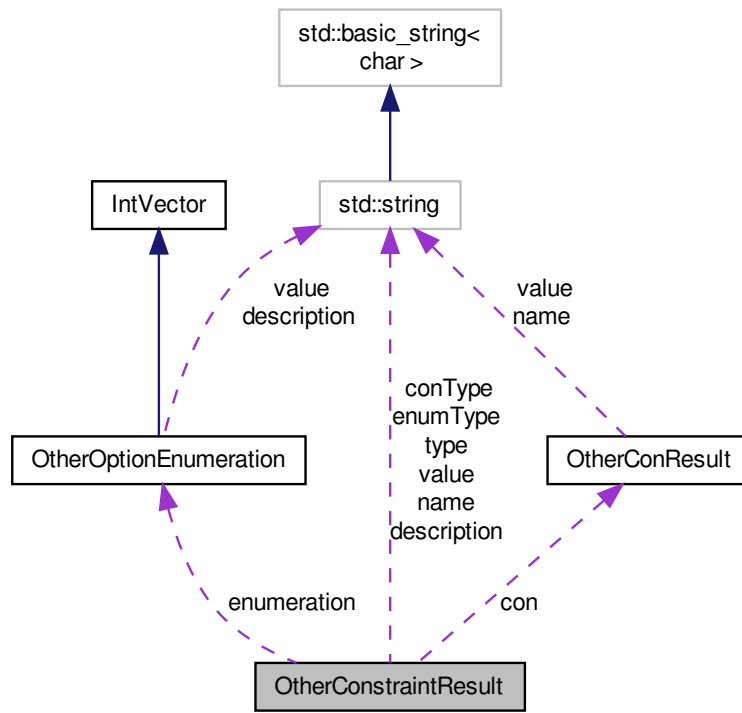
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OOption.h](#)

6.210 OtherConstraintResult Class Reference

The [OtherConstraintResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherConstraintResult:



Public Member Functions

- `OtherConstraintResult ()`
Default constructor.
- `~OtherConstraintResult ()`
Class destructor.
- `bool isEqual (OtherConstraintResult *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfCon`
the number of constraints which have values for this particular type of result
- `int numberOfEnumerations`
the number of distinct values for this particular type of result
- `std::string name`
the name of the result the user is defining

- `std::string value`
this element allows a specific value associated with this particular type of result
- `std::string type`
type of the result value (integer, double, boolean, string)
- `std::string description`
a brief description of the type of result
- `OtherConResult ** con`
- `std::string conType`
type of the values in the con array
- `OtherOptionEnumeration ** enumeration`
- `std::string enumType`
type of the values in the enumeration array

6.210.1 Detailed Description

The `OtherConstraintResult` Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with constraints.

Definition at line 1753 of file OSResult.h.

6.210.2 Constructor & Destructor Documentation

6.210.2.1 `OtherConstraintResult::OtherConstraintResult ()`

Default constructor.

6.210.2.2 `OtherConstraintResult::~~OtherConstraintResult ()`

Class destructor.

6.210.3 Member Function Documentation

6.210.3.1 `bool OtherConstraintResult::isEqual (OtherConstraintResult * that)`

A function to check for the equality of two objects.

6.210.3.2 bool OtherConstraintResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.210.4 Member Data Documentation**6.210.4.1 int OtherConstraintResult::numberOfCon**

the number of constraints which have values for this particular type of result

Definition at line 1760 of file OSResult.h.

6.210.4.2 int OtherConstraintResult::numberOfEnumerations

the number of distinct values for this particular type of result

Definition at line 1765 of file OSResult.h.

6.210.4.3 std::string OtherConstraintResult::name

the name of the result the user is defining

Definition at line 1768 of file OSResult.h.

6.210.4.4 std::string OtherConstraintResult::value

this element allows a specific value associated with this particular type of result

Definition at line 1773 of file OSResult.h.

6.210.4.5 std::string OtherConstraintResult::type

type of the result value (integer, double, boolean, string)

Definition at line 1776 of file OSResult.h.

6.210.4.6 std::string OtherConstraintResult::description

a brief description of the type of result

Definition at line 1779 of file OSResult.h.

6.210.4.7 OtherConResult OtherConstraintResult::con**

Definition at line 1786 of file OSResult.h.

6.210.4.8 std::string OtherConstraintResult::conType

type of the values in the con array

Definition at line 1789 of file OSResult.h.

6.210.4.9 OtherOptionEnumeration** OtherConstraintResult::enumeration

Definition at line 1795 of file OSResult.h.

6.210.4.10 std::string OtherConstraintResult::enumType

type of the values in the enumeration array

Definition at line 1798 of file OSResult.h.

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

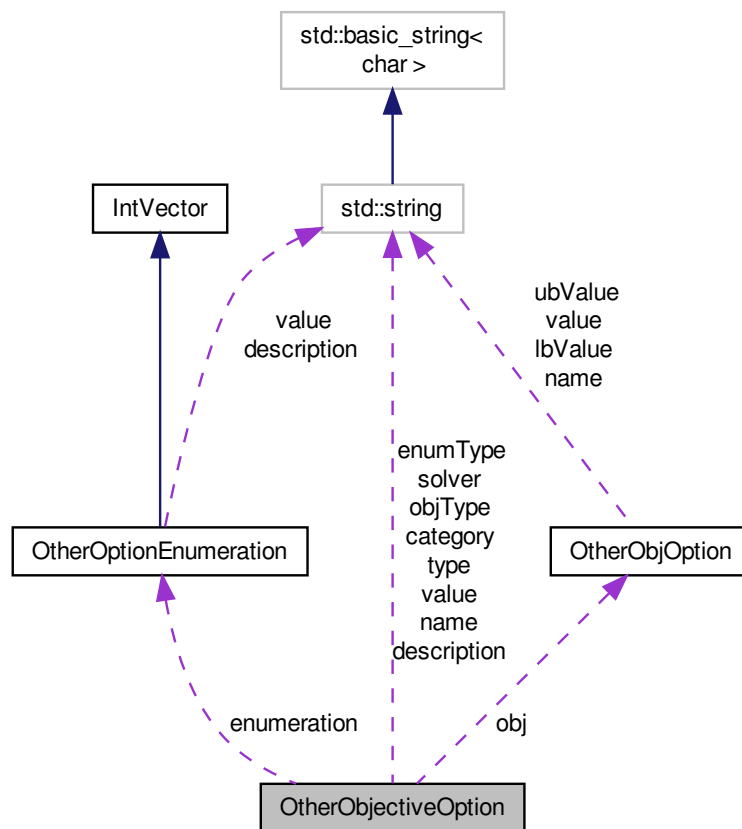
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.211 OtherObjectiveOption Class Reference

the [OtherObjectiveOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherObjectiveOption:



Public Member Functions

- [OtherObjectiveOption](#) ()
Default constructor.
- [~OtherObjectiveOption](#) ()
Class destructor.
- bool [isEqual](#) ([OtherObjectiveOption](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([OtherObjectiveOption](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setObj](#) (int [numberOfObj](#), [OtherObjOption](#) **obj)
A function to set an array of <obj> elements.
- bool [addObj](#) (int idx, std::string [value](#), std::string lbValue, std::string ubValue)
A function to add a <obj> element.

Public Attributes

- int [numberOfObj](#)
number of <obj> children
- int [numberOfEnumerations](#)
number of <enumeration> child elements
- std::string [name](#)
name of the option
- std::string [value](#)
value of the option
- std::string [solver](#)
name of the solver to which this option applies
- std::string [category](#)
name of the category into which this option falls
- std::string [type](#)
type of the option value (integer, double, boolean, string)
- std::string [description](#)
description of the option
- [OtherObjOption](#) ** [obj](#)
array of option values
- std::string [objType](#)
type of the values in the obj array
- [OtherOptionEnumeration](#) ** [enumeration](#)
- std::string [enumType](#)
type of the values in the enumeration array

6.211.1 Detailed Description

the [OtherObjectiveOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2574 of file OSOption.h.

6.211.2 Constructor & Destructor Documentation

6.211.2.1 OtherObjectiveOption::OtherObjectiveOption ()

Default constructor.

6.211.2.2 OtherObjectiveOption::~~OtherObjectiveOption ()

Class destructor.

6.211.3 Member Function Documentation

6.211.3.1 bool OtherObjectiveOption::isEqual (OtherObjectiveOption * *that*)

A function to check for the equality of two objects.

6.211.3.2 bool OtherObjectiveOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.211.3.3 bool OtherObjectiveOption::deepCopyFrom (OtherObjectiveOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.211.3.4 `bool OtherObjectiveOption::setObj (int numberOfObj, OtherObjOption ** obj)`

A function to set an array of <obj> elements.

Parameters

<i>numberOfObj,:</i>	number of <obj> elements to be set
<i>obj,:</i>	the array of <obj> elements that are to be set

6.211.3.5 `bool OtherObjectiveOption::addObj (int idx, std::string value, std::string lbValue, std::string ubValue)`

A function to add a <obj> element.

Parameters

<i>idx,:</i>	the index of the objective
<i>value,:</i>	the value associated with this objective
<i>lbValue,:</i>	a lower bound associated with this objective
<i>ubValue,:</i>	an upper bound associated with this objective

6.211.4 Member Data Documentation

6.211.4.1 `int OtherObjectiveOption::numberOfObj`

number of <obj> children

Definition at line 2579 of file OSOption.h.

6.211.4.2 `int OtherObjectiveOption::numberOfEnumerations`

number of <enumeration> child elements

Definition at line 2582 of file OSOption.h.

6.211.4.3 `std::string OtherObjectiveOption::name`

name of the option

Definition at line 2585 of file OSOption.h.

6.211.4.4 `std::string OtherObjectiveOption::value`

value of the option

Definition at line 2588 of file OSOption.h.

6.211.4.5 `std::string OtherObjectiveOption::solver`

name of the solver to which this option applies

Definition at line 2591 of file OSOption.h.

6.211.4.6 `std::string OtherObjectiveOption::category`

name of the category into which this option falls

Definition at line 2594 of file OSOption.h.

6.211.4.7 `std::string OtherObjectiveOption::type`

type of the option value (integer, double, boolean, string)

Definition at line 2597 of file OSOption.h.

6.211.4.8 `std::string OtherObjectiveOption::description`

description of the option

Definition at line 2600 of file OSOption.h.

6.211.4.9 `OtherObjOption** OtherObjectiveOption::obj`

array of option values

Definition at line 2603 of file OSOption.h.

6.211.4.10 `std::string OtherObjectiveOption::objType`

type of the values in the obj array

Definition at line 2606 of file OSOption.h.

6.211.4.11 `OtherOptionEnumeration** OtherObjectiveOption::enumeration`

Definition at line 2612 of file OSOption.h.

6.211.4.12 `std::string OtherObjectiveOption::enumType`

type of the values in the enumeration array

Definition at line 2615 of file OSOption.h.

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

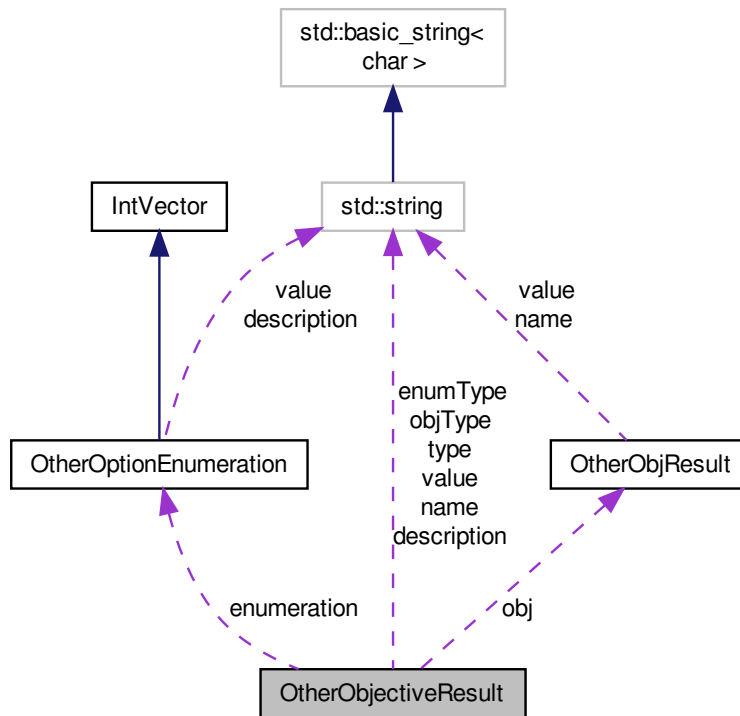
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.212 OtherObjectiveResult Class Reference

The [OtherObjectiveResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherObjectiveResult:



Public Member Functions

- `OtherObjectiveResult ()`
Default constructor.
- `~OtherObjectiveResult ()`
Class destructor.
- `bool isEqual (OtherObjectiveResult *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfObj`
the number of objectives which have values for this particular type of result
- `int numberOfEnumerations`
the number of distinct values for this particular type of result
- `std::string name`

- the name of the result the user is defining*
- `std::string` [value](#)
- this element allows a specific value associated with this particular type of result*
- `std::string` [type](#)
- type of the result value (integer, double, boolean, string)*
- `std::string` [description](#)
- a brief description of the type of result*
- [OtherObjResult](#) ****** [obj](#)
- `std::string` [objType](#)
- type of the values in the obj array*
- [OtherOptionEnumeration](#) ****** [enumeration](#)
- `std::string` [enumType](#)
- type of the values in the enumeration array*

6.212.1 Detailed Description

The [OtherObjectiveResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with objective functions

Definition at line 1435 of file OSResult.h.

6.212.2 Constructor & Destructor Documentation

6.212.2.1 OtherObjectiveResult::OtherObjectiveResult ()

Default constructor.

6.212.2.2 OtherObjectiveResult::~~OtherObjectiveResult ()

Class destructor.

6.212.3 Member Function Documentation

6.212.3.1 bool OtherObjectiveResult::IsEqual (OtherObjectiveResult * *that*)

A function to check for the equality of two objects.

6.212.3.2 `bool OtherObjectiveResult::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.212.4 Member Data Documentation**6.212.4.1** `int OtherObjectiveResult::numberOfObj`

the number of objectives which have values for this particular type of result

Definition at line 1442 of file OSResult.h.

6.212.4.2 `int OtherObjectiveResult::numberOfEnumerations`

the number of distinct values for this particular type of result

Definition at line 1447 of file OSResult.h.

6.212.4.3 `std::string OtherObjectiveResult::name`

the name of the result the user is defining

Definition at line 1450 of file OSResult.h.

6.212.4.4 `std::string OtherObjectiveResult::value`

this element allows a specific value associated with this particular type of result

Definition at line 1455 of file OSResult.h.

6.212.4.5 `std::string OtherObjectiveResult::type`

type of the result value (integer, double, boolean, string)

Definition at line 1458 of file OSResult.h.

6.212.4.6 `std::string OtherObjectiveResult::description`

a brief description of the type of result

Definition at line 1461 of file OSResult.h.

6.212.4.7 `OtherObjResult** OtherObjectiveResult::obj`

Definition at line 1467 of file OSResult.h.

6.212.4.8 `std::string OtherObjectiveResult::objType`

type of the values in the obj array

Definition at line 1470 of file OSResult.h.

6.212.4.9 OtherOptionEnumeration** OtherObjectiveResult::enumeration

Definition at line 1476 of file OSResult.h.

6.212.4.10 std::string OtherObjectiveResult::enumType

type of the values in the enumeration array

Definition at line 1479 of file OSResult.h.

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

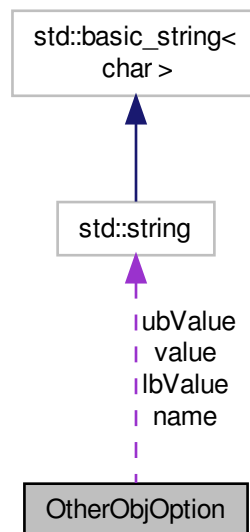
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.213 OtherObjOption Class Reference

the [OtherObjOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherObjOption:



Public Member Functions

- [OtherObjOption](#) ()
Default constructor.
- [~OtherObjOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([OtherObjOption](#) *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant)

A function to make a random instance of this class.

- bool [deepCopyFrom](#) ([OtherObjOption](#) *that)

A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
variable index
- std::string [name](#)
optional variable name
- std::string [value](#)
value of the option
- std::string [lbValue](#)
lower bound on the value
- std::string [ubValue](#)
lower bound on the value

6.213.1 Detailed Description

the [OtherObjOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2509 of file OSOption.h.

6.213.2 Constructor & Destructor Documentation

6.213.2.1 OtherObjOption::OtherObjOption ()

Default constructor.

6.213.2.2 OtherObjOption::~~OtherObjOption ()

Class destructor.

6.213.3 Member Function Documentation

6.213.3.1 `bool OtherObjOption::isEqual (OtherObjOption * that)`

A function to check for the equality of two objects.

6.213.3.2 `bool OtherObjOption::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.213.3.3 `bool OtherObjOption::deepCopyFrom (OtherObjOption * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.213.4 Member Data Documentation

6.213.4.1 `int OtherObjOption::idx`

variable index

Definition at line 2514 of file OSOption.h.

6.213.4.2 `std::string OtherObjOption::name`

optional variable name

Definition at line 2517 of file OSOption.h.

6.213.4.3 `std::string OtherObjOption::value`

value of the option

Definition at line 2520 of file OSOption.h.

6.213.4.4 `std::string OtherObjOption::lbValue`

lower bound on the value

Definition at line 2524 of file OSOption.h.

6.213.4.5 `std::string OtherObjOption::ubValue`

lower bound on the value

Definition at line 2527 of file OSOption.h.

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

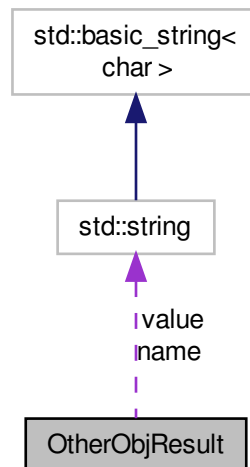
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.214 OtherObjResult Class Reference

The [OtherObjResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherObjResult:



Public Member Functions

- [OtherObjResult](#) ()
Default constructor.
- [~OtherObjResult](#) ()
Class destructor.
- `bool` [IsEqual](#) ([OtherObjResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [idx](#)
idx is the index on a objective function

- `std::string` [name](#)
optional name
- `std::string` [value](#)
value is a value associated with an objective function indexed by `idx`

6.214.1 Detailed Description

The [OtherObjResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides general result information for objective functions.

Definition at line 1379 of file OSResult.h.

6.214.2 Constructor & Destructor Documentation

6.214.2.1 OtherObjResult::OtherObjResult ()

Default constructor.

6.214.2.2 OtherObjResult::~OtherObjResult ()

Class destructor.

6.214.3 Member Function Documentation

6.214.3.1 bool OtherObjResult::isEqual (OtherObjResult * *that*)

A function to check for the equality of two objects.

6.214.3.2 bool OtherObjResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.214.4 Member Data Documentation

6.214.4.1 int OtherObjResult::idx

idx is the index on a objective function

Definition at line 1384 of file OSResult.h.

6.214.4.2 std::string OtherObjResult::name

optional name

Definition at line 1387 of file OSResult.h.

6.214.4.3 std::string OtherObjResult::value

value is a value associated with an objective function indexed by idx

Definition at line 1392 of file OSResult.h.

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

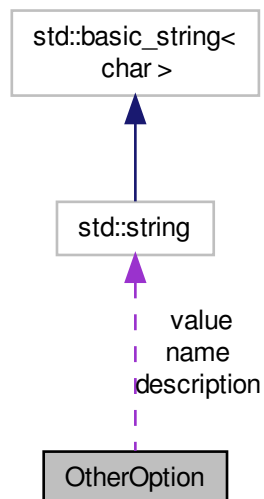
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.215 OtherOption Class Reference

the [OtherOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherOption:



Public Member Functions

- [OtherOption](#) ()
Default constructor.
- [~OtherOption](#) ()
Class destructor.
- bool [isEqual](#) ([OtherOption](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([OtherOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string [name](#)
the name of the option
- std::string [value](#)
the value of the option
- std::string [description](#)
the description of the option

6.215.1 Detailed Description

the [OtherOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to the [OtherOption](#) element in the OSoL schema.

Definition at line 152 of file OSOption.h.

6.215.2 Constructor & Destructor Documentation

6.215.2.1 OtherOption::OtherOption ()

Default constructor.

6.215.2.2 OtherOption::~~OtherOption ()

Class destructor.

6.215.3 Member Function Documentation

6.215.3.1 bool OtherOption::isEqual (OtherOption * *that*)

A function to check for the equality of two objects.

6.215.3.2 bool OtherOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.215.3.3 bool OtherOption::deepCopyFrom (OtherOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.215.4 Member Data Documentation

6.215.4.1 std::string OtherOption::name

the name of the option

Definition at line 157 of file OSOption.h.

6.215.4.2 std::string OtherOption::value

the value of the option

Definition at line 160 of file OSOption.h.

6.215.4.3 std::string OtherOption::description

the description of the option

Definition at line 163 of file OSOption.h.

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

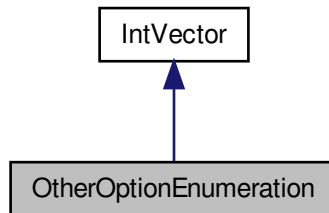
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.216 OtherOptionEnumeration Class Reference

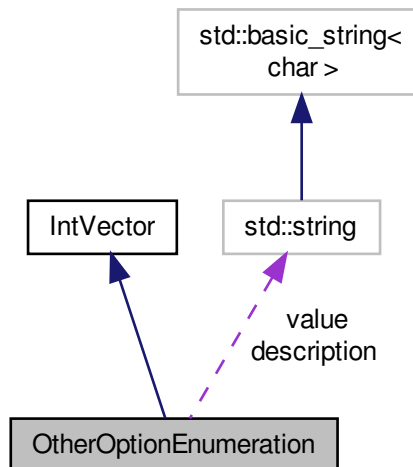
brief an integer vector data structure used in [OSOption](#) and [OSResult](#)

```
#include <OSGeneral.h>
```

Inheritance diagram for OtherOptionEnumeration:



Collaboration diagram for OtherOptionEnumeration:



Public Member Functions

- [OtherOptionEnumeration](#) ()
- [~OtherOptionEnumeration](#) ()
- [OtherOptionEnumeration](#) (int n)

alternate constructor

- `bool IsEqual (OtherOptionEnumeration *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant, int iMin, int iMax)`
A function to make a random instance of this class.
- `bool deepCopyFrom (OtherOptionEnumeration *that)`
A function to make a deep copy of an instance of this class.
- `bool setOtherOptionEnumeration (std::string value, std::string description, int *i, int ni)`
Set the indices for a particular level in an enumeration.
- `std::string getValue ()`
Get the value for a particular level in an enumeration.
- `std::string getDescription ()`
Get the description for a particular level in an enumeration.

Public Attributes

- `std::string value`
- `std::string description`

6.216.1 Detailed Description

brief an integer vector data structure used in [OSOption](#) and [OSResult](#)

This class extends [IntVector](#) by adding two string-valued elements, value and description

Definition at line 549 of file `OSGeneral.h`.

6.216.2 Constructor & Destructor Documentation

6.216.2.1 `OtherOptionEnumeration::OtherOptionEnumeration ()`

6.216.2.2 `OtherOptionEnumeration::~~OtherOptionEnumeration ()`

6.216.2.3 `OtherOptionEnumeration::OtherOptionEnumeration (int n)`

alternate constructor

6.216.3 Member Function Documentation

6.216.3.1 `bool OtherOptionEnumeration::IsEqual (OtherOptionEnumeration * that)`

A function to check for the equality of two objects.

6.216.3.2 `bool OtherOptionEnumeration::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest value (inclusive) that an entry in this vector can take
<i>iMax</i> ,:	greatest value (inclusive) that an entry in this vector can take

Reimplemented from [IntVector](#).

6.216.3.3 bool OtherOptionEnumeration::deepCopyFrom (OtherOptionEnumeration * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.216.3.4 bool OtherOptionEnumeration::setOtherOptionEnumeration (std::string value, std::string description, int * i, int ni)

Set the indices for a particular level in an enumeration.

Parameters

<i>value</i>	represents the value of this enumeration member
<i>description</i>	holds additional information about this value
<i>i</i>	contains the array of indices
<i>ni</i>	contains the number of elements in i

6.216.3.5 std::string OtherOptionEnumeration::getValue ()

Get the value for a particular level in an enumeration.

6.216.3.6 std::string OtherOptionEnumeration::getDescription ()

Get the description for a particular level in an enumeration.

6.216.4 Member Data Documentation

6.216.4.1 std::string OtherOptionEnumeration::value

Definition at line 552 of file OSGeneral.h.

6.216.4.2 std::string OtherOptionEnumeration::description

Definition at line 553 of file OSGeneral.h.

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

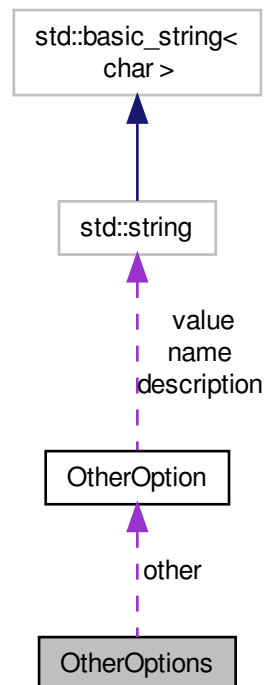
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSGeneral.h](#)

6.217 OtherOptions Class Reference

the [OtherOptions](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherOptions:



Public Member Functions

- [OtherOptions](#) ()
Default constructor.
- [~OtherOptions](#) ()
Class destructor.
- bool [IsEqual](#) ([OtherOptions](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([OtherOptions](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setOther](#) (int numberOfOptions, [OtherOption](#) **other)
A function to set an array of <other> elements.

- bool [addOther](#) (std::string name, std::string value, std::string description)
A function to add an <other> element.

Public Attributes

- int [numberOfOtherOptions](#)
the number of other options
- [OtherOption](#) ** [other](#)
the list of other options

6.217.1 Detailed Description

the [OtherOptions](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to the [OtherOptions](#) element in the OSoL schema.

Definition at line 211 of file OSOption.h.

6.217.2 Constructor & Destructor Documentation

6.217.2.1 OtherOptions::OtherOptions ()

Default constructor.

6.217.2.2 OtherOptions::~~OtherOptions ()

Class destructor.

6.217.3 Member Function Documentation

6.217.3.1 bool OtherOptions::isEqual (OtherOptions * that)

A function to check for the equality of two objects.

6.217.3.2 `bool OtherOptions::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.217.3.3 `bool OtherOptions::deepCopyFrom (OtherOptions * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.217.3.4 `bool OtherOptions::setOther (int numberOfOptions, OtherOption ** other)`

A function to set an array of <other> elements.

Parameters

<i>numberOfOptions</i> ,:	number of <other> elements to be set
<i>other</i> ,:	the array of <other> elements that are to be set

6.217.3.5 `bool OtherOptions::addOther (std::string name, std::string value, std::string description)`

A function to add an <other> element.

Parameters

<i>name</i> ,:	the name of the <other> element to be added (required)
<i>value</i> ,:	the value of the <other> element to be added (optional)
<i>description</i> ,:	a description of the <other> element (optional)

6.217.4 Member Data Documentation

6.217.4.1 `int OtherOptions::numberOfOtherOptions`

the number of other options

Definition at line 216 of file OSOption.h.

6.217.4.2 `OtherOption** OtherOptions::other`

the list of other options

Definition at line 219 of file OSOption.h.

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

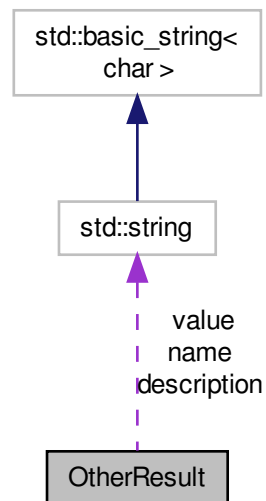
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.218 OtherResult Class Reference

The [OtherResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherResult:



Public Member Functions

- [OtherResult](#) ()
Default constructor.
- [~OtherResult](#) ()
Class destructor.
- `bool` [isEqual](#) ([OtherResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `std::string` [name](#)
the name of the other result

- `std::string value`
the value of the other result
- `std::string description`
the description of the other result

6.218.1 Detailed Description

The [OtherResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 10/09/2009

Since

OS 2.0

Remarks

A data structure class that corresponds to an xml element in the OSrL schema.

Definition at line 161 of file OSResult.h.

6.218.2 Constructor & Destructor Documentation

6.218.2.1 OtherResult::OtherResult ()

Default constructor.

6.218.2.2 OtherResult::~~OtherResult ()

Class destructor.

6.218.3 Member Function Documentation

6.218.3.1 bool OtherResult::isEqual (OtherResult * that)

A function to check for the equality of two objects.

6.218.3.2 bool OtherResult::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.218.4 Member Data Documentation

6.218.4.1 `std::string OtherResult::name`

the name of the other result

Definition at line 167 of file OSResult.h.

6.218.4.2 `std::string OtherResult::value`

the value of the other result

Definition at line 170 of file OSResult.h.

6.218.4.3 `std::string OtherResult::description`

the description of the other result

Definition at line 173 of file OSResult.h.

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

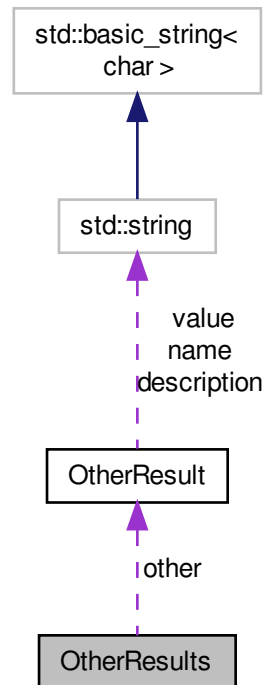
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.219 OtherResults Class Reference

The [OtherResults](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherResults:



Public Member Functions

- `OtherResults ()`
Default constructor.
- `~OtherResults ()`
Class destructor.
- `bool isEqual (OtherResults *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfOtherResults`
the number of other results
- `OtherResult ** other`
the array of other results

6.219.1 Detailed Description

The [OtherResults](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 10/09/2009

Since

OS 1.0

Remarks

A data structure class that corresponds to an xml element in the OSrL schema.

Definition at line 215 of file OSResult.h.

6.219.2 Constructor & Destructor Documentation

6.219.2.1 OtherResults::OtherResults ()

Default constructor.

6.219.2.2 OtherResults::~~OtherResults ()

Class destructor.

6.219.3 Member Function Documentation

6.219.3.1 bool OtherResults::IsEqual (OtherResults * *that*)

A function to check for the equality of two objects.

6.219.3.2 bool OtherResults::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.219.4 Member Data Documentation

6.219.4.1 int OtherResults::numberOfOtherResults

the number of other results

Definition at line 221 of file OSResult.h.

6.219.4.2 OtherResult** OtherResults::other

the array of other results

Definition at line 224 of file OSResult.h.

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

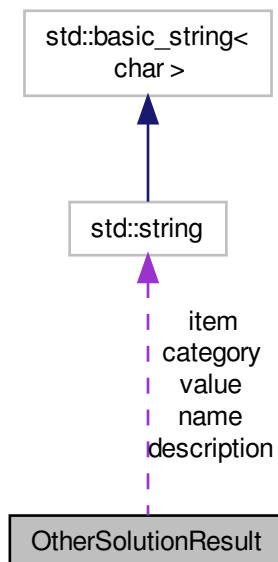
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.220 OtherSolutionResult Class Reference

The [OtherSolutionResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherSolutionResult:



Public Member Functions

- [OtherSolutionResult](#) ()
Default constructor.
- [~OtherSolutionResult](#) ()

Class destructor.

- bool `isEqual` (`OtherSolutionResult *that`)
A function to check for the equality of two objects.
- bool `setRandom` (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string `name`
the name of the result the user is defining
- std::string `value`
the value associated with the result the user is defining
- std::string `category`
this element allows a specific category to be associated with this particular type of result
- std::string `description`
a brief description of the type of result
- int `numberOfItems`
the number of items contained in this otherSolutionResult
- std::string * `item`
an array of items (string-valued)

6.220.1 Detailed Description

The `OtherSolutionResult` Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with the solution.

Definition at line 1905 of file OSResult.h.

6.220.2 Constructor & Destructor Documentation

6.220.2.1 OtherSolutionResult::OtherSolutionResult ()

Default constructor.

6.220.2.2 OtherSolutionResult::~~OtherSolutionResult ()

Class destructor.

6.220.3 Member Function Documentation

6.220.3.1 bool OtherSolutionResult::isEqual (OtherSolutionResult * *that*)

A function to check for the equality of two objects.

6.220.3.2 bool OtherSolutionResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.220.4 Member Data Documentation

6.220.4.1 std::string OtherSolutionResult::name

the name of the result the user is defining

Definition at line 1910 of file OSResult.h.

6.220.4.2 std::string OtherSolutionResult::value

the value associated with the result the user is defining

Definition at line 1913 of file OSResult.h.

6.220.4.3 std::string OtherSolutionResult::category

this element allows a specific category to be associated with this particular type of result

Definition at line 1918 of file OSResult.h.

6.220.4.4 std::string OtherSolutionResult::description

a brief description of the type of result

Definition at line 1921 of file OSResult.h.

6.220.4.5 int OtherSolutionResult::numberOfItems

the number of items contained in this otherSolutionResult

Definition at line 1925 of file OSResult.h.

6.220.4.6 std::string* OtherSolutionResult::item

an array of items (string-valued)

Definition at line 1929 of file OSResult.h.

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

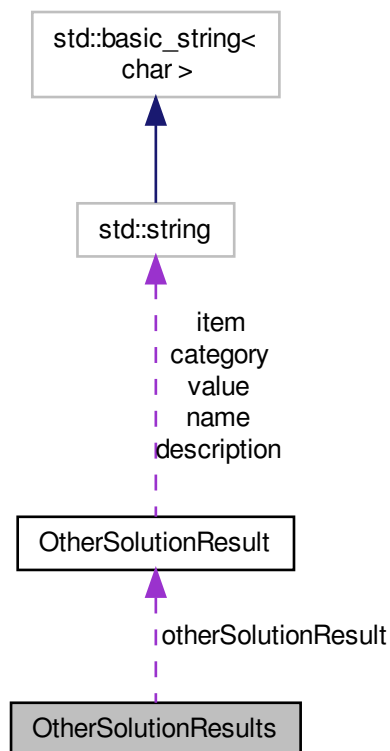
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.221 OtherSolutionResults Class Reference

The [OtherSolutionResults](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherSolutionResults:



Public Member Functions

- [OtherSolutionResults](#) ()
Default constructor.
- [~OtherSolutionResults](#) ()
Class destructor.
- bool [IsEqual](#) ([OtherSolutionResults](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [numberOfOtherSolutionResults](#)
the number of elements in the pointer of [OtherSolutionResult](#) objects
- [OtherSolutionResult](#) ** [otherSolutionResult](#)
otherSolutionResult is a pointer to an array of [OtherSolutionResult](#) objects

6.221.1 Detailed Description

The [OtherSolutionResults](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with the solution.

Definition at line 1974 of file OSResult.h.

6.221.2 Constructor & Destructor Documentation

6.221.2.1 OtherSolutionResults::OtherSolutionResults ()

Default constructor.

6.221.2.2 OtherSolutionResults::~~OtherSolutionResults ()

Class destructor.

6.221.3 Member Function Documentation

6.221.3.1 bool OtherSolutionResults::isEqual (OtherSolutionResults * that)

A function to check for the equality of two objects.

6.221.3.2 bool OtherSolutionResults::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.221.4 Member Data Documentation

6.221.4.1 int OtherSolutionResults::numberOfOtherSolutionResults

the number of elements in the pointer of [OtherSolutionResult](#) objects

Definition at line 1979 of file OSResult.h.

6.221.4.2 OtherSolutionResult** OtherSolutionResults::otherSolutionResult

otherSolutionResult is a pointer to an array of [OtherSolutionResult](#) objects

Definition at line 1984 of file OSResult.h.

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

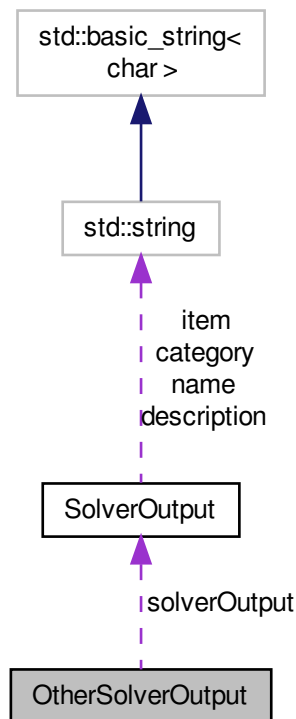
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h

6.222 OtherSolverOutput Class Reference

The [OtherSolverOutput](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherSolverOutput:



Public Member Functions

- [OtherSolverOutput](#) ()
Default constructor.
- [~OtherSolverOutput](#) ()
Class destructor.
- bool [isEqual](#) ([OtherSolverOutput](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [numberOfSolverOutputs](#)
the number of elements in the pointer of [SolverOutput](#) objects
- [SolverOutput](#) ** [solverOutput](#)
solverOutput is a pointer to an array of [SolverOutput](#) objects

6.222.1 Detailed Description

The [OtherSolverOutput](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with the solution.

Definition at line 2183 of file OSResult.h.

6.222.2 Constructor & Destructor Documentation

6.222.2.1 [OtherSolverOutput::OtherSolverOutput](#) ()

Default constructor.

6.222.2.2 [OtherSolverOutput::~~OtherSolverOutput](#) ()

Class destructor.

6.222.3 Member Function Documentation

6.222.3.1 `bool OtherSolverOutput::isEqual (OtherSolverOutput * that)`

A function to check for the equality of two objects.

6.222.3.2 `bool OtherSolverOutput::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.222.4 Member Data Documentation

6.222.4.1 `int OtherSolverOutput::numberOfSolverOutputs`

the number of elements in the pointer of [SolverOutput](#) objects

Definition at line 2189 of file OSResult.h.

6.222.4.2 `SolverOutput** OtherSolverOutput::solverOutput`

solverOutput is a pointer to an array of [SolverOutput](#) objects

Definition at line 2193 of file OSResult.h.

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

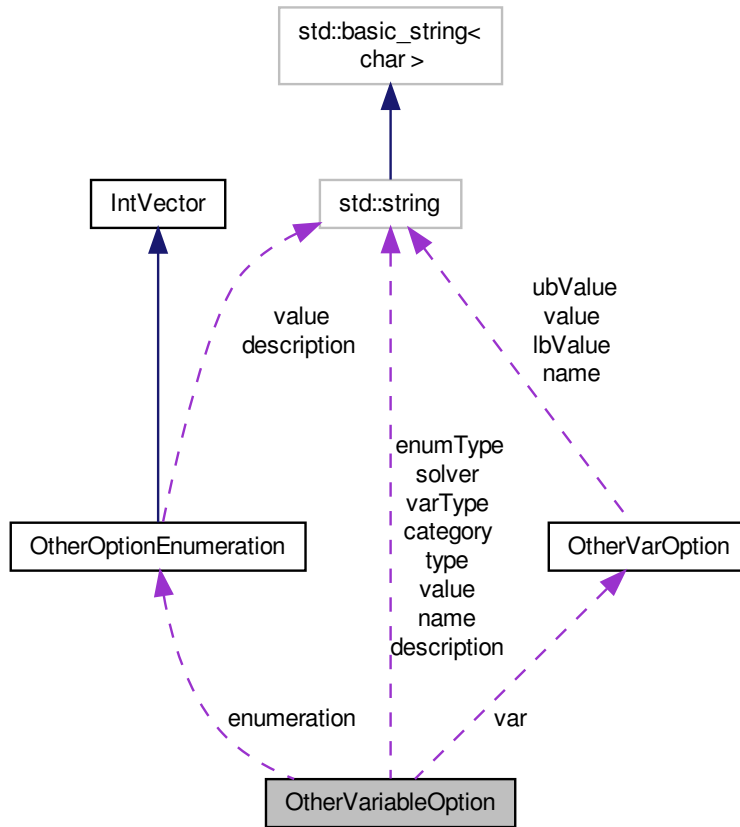
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.223 OtherVariableOption Class Reference

the [OtherVariableOption](#) class.

```
#include <OSOption.h>
```


Collaboration diagram for OtherVariableOption:



Public Member Functions

- [OtherVariableOption](#) ()
Default constructor.
- [~OtherVariableOption](#) ()
Class destructor.
- `bool` [IsEqual](#) ([OtherVariableOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([OtherVariableOption](#) *that)
A function to make a deep copy of an instance of this class.
- `bool` [setVar](#) (int numberOfVar, [OtherVarOption](#) **var)
A function to set an array of elements.
- `bool` [addVar](#) (int idx, `std::string` value, `std::string` lbValue, `std::string` ubValue)
A function to add a element.

Public Attributes

- int [numberOfVar](#)
number of child elements
- int [numberOfEnumerations](#)
number of <enumeration> child elements
- std::string [name](#)
name of the option
- std::string [value](#)
value of the option
- std::string [solver](#)
name of the solver to which this option applies
- std::string [category](#)
name of the category into which this option falls
- std::string [type](#)
type of the option value (integer, double, boolean, string)
- std::string [description](#)
description of the option
- [OtherVarOption](#) ** [var](#)
array of option values
- std::string [varType](#)
type of the values in the var array
- [OtherOptionEnumeration](#) ** [enumeration](#)
- std::string [enumType](#)
type of the values in the enumeration array

6.223.1 Detailed Description

the [OtherVariableOption](#) class.

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1989 of file OSOption.h.

6.223.2 Constructor & Destructor Documentation

6.223.2.1 OtherVariableOption::OtherVariableOption ()

Default constructor.

6.223.2.2 OtherVariableOption::~~OtherVariableOption ()

Class destructor.

6.223.3 Member Function Documentation

6.223.3.1 bool OtherVariableOption::isEqual (OtherVariableOption * *that*)

A function to check for the equality of two objects.

6.223.3.2 `bool OtherVariableOption::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.223.3.3 `bool OtherVariableOption::deepCopyFrom (OtherVariableOption * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.223.3.4 `bool OtherVariableOption::setVar (int numberOfVar, OtherVarOption ** var)`

A function to set an array of *elements*.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements</i> to be set
<i>var</i> ,:	the array of <i>elements</i> that are to be set

6.223.3.5 `bool OtherVariableOption::addVar (int idx, std::string value, std::string lbValue, std::string ubValue)`

A function to add a *element*.

Parameters

<i>idx</i> ,:	the index of the variable
<i>value</i> ,:	the value associated with this variable
<i>lbValue</i> ,:	a lower bound associated with this variable
<i>ubValue</i> ,:	an upper bound associated with this variable

6.223.4 Member Data Documentation

6.223.4.1 `int OtherVariableOption::numberOfVar`

number of *child elements*

Definition at line 1994 of file OSOption.h.

6.223.4.2 `int OtherVariableOption::numberOfEnumerations`

number of <enumeration> child elements

Definition at line 1997 of file OSOption.h.

6.223.4.3 std::string OtherVariableOption::name

name of the option

Definition at line 2000 of file OOption.h.

6.223.4.4 std::string OtherVariableOption::value

value of the option

Definition at line 2003 of file OOption.h.

6.223.4.5 std::string OtherVariableOption::solver

name of the solver to which this option applies

Definition at line 2006 of file OOption.h.

6.223.4.6 std::string OtherVariableOption::category

name of the category into which this option falls

Definition at line 2009 of file OOption.h.

6.223.4.7 std::string OtherVariableOption::type

type of the option value (integer, double, boolean, string)

Definition at line 2012 of file OOption.h.

6.223.4.8 std::string OtherVariableOption::description

description of the option

Definition at line 2015 of file OOption.h.

6.223.4.9 OtherVarOption OtherVariableOption::var**

array of option values

Definition at line 2018 of file OOption.h.

6.223.4.10 std::string OtherVariableOption::varType

type of the values in the var array

Definition at line 2021 of file OOption.h.

6.223.4.11 OtherOptionEnumeration OtherVariableOption::enumeration**

Definition at line 2027 of file OOption.h.

6.223.4.12 std::string OtherVariableOption::enumType

type of the values in the enumeration array

Definition at line 2030 of file OOption.h.

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

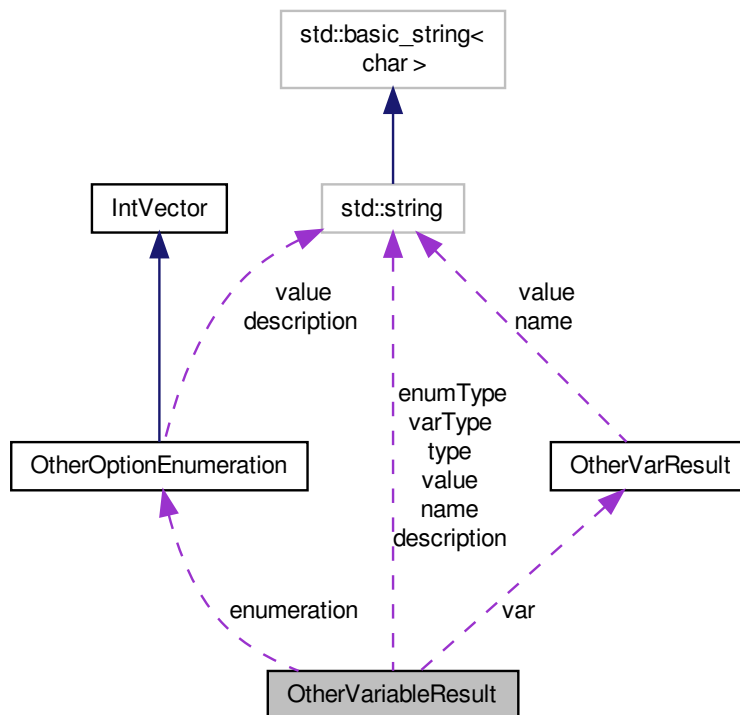
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OOption.h](#)

6.224 OtherVariableResult Class Reference

The [OtherVariableResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherVariableResult:



Public Member Functions

- [OtherVariableResult](#) ()
Default constructor.
- [~OtherVariableResult](#) ()
Class destructor.
- `bool` [isEqual](#) ([OtherVariableResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [numberOfVar](#)

- the number of variables which have values for this particular type of result*
- int [numberOfEnumerations](#)
 - the number of distinct values for this particular type of result*
- std::string [name](#)
 - the name of the result the user is defining*
- std::string [value](#)
 - this element allows a specific value associated with this particular type of result*
- std::string [type](#)
 - type of the result value (integer, double, boolean, string)*
- std::string [description](#)
 - a brief description of the type of result*
- [OtherVarResult](#) ** [var](#)
- std::string [varType](#)
 - type of the values in the var array*
- [OtherOptionEnumeration](#) ** [enumeration](#)
- std::string [enumType](#)
 - type of the values in the enumeration array*

6.224.1 Detailed Description

The [OtherVariableResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with variables

Definition at line 1122 of file OSResult.h.

6.224.2 Constructor & Destructor Documentation

6.224.2.1 OtherVariableResult::OtherVariableResult ()

Default constructor.

6.224.2.2 OtherVariableResult::~OtherVariableResult ()

Class destructor.

6.224.3 Member Function Documentation

6.224.3.1 `bool OtherVariableResult::isEqual (OtherVariableResult * that)`

A function to check for the equality of two objects.

6.224.3.2 `bool OtherVariableResult::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.224.4 Member Data Documentation

6.224.4.1 `int OtherVariableResult::numberOfVar`

the number of variables which have values for this particular type of result

Definition at line 1129 of file OSResult.h.

6.224.4.2 `int OtherVariableResult::numberOfEnumerations`

the number of distinct values for this particular type of result

Definition at line 1134 of file OSResult.h.

6.224.4.3 `std::string OtherVariableResult::name`

the name of the result the user is defining

Definition at line 1137 of file OSResult.h.

6.224.4.4 `std::string OtherVariableResult::value`

this element allows a specific value associated with this particular type of result

Definition at line 1142 of file OSResult.h.

6.224.4.5 `std::string OtherVariableResult::type`

type of the result value (integer, double, boolean, string)

Definition at line 1145 of file OSResult.h.

6.224.4.6 `std::string OtherVariableResult::description`

a brief description of the type of result

Definition at line 1148 of file OSResult.h.

6.224.4.7 `OtherVarResult** OtherVariableResult::var`

Definition at line 1154 of file OSResult.h.

6.224.4.8 std::string OtherVariableResult::varType

type of the values in the var array

Definition at line 1157 of file OSResult.h.

6.224.4.9 OtherOptionEnumeration** OtherVariableResult::enumeration

Definition at line 1163 of file OSResult.h.

6.224.4.10 std::string OtherVariableResult::enumType

type of the values in the enumeration array

Definition at line 1166 of file OSResult.h.

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

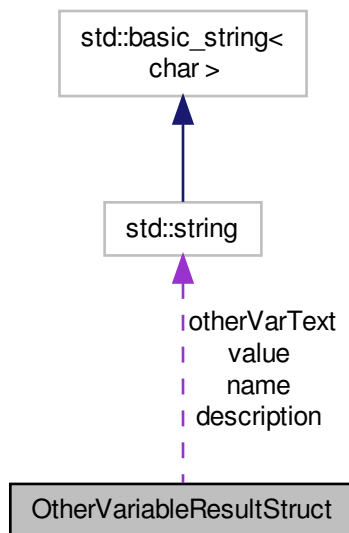
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.225 OtherVariableResultStruct Struct Reference

A structure to information about an [OtherVariableResult](#) element.

```
#include <OSrLParserData.h>
```

Collaboration diagram for OtherVariableResultStruct:



Public Attributes

- `std::string` [name](#)

- name holds the text of the name attribute of the [OtherVariableResult](#) element*
- `std::string` [description](#)
description holds the text of the description attribute of the [OtherVariableResult](#) element
- `std::string` [value](#)
value holds the text of the value attribute of the [OtherVariableResult](#) element
- `int` [numberOfVar](#)
numberOfVar holds the number of variables in the array of the [OtherVariableResult](#) element
- `std::string *` [otherVarText](#)
otherVarText is a pointer to an array with number of elements equal to the number of variables.
- `int *` [otherVarIndex](#)
otherVarIndex is a pointer to an array with number of elements equal to the number of variables.

6.225.1 Detailed Description

A structure to information about an [OtherVariableResult](#) element.

Definition at line 29 of file OSrLParserData.h.

6.225.2 Member Data Documentation

6.225.2.1 `std::string OtherVariableResultStruct::name`

`name` holds the text of the name attribute of the [OtherVariableResult](#) element

Definition at line 34 of file OSrLParserData.h.

6.225.2.2 `std::string OtherVariableResultStruct::description`

`description` holds the text of the description attribute of the [OtherVariableResult](#) element

Definition at line 39 of file OSrLParserData.h.

6.225.2.3 `std::string OtherVariableResultStruct::value`

`value` holds the text of the value attribute of the [OtherVariableResult](#) element

Definition at line 44 of file OSrLParserData.h.

6.225.2.4 `int OtherVariableResultStruct::numberOfVar`

`numberOfVar` holds the number of variables in the array of the [OtherVariableResult](#) element

Definition at line 49 of file OSrLParserData.h.

6.225.2.5 `std::string* OtherVariableResultStruct::otherVarText`

`otherVarText` is a pointer to an array with number of elements equal to the number of variables.

Each element of the array is the value of the variable corresponding to the [OtherVariableResult](#), e.g. a variable name or variable reduced cost, etc.

Definition at line 57 of file OSrLParserData.h.

6.225.2.6 `int* OtherVariableResultStruct::otherVarIndex`

`otherVarIndex` is a pointer to an array with number of elements equal to the number of variables.

Each element of the array is the index of the variable corresponding to the [OtherVariableResult](#), e.g. a variable name or variable reduced cost, etc.

Definition at line 66 of file OSrLParserData.h.

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

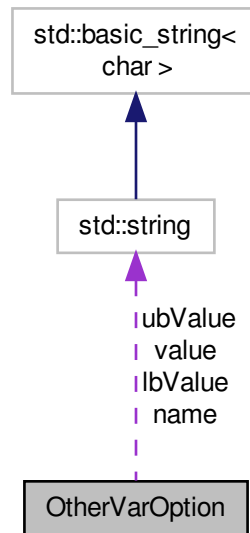
- [/home/ted/COIN/trunk/OS/src/OSParsers/OSrLParserData.h](#)

6.226 OtherVarOption Class Reference

the [OtherVarOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for OtherVarOption:



Public Member Functions

- [OtherVarOption](#) ()
Default constructor.
- [~OtherVarOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([OtherVarOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([OtherVarOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [idx](#)
variable index
- std::string [name](#)
optional variable name
- std::string [value](#)
value of the option
- std::string [lbValue](#)
lower bound on the value
- std::string [ubValue](#)
lower bound on the value

6.226.1 Detailed Description

the [OtherVarOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1927 of file OSOption.h.

6.226.2 Constructor & Destructor Documentation

6.226.2.1 OtherVarOption::OtherVarOption ()

Default constructor.

6.226.2.2 OtherVarOption::~~OtherVarOption ()

Class destructor.

6.226.3 Member Function Documentation

6.226.3.1 bool OtherVarOption::isEqual (OtherVarOption * *that*)

A function to check for the equality of two objects.

6.226.3.2 `bool OtherVarOption::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.226.3.3 `bool OtherVarOption::deepCopyFrom (OtherVarOption * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.226.4 Member Data Documentation

6.226.4.1 `int OtherVarOption::idx`

variable index

Definition at line 1932 of file OSOption.h.

6.226.4.2 `std::string OtherVarOption::name`

optional variable name

Definition at line 1935 of file OSOption.h.

6.226.4.3 `std::string OtherVarOption::value`

value of the option

Definition at line 1938 of file OSOption.h.

6.226.4.4 `std::string OtherVarOption::lbValue`

lower bound on the value

Definition at line 1941 of file OSOption.h.

6.226.4.5 `std::string OtherVarOption::ubValue`

lower bound on the value

Definition at line 1944 of file OSOption.h.

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

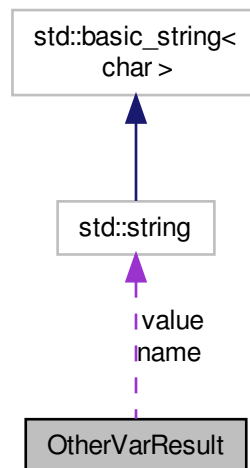
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.227 OtherVarResult Class Reference

[OtherVarResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for OtherVarResult:



Public Member Functions

- [OtherVarResult](#) ()
Default constructor.
- [~OtherVarResult](#) ()
Class destructor.
- `bool` [isEqual](#) ([OtherVarResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [idx](#)
the index of a variable in the solution
- `std::string` [name](#)
optional name
- `std::string` [value](#)
value holds a general value associated with a variable, for example, rather than the value of a variable we may wish to store the variable name associated with the variable with index idx, hence we want a string here and not a double

6.227.1 Detailed Description

[OtherVarResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class used to provide solution information associated with the variables.

Definition at line 1063 of file OSResult.h.

6.227.2 Constructor & Destructor Documentation

6.227.2.1 OtherVarResult::OtherVarResult ()

Default constructor.

6.227.2.2 OtherVarResult::~~OtherVarResult ()

Class destructor.

6.227.3 Member Function Documentation

6.227.3.1 bool OtherVarResult::isEqual (OtherVarResult * *that*)

A function to check for the equality of two objects.

6.227.3.2 bool OtherVarResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.227.4 Member Data Documentation

6.227.4.1 int OtherVarResult::idx

the index of a variable in the solution

Definition at line 1068 of file OSResult.h.

6.227.4.2 std::string OtherVarResult::name

optional name

Definition at line 1071 of file OSResult.h.

6.227.4.3 std::string OtherVarResult::value

value holds a general value associated with a variable, for example, rather than the value of a variable we may wish to store the variable name associated with the variable with index idx, hence we want a string here and not a double

Definition at line 1079 of file OSResult.h.

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

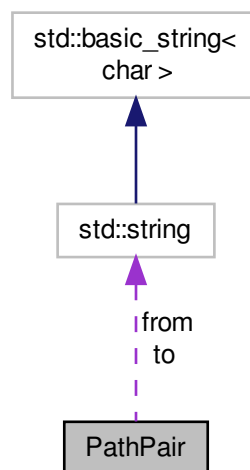
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.228 PathPair Class Reference

the [PathPair](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for PathPair:



Public Member Functions

- [PathPair](#) ()
Default constructor.
- [~PathPair](#) ()
Class destructor.
- bool [isEqual](#) ([PathPair](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([PathPair](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string [from](#)
the file or directory to move/copy from
- std::string [to](#)
the file or directory to move/copy to
- bool [makeCopy](#)
record whether a copy is to be made

6.228.1 Detailed Description

the [PathPair](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 851 of file OSOption.h.

6.228.2 Constructor & Destructor Documentation

6.228.2.1 [PathPair::PathPair](#) ()

Default constructor.

6.228.2.2 PathPair::~~PathPair ()

Class destructor.

6.228.3 Member Function Documentation

6.228.3.1 bool PathPair::isEqual (PathPair * *that*)

A function to check for the equality of two objects.

6.228.3.2 bool PathPair::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.228.3.3 bool PathPair::deepCopyFrom (PathPair * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.228.4 Member Data Documentation

6.228.4.1 std::string PathPair::from

the file or directory to move/copy from

Definition at line 856 of file OSOption.h.

6.228.4.2 std::string PathPair::to

the file or directory to move/copy to

Definition at line 859 of file OSOption.h.

6.228.4.3 bool PathPair::makeCopy

record whether a copy is to be made

Definition at line 862 of file OSOption.h.

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

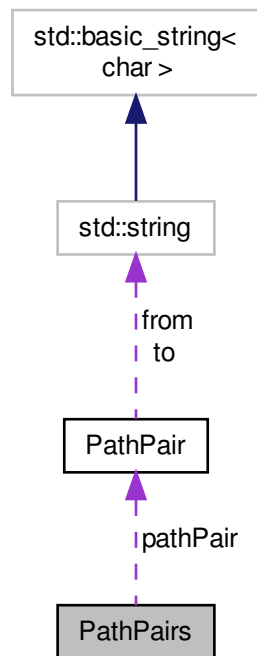
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.229 PathPairs Class Reference

the [PathPairs](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for PathPairs:



Public Member Functions

- [PathPairs](#) ()
Default constructor.
- [~PathPairs](#) ()
Class destructor.
- bool [isEqual](#) ([PathPairs](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([PathPairs](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setPathPair](#) (int numberOfPathPairs, [PathPair](#) **pathPair)
A function to set an array of <pathPair> elements.
- bool [setPathPair](#) (std::string *from, std::string *to, bool *makeCopy, int numberOfPathPairs)

Alternate signature for this function.

- bool [addPathPair](#) (std::string fromPath, std::string toPath, bool makeCopy)

A function to add a <pathPair> element.

Public Attributes

- int [numberOfPathPairs](#)

the number of <path> children

- [PathPair](#) ** [pathPair](#)

the list of directory and file paths

6.229.1 Detailed Description

the [PathPairs](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 910 of file OSOption.h.

6.229.2 Constructor & Destructor Documentation

6.229.2.1 [PathPairs::PathPairs](#) ()

Default constructor.

6.229.2.2 [PathPairs::~~PathPairs](#) ()

Class destructor.

6.229.3 Member Function Documentation

6.229.3.1 [bool PathPairs::isEqual](#) ([PathPairs](#) * *that*)

A function to check for the equality of two objects.

6.229.3.2 bool PathPairs::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.229.3.3 bool PathPairs::deepCopyFrom (PathPairs * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.229.3.4 bool PathPairs::setPathPair (int *numberOfPathPairs*, PathPair ** *pathPair*)

A function to set an array of <pathPair> elements.

Parameters

<i>numberOfPathPairs</i> ,:	number of <pathPair> elements to be set
<i>path</i> ,:	the array of <pathPair> elements that are to be set

6.229.3.5 bool PathPairs::setPathPair (std::string * *from*, std::string * *to*, bool * *makeCopy*, int *numberOfPathPairs*)

Alternate signature for this function.

Parameters

<i>from</i> ,:	array containing a list of objects to be moved
<i>to</i> ,:	array containing a list of destinations
<i>makeCopy</i> ,:	records whether each object is to be moved or copied
<i>numberOfPathPairs</i> ,:	number of <pathPair> elements to be set

6.229.3.6 bool PathPairs::addPathPair (std::string *fromPath*, std::string *toPath*, bool *makeCopy*)

A function to add a <pathPair> element.

Parameters

<i>fromPath</i> ,:	the path from which to copy or move
<i>toPath</i> ,:	the path to which to copy or move
<i>makecopy</i> ,:	tracks whether a copy is to be made

6.229.4 Member Data Documentation

6.229.4.1 int PathPairs::numberOfPathPairs

the number of <path> children

Definition at line 915 of file OSOption.h.

6.229.4.2 PathPair** PathPairs::pathPair

the list of directory and file paths

Definition at line 918 of file OSOption.h.

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

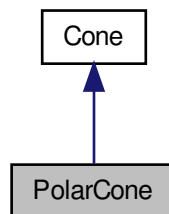
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.230 PolarCone Class Reference

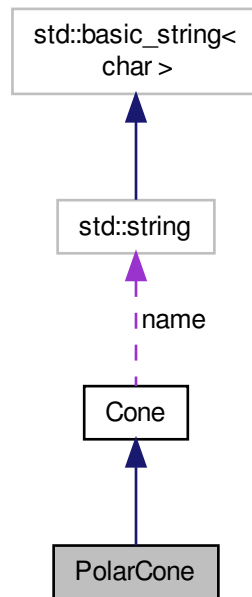
The in-memory representation of a polar cone.

```
#include <OSInstance.h>
```

Inheritance diagram for PolarCone:



Collaboration diagram for PolarCone:



Public Member Functions

- `PolarCone ()`
The `PolarCone` class constructor.
- `~PolarCone ()`
The `PolarCone` class destructor.
- virtual `std::string getConeName ()`
- `bool isEqual (PolarCone *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant, int iMin, int iMax)`
A function to make a random instance of this class.
- `bool deepCopyFrom (PolarCone *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `int numberOfRows`
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- `int numberOfColumns`
- `int numberOfOtherIndexes`
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in ENUM_CONE_TYPE)
- int [idx](#)
cones are referenced by an (automatically created) index
- int [referenceConeldx](#)
Polar cones use a reference to another, previously defined cone.

6.230.1 Detailed Description

The in-memory representation of a polar cone.

Definition at line 1466 of file OSInstance.h.

6.230.2 Constructor & Destructor Documentation

6.230.2.1 PolarCone::PolarCone ()

The [PolarCone](#) class constructor.

6.230.2.2 PolarCone::~~PolarCone ()

The [PolarCone](#) class destructor.

6.230.3 Member Function Documentation

6.230.3.1 virtual std::string PolarCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.230.3.2 bool PolarCone::isEqual (PolarCone * that)

A function to check for the equality of two objects.

6.230.3.3 bool PolarCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.230.3.4 bool PolarCone::deepCopyFrom (PolarCone * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.230.4 Member Data Documentation

6.230.4.1 int PolarCone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 1479 of file OSInstance.h.

6.230.4.2 int PolarCone::numberOfColumns

Definition at line 1480 of file OSInstance.h.

6.230.4.3 int PolarCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 1487 of file OSInstance.h.

6.230.4.4 int* PolarCone::otherIndexes

Definition at line 1488 of file OSInstance.h.

6.230.4.5 int PolarCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 1491 of file OSInstance.h.

6.230.4.6 int PolarCone::idx

cones are referenced by an (automatically created) index

Definition at line 1494 of file OSInstance.h.

6.230.4.7 int PolarCone::referenceConeldx

Polar cones use a reference to another, previously defined cone.

Definition at line 1497 of file OSInstance.h.

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

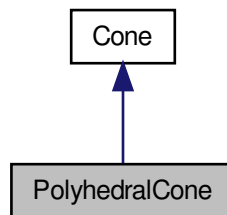
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.231 PolyhedralCone Class Reference

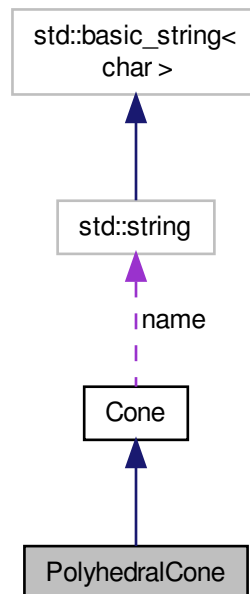
The in-memory representation of a polyhedral cone.

```
#include <OSInstance.h>
```

Inheritance diagram for PolyhedralCone:



Collaboration diagram for PolyhedralCone:



Public Member Functions

- [PolyhedralCone](#) ()
The [PolyhedralCone](#) class constructor.
- [~PolyhedralCone](#) ()
The [PolyhedralCone](#) class destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [PolyhedralCone](#) object in XML format.
- bool [IsEqual](#) ([PolyhedralCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([PolyhedralCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in `ENUM_CONE_TYPE`)
- int [idx](#)
cones are referenced by an (automatically created) index
- int [referenceMatrixIdx](#)
Polyhedral cones use a reference to a previously defined matrix for the extreme rays.

6.231.1 Detailed Description

The in-memory representation of a polyhedral cone.

Definition at line 786 of file `OSInstance.h`.

6.231.2 Constructor & Destructor Documentation

6.231.2.1 [PolyhedralCone::PolyhedralCone](#) ()

The [PolyhedralCone](#) class constructor.

6.231.2.2 [PolyhedralCone::~~PolyhedralCone](#) ()

The [PolyhedralCone](#) class destructor.

6.231.3 Member Function Documentation

6.231.3.1 virtual std::string PolyhedralCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.231.3.2 virtual std::string PolyhedralCone::getConeInXML () [virtual]

Write a [PolyhedralCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.231.3.3 bool PolyhedralCone::isEqual (PolyhedralCone * that)

A function to check for the equality of two objects.

6.231.3.4 bool PolyhedralCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.231.3.5 bool PolyhedralCone::deepCopyFrom (PolyhedralCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.231.4 Member Data Documentation

6.231.4.1 int PolyhedralCone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 799 of file OSInstance.h.

6.231.4.2 int PolyhedralCone::numberOfColumns

Definition at line 800 of file OSInstance.h.

6.231.4.3 int PolyhedralCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 807 of file OSInstance.h.

6.231.4.4 int* PolyhedralCone::otherIndexes

Definition at line 808 of file OSInstance.h.

6.231.4.5 int PolyhedralCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 811 of file OSInstance.h.

6.231.4.6 int PolyhedralCone::idx

cones are referenced by an (automatically created) index

Definition at line 814 of file OSInstance.h.

6.231.4.7 int PolyhedralCone::referenceMatrixIdx

Polyhedral cones use a reference to a previously defined matrix for the extreme rays.

Definition at line 817 of file OSInstance.h.

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

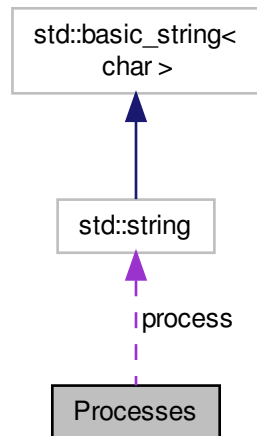
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.232 Processes Class Reference

the [Processes](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for Processes:



Public Member Functions

- [Processes](#) ()
Default constructor.
- [~Processes](#) ()
Class destructor.
- bool [isEqual](#) ([Processes](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([Processes](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setProcess](#) (int [numberOfProcesses](#), std::string *[process](#))
A function to set an array of <process> elements.
- bool [addProcess](#) (std::string [process](#))
A function to add a <process> element.

Public Attributes

- int [numberOfProcesses](#)
the number of <process> children
- std::string * [process](#)
the list of processes

6.232.1 Detailed Description

the [Processes](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 993 of file OSOption.h.

6.232.2 Constructor & Destructor Documentation

6.232.2.1 Processes::Processes ()

Default constructor.

6.232.2.2 Processes::~~Processes ()

Class destructor.

6.232.3 Member Function Documentation

6.232.3.1 bool Processes::isEqual (Processes * *that*)

A function to check for the equality of two objects.

6.232.3.2 bool Processes::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.232.3.3 bool Processes::deepCopyFrom (Processes * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.232.3.4 bool Processes::setProcess (int *numberOfProcesses*, std::string * *process*)

A function to set an array of <process> elements.

Parameters

<i>numberOfProcesses,:</i>	number of <process> elements to be set
<i>path,:</i>	the array of <process> elements that are to be set

6.232.3.5 bool Processes::addProcess (std::string *process*)

A function to add a <process> element.

Parameters

<i>process,:</i>	the ID of the process to be added
------------------	-----------------------------------

6.232.4 Member Data Documentation

6.232.4.1 int Processes::numberOfProcesses

the number of <process> children

Definition at line 998 of file OSOption.h.

6.232.4.2 std::string* Processes::process

the list of processes

Definition at line 1001 of file OSOption.h.

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

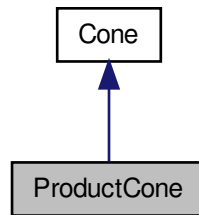
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h>

6.233 ProductCone Class Reference

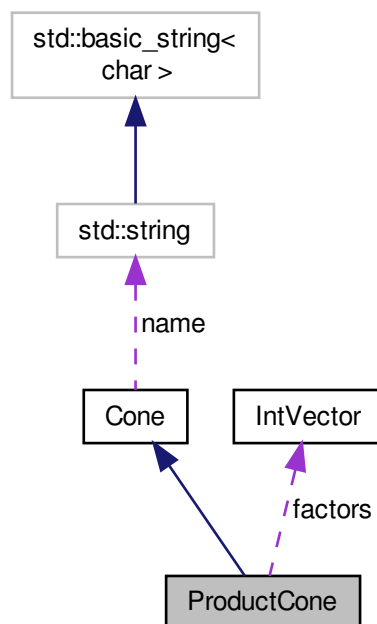
The in-memory representation of a product cone.

```
#include <OSInstance.h>
```

Inheritance diagram for ProductCone:



Collaboration diagram for ProductCone:



Public Member Functions

- [ProductCone](#) ()
The [ProductCone](#) class constructor.
- [~ProductCone](#) ()

The [ProductCone](#) class destructor.

- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()

Write a [ProductCone](#) object in XML format.

- bool [isEqual](#) ([ProductCone](#) *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

- bool [deepCopyFrom](#) ([ProductCone](#) *that)

A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

- int * [otherIndexes](#)
- int [coneType](#)

The type of the cone (one of the values in `ENUM_CONE_TYPE`)

- int [idx](#)

cones are referenced by an (automatically created) index

- [IntVector](#) * [factors](#)

the list of "factors" contributing to the product each factor contains a reference to a previously defined cone

6.233.1 Detailed Description

The in-memory representation of a product cone.

Definition at line 1249 of file `OSInstance.h`.

6.233.2 Constructor & Destructor Documentation

6.233.2.1 [ProductCone::ProductCone](#) ()

The [ProductCone](#) class constructor.

6.233.2.2 [ProductCone::~~ProductCone](#) ()

The [ProductCone](#) class destructor.

6.233.3 Member Function Documentation

6.233.3.1 virtual std::string [ProductCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.233.3.2 virtual std::string ProductCone::getConeInXML () [virtual]

Write a [ProductCone](#) object in XML format.

This is used by [OSILWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.233.3.3 bool ProductCone::isEqual (ProductCone * that)

A function to check for the equality of two objects.

6.233.3.4 bool ProductCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.233.3.5 bool ProductCone::deepCopyFrom (ProductCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.233.4 Member Data Documentation

6.233.4.1 int ProductCone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 1262 of file OSInstance.h.

6.233.4.2 int ProductCone::numberOfColumns

Definition at line 1263 of file OSInstance.h.

6.233.4.3 int ProductCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 1270 of file OSInstance.h.

6.233.4.4 int* ProductCone::otherIndexes

Definition at line 1271 of file OSInstance.h.

6.233.4.5 int ProductCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 1274 of file OSInstance.h.

6.233.4.6 int ProductCone::idx

cones are referenced by an (automatically created) index

Definition at line 1277 of file OSInstance.h.

6.233.4.7 IntVector* ProductCone::factors

the list of "factors" contributing to the product each factor contains a reference to a previously defined cone

Definition at line 1282 of file OSInstance.h.

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

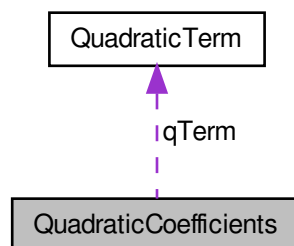
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.234 QuadraticCoefficients Class Reference

The in-memory representation of the <quadraticCoefficients> element.

```
#include <OSInstance.h>
```

Collaboration diagram for QuadraticCoefficients:



Public Member Functions

- [QuadraticCoefficients](#) ()
The [QuadraticCoefficients](#) class constructor.
- [~QuadraticCoefficients](#) ()
The [QuadraticCoefficients](#) class destructor.
- bool [IsEqual](#) ([QuadraticCoefficients](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfQuadraticTerms](#)
numberOfQuadraticTerms is the number of quadratic terms in the `<quadraticCoefficients>` element.
- [QuadraticTerm](#) ** [qTerm](#)
qTerm is a pointer to an array of [QuadraticTerm](#) object pointers

6.234.1 Detailed Description

The in-memory representation of the `<quadraticCoefficients>` element.

Definition at line 380 of file OSInstance.h.

6.234.2 Constructor & Destructor Documentation

6.234.2.1 QuadraticCoefficients::QuadraticCoefficients ()

The [QuadraticCoefficients](#) class constructor.

6.234.2.2 QuadraticCoefficients::~~QuadraticCoefficients ()

The [QuadraticCoefficients](#) class destructor.

6.234.3 Member Function Documentation

6.234.3.1 bool QuadraticCoefficients::IsEqual ([QuadraticCoefficients](#) * that)

A function to check for the equality of two objects.

6.234.4 Member Data Documentation

6.234.4.1 int QuadraticCoefficients::numberOfQuadraticTerms

numberOfQuadraticTerms is the number of quadratic terms in the `<quadraticCoefficients>` element.

Definition at line 393 of file OSInstance.h.

6.234.4.2 QuadraticTerm** QuadraticCoefficients::qTerm

qTerm is a pointer to an array of [QuadraticTerm](#) object pointers

Definition at line 397 of file OSInstance.h.

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

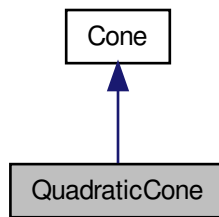
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.235 QuadraticCone Class Reference

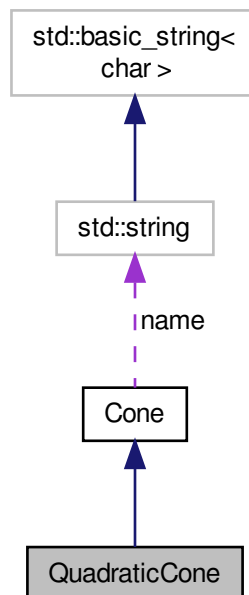
The in-memory representation of a quadratic cone.

```
#include <OSInstance.h>
```

Inheritance diagram for QuadraticCone:



Collaboration diagram for QuadraticCone:



Public Member Functions

- [QuadraticCone](#) ()
The [QuadraticCone](#) class constructor.
- [~QuadraticCone](#) ()
The [QuadraticCone](#) class destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [QuadraticCone](#) object in XML format.
- bool [IsEqual](#) ([QuadraticCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([QuadraticCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in `ENUM_CONE_TYPE`)
- int [idx](#)
cones are referenced by an (automatically created) index
- double [normScaleFactor](#)
quadratic cones normally are of the form $x_0 \geq x_1^2 + x_2^2 + \dots$
- int [distortionMatrixIdx](#)
- int [axisDirection](#)
*The index of the first component can be changed Since there are possibly many dimensions, the index is coded as $i_0 * n_1 * n_2 * \dots$*

6.235.1 Detailed Description

The in-memory representation of a quadratic cone.

Definition at line 860 of file OSInstance.h.

6.235.2 Constructor & Destructor Documentation

6.235.2.1 QuadraticCone::QuadraticCone ()

The [QuadraticCone](#) class constructor.

6.235.2.2 QuadraticCone::~~QuadraticCone ()

The [QuadraticCone](#) class destructor.

6.235.3 Member Function Documentation

6.235.3.1 virtual std::string QuadraticCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.235.3.2 virtual std::string QuadraticCone::getConeInXML () [virtual]

Write a [QuadraticCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.235.3.3 bool QuadraticCone::isEqual (QuadraticCone * that)

A function to check for the equality of two objects.

6.235.3.4 bool QuadraticCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.235.3.5 bool QuadraticCone::deepCopyFrom (QuadraticCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.235.4 Member Data Documentation

6.235.4.1 int QuadraticCone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 873 of file OSInstance.h.

6.235.4.2 int QuadraticCone::numberOfColumns

Definition at line 874 of file OSInstance.h.

6.235.4.3 int QuadraticCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 881 of file OSInstance.h.

6.235.4.4 int* QuadraticCone::otherIndexes

Definition at line 882 of file OSInstance.h.

6.235.4.5 int QuadraticCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 885 of file OSInstance.h.

6.235.4.6 int QuadraticCone::idx

cones are referenced by an (automatically created) index

Definition at line 888 of file OSInstance.h.

6.235.4.7 double QuadraticCone::normScaleFactor

quadratic cones normally are of the form $x_0 \geq x_1^2 + x_2^2 + \dots$

However, the appearance can be modified using a norm factor k and a distortion matrix M to the form $x_0 \geq p(x_1, x_2, \dots) M(x_1, x_2, \dots)'$: $k=1$, $M=-1$.

Definition at line 896 of file OSInstance.h.

6.235.4.8 int QuadraticCone::distortionMatrixIdx

Definition at line 897 of file OSInstance.h.

6.235.4.9 int QuadraticCone::axisDirection

The index of the first component can be changed Since there are possibly many dimensions, the index is coded as $i_0 * n_1 * n_2 * \dots$

- $i_1 * n_2 * n_3 * \dots + \dots + i_r$, where i_0, i_1 , etc are zero-based indexes for the different dimensions: $i_0 = 0, 1, \dots, n_0 - 1$, where n_0 is the number of rows, $i_1 = 0, 1, \dots, n_1 - 1$, where n_1 is the number of columns, and so on for higher dimensions (if any) (: 0)

Definition at line 908 of file OSInstance.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.236 QuadraticTerm Class Reference

The in-memory representation of the `<qTerm>` element.

```
#include <OSInstance.h>
```

Public Member Functions

- [QuadraticTerm](#) ()
The [QuadraticTerm](#) class constructor.
- [~QuadraticTerm](#) ()
The [QuadraticTerm](#) class destructor.
- `bool` [IsEqual](#) ([QuadraticTerm](#) *that)
A function to check for the equality of two objects.

Public Attributes

- `int` [idx](#)
idx is the index of the row in which the quadratic term appears
- `int` [idxOne](#)
idxOne is the index of the first variable in the quadratic term
- `int` [idxTwo](#)
idxTwo is the index of the second variable in the quadratic term
- `double` [coef](#)
coef is the coefficient of the quadratic term

6.236.1 Detailed Description

The in-memory representation of the `<qTerm>` element.

Remarks

quadratic terms can be stored efficiently by storing the index of each variable, the coefficient of the quadratic term, and the row in which it appears

Definition at line 340 of file `OSInstance.h`.

6.236.2 Constructor & Destructor Documentation

6.236.2.1 [QuadraticTerm::QuadraticTerm](#) ()

The [QuadraticTerm](#) class constructor.

6.236.2.2 [QuadraticTerm::~~QuadraticTerm](#) ()

The [QuadraticTerm](#) class destructor.

6.236.3 Member Function Documentation

6.236.3.1 bool QuadraticTerm::isEqual (QuadraticTerm * that)

A function to check for the equality of two objects.

6.236.4 Member Data Documentation

6.236.4.1 int QuadraticTerm::idx

idx is the index of the row in which the quadratic term appears

Definition at line 353 of file OSInstance.h.

6.236.4.2 int QuadraticTerm::idxOne

idxOne is the index of the first variable in the quadratic term

Definition at line 358 of file OSInstance.h.

6.236.4.3 int QuadraticTerm::idxTwo

idxTwo is the index of the second variable in the quadratic term

Definition at line 363 of file OSInstance.h.

6.236.4.4 double QuadraticTerm::coef

coef is the coefficient of the quadratic term

Definition at line 366 of file OSInstance.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.237 QuadraticTerms Class Reference

a data structure for holding quadratic terms

```
#include <OSGeneral.h>
```

Public Member Functions

- [QuadraticTerms](#) ()
Default constructor.
- [~QuadraticTerms](#) ()

Public Attributes

- int * [rowIndexes](#)
rowIndexes holds an integer array of row indexes of all the quadratic terms.
- int * [varOneIndexes](#)
varOneIndexes holds an integer array of the first variable indexes of all the quadratic terms.

- `int * varTwoIndexes`
varTwoIndexes holds an integer array of the second variable indexes of all the quadratic terms.
- `double * coefficients`
coefficients holds a double array all the quadratic term coefficients.

6.237.1 Detailed Description

a data structure for holding quadratic terms

Definition at line 431 of file OSGeneral.h.

6.237.2 Constructor & Destructor Documentation

6.237.2.1 QuadraticTerms::QuadraticTerms ()

Default constructor.

6.237.2.2 QuadraticTerms::~~QuadraticTerms ()

6.237.3 Member Data Documentation

6.237.3.1 `int* QuadraticTerms::rowIndexes`

rowIndexes holds an integer array of row indexes of all the quadratic terms.

A negative integer corresponds to an objective row, e.g. -1 for 1st objective and -2 for 2nd.

Definition at line 440 of file OSGeneral.h.

6.237.3.2 `int* QuadraticTerms::varOneIndexes`

varOneIndexes holds an integer array of the first variable indexes of all the quadratic terms.

Definition at line 445 of file OSGeneral.h.

6.237.3.3 `int* QuadraticTerms::varTwoIndexes`

varTwoIndexes holds an integer array of the second variable indexes of all the quadratic terms.

Definition at line 450 of file OSGeneral.h.

6.237.3.4 `double* QuadraticTerms::coefficients`

coefficients holds a double array all the quadratic term coefficients.

Definition at line 455 of file OSGeneral.h.

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

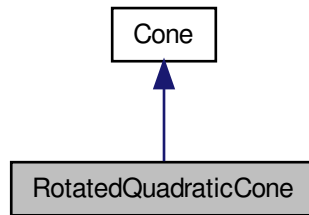
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h`

6.238 RotatedQuadraticCone Class Reference

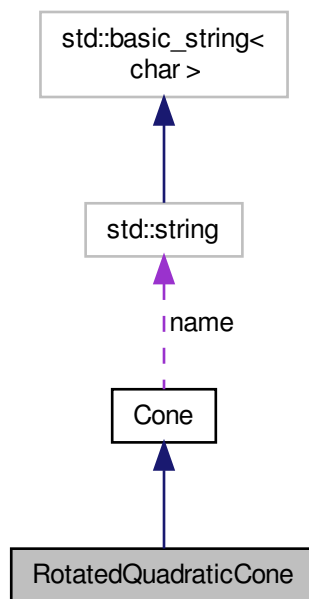
The in-memory representation of a rotated quadratic cone.

```
#include <OSInstance.h>
```

Inheritance diagram for RotatedQuadraticCone:



Collaboration diagram for RotatedQuadraticCone:



Public Member Functions

- [RotatedQuadraticCone](#) ()
The [RotatedQuadraticCone](#) class constructor.
- [~RotatedQuadraticCone](#) ()
The [RotatedQuadraticCone](#) class destructor.

- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [RotatedQuadraticCone](#) object in XML format.
- bool [IsEqual](#) ([RotatedQuadraticCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([RotatedQuadraticCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in `ENUM_CONE_TYPE`)
- int [idx](#)
cones are referenced by an (automatically created) index
- double [normScaleFactor](#)
rotated quadratic cones normally are of the form $x_0x_1 \geq x_2^2 + x_3^2 + \dots$
- int [distortionMatrixIdx](#)
- int [firstAxisDirection](#)
*The indices of the first two component can be changed Since there are possibly many dimensions, each index is coded as $i_0*n_1*n_2*\dots$*
- int [secondAxisDirection](#)

6.238.1 Detailed Description

The in-memory representation of a rotated quadratic cone.

Definition at line 951 of file OSInstance.h.

6.238.2 Constructor & Destructor Documentation

6.238.2.1 [RotatedQuadraticCone::RotatedQuadraticCone](#) ()

The [RotatedQuadraticCone](#) class constructor.

6.238.2.2 [RotatedQuadraticCone::~~RotatedQuadraticCone](#) ()

The [RotatedQuadraticCone](#) class destructor.

6.238.3 Member Function Documentation

6.238.3.1 virtual std::string RotatedQuadraticCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.238.3.2 virtual std::string RotatedQuadraticCone::getConeInXML () [virtual]

Write a [RotatedQuadraticCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.238.3.3 bool RotatedQuadraticCone::isEqual (RotatedQuadraticCone * that)

A function to check for the equality of two objects.

6.238.3.4 bool RotatedQuadraticCone::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.238.3.5 bool RotatedQuadraticCone::deepCopyFrom (RotatedQuadraticCone * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.238.4 Member Data Documentation

6.238.4.1 int RotatedQuadraticCone::numberOfRows

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 964 of file OSInstance.h.

6.238.4.2 int RotatedQuadraticCone::numberOfColumns

Definition at line 965 of file OSInstance.h.

6.238.4.3 int RotatedQuadraticCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 972 of file OSInstance.h.

6.238.4.4 int* RotatedQuadraticCone::otherIndexes

Definition at line 973 of file OSInstance.h.

6.238.4.5 int RotatedQuadraticCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 976 of file OSInstance.h.

6.238.4.6 int RotatedQuadraticCone::idx

cones are referenced by an (automatically created) index

Definition at line 979 of file OSInstance.h.

6.238.4.7 double RotatedQuadraticCone::normScaleFactor

rotated quadratic cones normally are of the form $x_0x_1 \geq x_2^2 + x_3^2 + \dots$

However, the appearance can be modified using a norm factor k and a distortion matrix M to the form $x_0x_1 \geq p(x_2, x_3, \dots) M(x_2, x_3, \dots)'$: $k=1$, $M=-1$.

Definition at line 987 of file OSInstance.h.

6.238.4.8 int RotatedQuadraticCone::distortionMatrixIdx

Definition at line 988 of file OSInstance.h.

6.238.4.9 int RotatedQuadraticCone::firstAxisDirection

The indices of the first two component can be changed Since there are possibly many dimensions, each index is coded as $i_0*n_1*n_2*\dots$

- $i_1*n_2*n_3*\dots + \dots + i_r$, where i_0, i_1 , etc are zero-based indexes for the different dimensions: $i_0 = 0, 1, \dots, n_0 - 1$, where n_0 is the number of rows, $i_1 = 0, 1, \dots, n_1 - 1$, where n_1 is the number of columns, and so on for higher dimensions (if any) : $i_0 = 0, i_1 = 1$.

Definition at line 999 of file OSInstance.h.

6.238.4.10 int RotatedQuadraticCone::secondAxisDirection

Definition at line 1000 of file OSInstance.h.

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

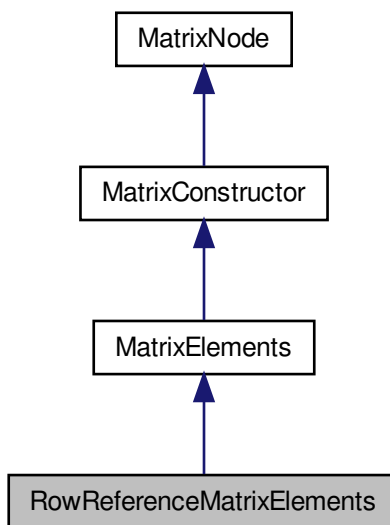
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.239 RowReferenceMatrixElements Class Reference

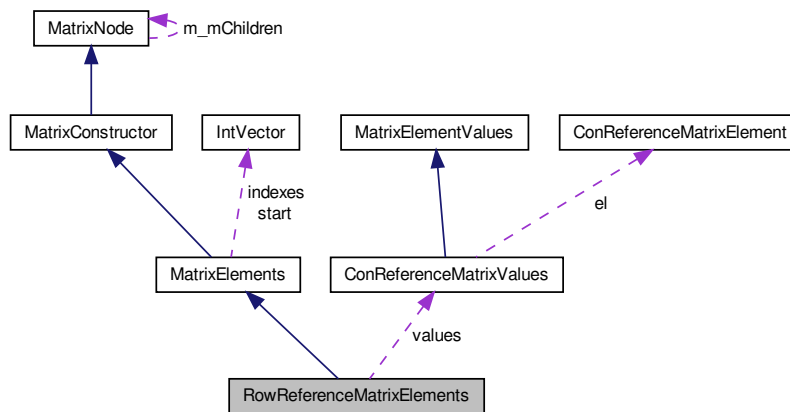
a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element references a row (if index is negative) or constraint (otherwise) This class allows the combining of row and constraint references in a single matrix constructor.

```
#include <OSMatrix.h>
```

Inheritance diagram for RowReferenceMatrixElements:



Collaboration diagram for RowReferenceMatrixElements:



Public Member Functions

- [RowReferenceMatrixElements](#) ()
- [~RowReferenceMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [RowReferenceMatrixElements](#) * [cloneMatrixNode](#) ()
- bool [isEqual](#) ([RowReferenceMatrixElements](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([RowReferenceMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ConReferenceMatrixValues](#) * [values](#)
The row references (indexes of core rows plus value type) of the elements.

6.239.1 Detailed Description

a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element references a row (if index is negative) or constraint (otherwise) This class allows the combining of row and constraint references in a single matrix constructor.

Definition at line 1265 of file OSMatrix.h.

6.239.2 Constructor & Destructor Documentation

6.239.2.1 RowReferenceMatrixElements::RowReferenceMatrixElements ()

6.239.2.2 RowReferenceMatrixElements::~~RowReferenceMatrixElements ()

6.239.3 Member Function Documentation

6.239.3.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE RowReferenceMatrixElements::getNodeType () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.239.3.2 virtual ENUM_MATRIX_TYPE RowReferenceMatrixElements::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.239.3.3 virtual std::string RowReferenceMatrixElements::getNodeName () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.239.3.4 virtual std::string RowReferenceMatrixElements::getMatrixNodeInXML () [virtual]

The following method writes the row reference elements in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Remarks

Since row reference elements do not exist in the schema, they must be separated into objective and constraint references.

Implements [MatrixNode](#).

6.239.3.5 virtual bool RowReferenceMatrixElements::alignsOnBlockBoundary (int firstRow, int firstColumn, int nRows, int nCols) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.239.3.6 virtual RowReferenceMatrixElements* RowReferenceMatrixElements::cloneMatrixNode () [virtual]

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.239.3.7 bool RowReferenceMatrixElements::isEqual (RowReferenceMatrixElements * that)

A function to check for the equality of two objects.

6.239.3.8 bool RowReferenceMatrixElements::setRandom (double density, bool conformant, int iMin, int iMax)

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin,:</i>	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax,:</i>	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.239.3.9 bool RowReferenceMatrixElements::deepCopyFrom (RowReferenceMatrixElements * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that,:</i>	the instance from which information is to be copied
---------------	---

Returns

whether the copy was created successfully

6.239.4 Member Data Documentation

6.239.4.1 ConReferenceMatrixValues* RowReferenceMatrixElements::values

The row references (indexes of core rows plus value type) of the elements.

This information is recycled from the ConReferenceMatrix class. Negative indices describe objectives (in which case the value type must be ENUM_CONREFERENCE_VALUETYPE_value).

Row reference elements are not part of the OSiL schema; they only exist in the consolidated form of a matrix (where they are the ONLY constructor).

Definition at line 1276 of file OSMatrix.h.

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

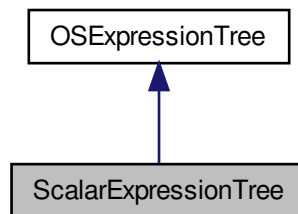
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h](#)

6.240 ScalarExpressionTree Class Reference

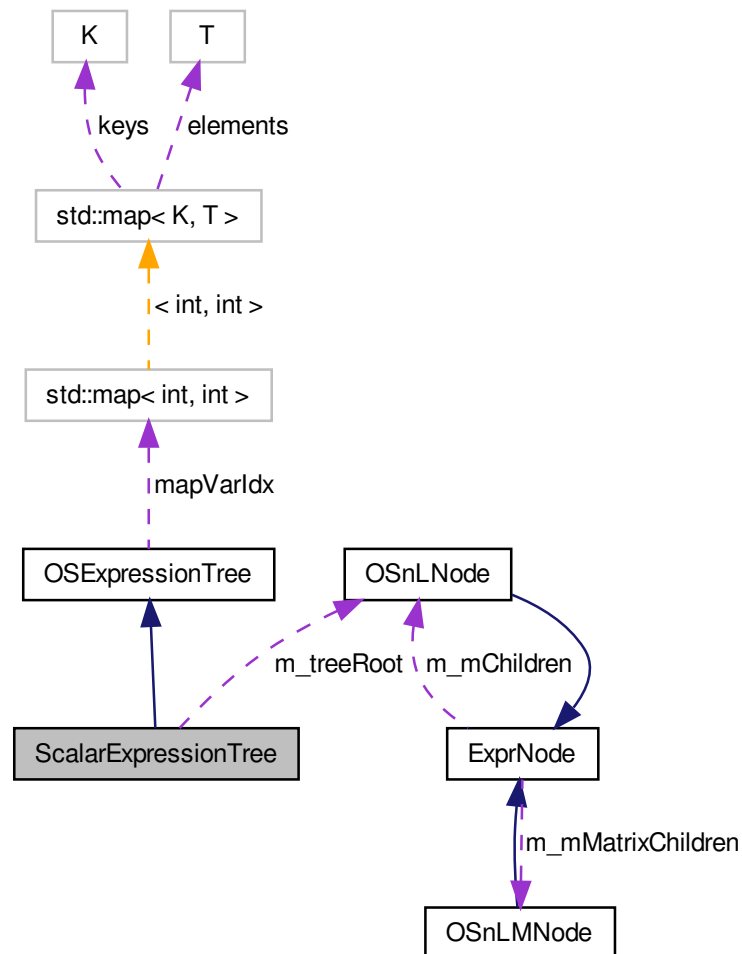
Used to hold part of the instance in memory.

```
#include <OExpressionTree.h>
```

Inheritance diagram for ScalarExpressionTree:



Collaboration diagram for ScalarExpressionTree:



Public Member Functions

- [ScalarExpressionTree](#) ()
default constructor.
- [~ScalarExpressionTree](#) ()
default destructor.
- `bool` [IsEqual](#) ([ScalarExpressionTree](#) *that)
A function to check for the equality of two objects.
- `std::vector< ExprNode * >` [getPrefixFromExpressionTree](#) ()
Get a vector of pointers to ExprNodes that correspond to a scalar-valued OSExpressionTree in prefix format.
- `std::vector< ExprNode * >` [getPostfixFromExpressionTree](#) ()
Get a vector of pointers to ExprNodes that correspond to a scalar-valued OSExpressionTree in postfix format.

- `std::map< int, int > * getVariableIndicesMap ()`
Retrieve a map of the indices of the variables that are in the expression tree.
- `double calculateFunction (double *x, bool new_x)`
Calculate the expression tree function value given the current variable values using the [calculateFunction](#) method of [OSnLNode](#).

Public Attributes

- `OSnLNode * m_treeRoot`
[m_treeRoot](#) holds the root node (of [OSnLNode](#) type) of the expression tree.

6.240.1 Detailed Description

Used to hold part of the instance in memory.

Remarks

This class stores the OSiL instance in memory as an expression tree.

Definition at line 99 of file `OSExpressionTree.h`.

6.240.2 Constructor & Destructor Documentation

6.240.2.1 `ScalarExpressionTree::ScalarExpressionTree ()`

default constructor.

6.240.2.2 `ScalarExpressionTree::~~ScalarExpressionTree ()`

default destructor.

6.240.3 Member Function Documentation

6.240.3.1 `bool ScalarExpressionTree::isEqual (ScalarExpressionTree * that)`

A function to check for the equality of two objects.

6.240.3.2 `std::vector<ExprNode*> ScalarExpressionTree::getPrefixFromExpressionTree ()`

Get a vector of pointers to `ExprNodes` that correspond to a scalar-valued [OSExpressionTree](#) in prefix format.

Returns

the expression tree as a vector of `ExprNodes` in prefix.

6.240.3.3 `std::vector<ExprNode*> ScalarExpressionTree::getPostfixFromExpressionTree ()`

Get a vector of pointers to `ExprNodes` that correspond to a scalar-valued [OSExpressionTree](#) in postfix format.

Returns

the expression tree as a vector of `ExprNodes` in postfix.

6.240.3.4 `std::map<int, int>* ScalarExpressionTree::getVariableIndicesMap ()`

Retrieve a map of the indices of the variables that are in the expression tree.

Returns

a map of the variables in the current expression tree.

6.240.3.5 `double ScalarExpressionTree::calculateFunction (double * x, bool new_x)`

Calculate the expression tree function value given the current variable values using the calculateFunction method of [OSnLNode](#).

If the function has been calculated, the method will retrieve it.

Parameters

<code>x</code>	holds the values of the variables in a double array.
<code>new_x</code>	is false if any evaluation method was previously called for the current x

Returns

the expression tree function value given the current variable values.

6.240.4 Member Data Documentation

6.240.4.1 `OSnLNode* ScalarExpressionTree::m_treeRoot`

`m_treeRoot` holds the root node (of [OSnLNode](#) type) of the expression tree.

Definition at line 106 of file `OSExpressionTree.h`.

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

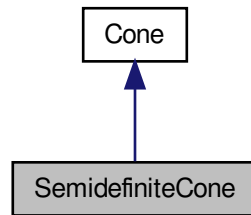
- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSExpressionTree.h`

6.241 SemidefiniteCone Class Reference

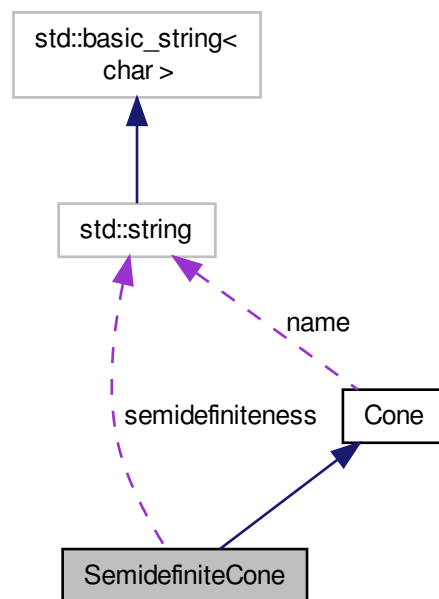
The in-memory representation of a cone of semidefinite matrices.

```
#include <OSInstance.h>
```

Inheritance diagram for SemidefiniteCone:



Collaboration diagram for SemidefiniteCone:



Public Member Functions

- [SemidefiniteCone](#) ()
The *SemidefiniteCone* class constructor.
- [~SemidefiniteCone](#) ()
The *SemidefiniteCone* class destructor.

- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [SemidefiniteCone](#) object in XML format.
- bool [IsEqual](#) ([SemidefiniteCone](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([SemidefiniteCone](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfRows](#)
Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.
- int [numberOfColumns](#)
- int [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- int * [otherIndexes](#)
- int [coneType](#)
The type of the cone (one of the values in ENUM_CONE_TYPE)
- int [idx](#)
cones are referenced by an (automatically created) index
- std::string [semidefiniteness](#)
we need to distinguish positive and negative semidefiniteness
- bool [isPositiveSemiDefinite](#)
information about semidefiniteness is also tracked in a boolean variable

6.241.1 Detailed Description

The in-memory representation of a cone of semidefinite matrices.

Definition at line 1046 of file OSInstance.h.

6.241.2 Constructor & Destructor Documentation

6.241.2.1 [SemidefiniteCone::SemidefiniteCone](#) ()

The [SemidefiniteCone](#) class constructor.

6.241.2.2 [SemidefiniteCone::~~SemidefiniteCone](#) ()

The [SemidefiniteCone](#) class destructor.

6.241.3 Member Function Documentation

6.241.3.1 virtual std::string [SemidefiniteCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

6.241.3.2 `virtual std::string SemidefiniteCone::getConeInXML () [virtual]`

Write a [SemidefiniteCone](#) object in XML format.

This is used by [OSiLWriter](#) to write a <cone> element.

Returns

the cone and its children as an XML string.

Implements [Cone](#).

6.241.3.3 `bool SemidefiniteCone::isEqual (SemidefiniteCone * that)`

A function to check for the equality of two objects.

6.241.3.4 `bool SemidefiniteCone::setRandom (double density, bool conformant, int iMin, int iMax)`

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [Cone](#).

6.241.3.5 `bool SemidefiniteCone::deepCopyFrom (SemidefiniteCone * that)`

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.241.4 Member Data Documentation

6.241.4.1 `int SemidefiniteCone::numberOfRows`

Every cone has (at least) two dimensions; no distinction is made between vector cones and matrix cones.

Definition at line 1059 of file OSInstance.h.

6.241.4.2 int SemidefiniteCone::numberOfColumns

Definition at line 1060 of file OSInstance.h.

6.241.4.3 int SemidefiniteCone::numberOfOtherIndexes

Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).

We therefore allow additional dimensions.

Definition at line 1067 of file OSInstance.h.

6.241.4.4 int* SemidefiniteCone::otherIndexes

Definition at line 1068 of file OSInstance.h.

6.241.4.5 int SemidefiniteCone::coneType

The type of the cone (one of the values in ENUM_CONE_TYPE)

Definition at line 1071 of file OSInstance.h.

6.241.4.6 int SemidefiniteCone::idx

cones are referenced by an (automatically created) index

Definition at line 1074 of file OSInstance.h.

6.241.4.7 std::string SemidefiniteCone::semidefiniteness

we need to distinguish positive and negative semidefiniteness

Definition at line 1077 of file OSInstance.h.

6.241.4.8 bool SemidefiniteCone::isPositiveSemiDefinite

information about semidefiniteness is also tracked in a boolean variable

Definition at line 1080 of file OSInstance.h.

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

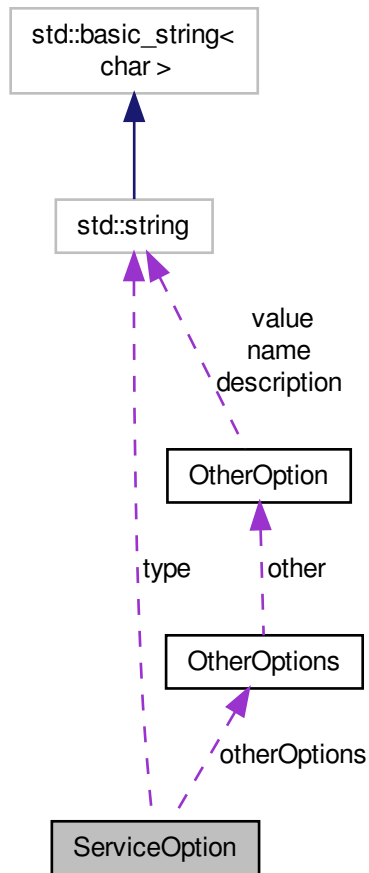
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.242 ServiceOption Class Reference

the [ServiceOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for ServiceOption:



Public Member Functions

- `ServiceOption ()`
Default constructor.
- `~ServiceOption ()`
Class destructor.
- `bool isEqual (ServiceOption *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.
- `bool deepCopyFrom (ServiceOption *that)`
A function to make a deep copy of an instance of this class.

Public Attributes

- `std::string` [type](#)
the service type
- [OtherOptions](#) * [otherOptions](#)
the list of other service options

6.242.1 Detailed Description

the [ServiceOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 610 of file OSOption.h.

6.242.2 Constructor & Destructor Documentation

6.242.2.1 ServiceOption::ServiceOption ()

Default constructor.

6.242.2.2 ServiceOption::~~ServiceOption ()

Class destructor.

6.242.3 Member Function Documentation

6.242.3.1 bool ServiceOption::isEqual ([ServiceOption](#) * *that*)

A function to check for the equality of two objects.

6.242.3.2 bool ServiceOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.242.3.3 bool ServiceOption::deepCopyFrom (ServiceOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.242.4 Member Data Documentation

6.242.4.1 std::string ServiceOption::type

the service type

Definition at line 615 of file OSOption.h.

6.242.4.2 OtherOptions* ServiceOption::otherOptions

the list of other service options

Definition at line 618 of file OSOption.h.

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

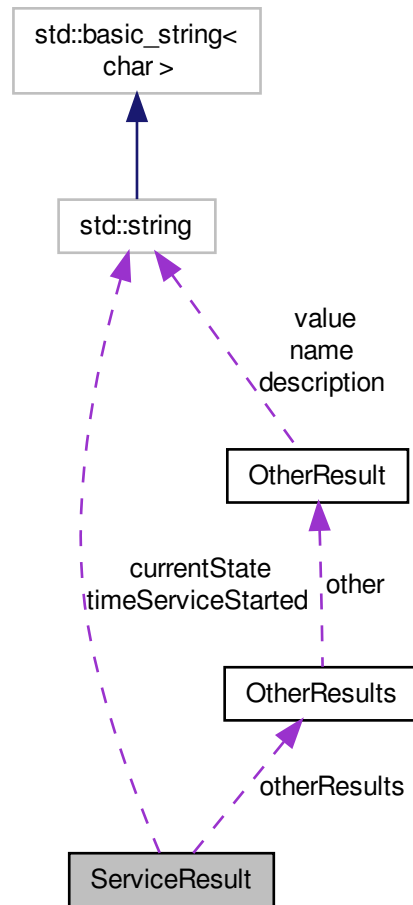
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.243 ServiceResult Class Reference

The [ServiceResult](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for ServiceResult:



Public Member Functions

- [ServiceResult](#) ()
Default constructor.
- [~ServiceResult](#) ()
Class destructor.
- `bool` [isEqual](#) ([ServiceResult](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string [currentState](#)
a string describing the current state of the service
- int [currentJobCount](#)
the number of jobs currently running
- int [totalJobsSoFar](#)
total jobs processed so far
- std::string [timeServiceStarted](#)
the time when the service was started
- double [serviceUtilization](#)
service utilization
- [OtherResults](#) * [otherResults](#)
a pointer to the [OtherResults](#) class

6.243.1 Detailed Description

The [ServiceResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the system information that is defined in the OSrL schema.

Definition at line 414 of file OSResult.h.

6.243.2 Constructor & Destructor Documentation

6.243.2.1 ServiceResult::ServiceResult ()

Default constructor.

6.243.2.2 ServiceResult::~~ServiceResult ()

Class destructor.

6.243.3 Member Function Documentation

6.243.3.1 bool ServiceResult::isEqual (ServiceResult * *that*)

A function to check for the equality of two objects.

6.243.3.2 bool `ServiceResult::setRandom` (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.243.4 Member Data Documentation

6.243.4.1 std::string `ServiceResult::currentState`

a string describing the current state of the service

Definition at line 419 of file `OSResult.h`.

6.243.4.2 int `ServiceResult::currentJobCount`

the number of jobs currently running

Definition at line 422 of file `OSResult.h`.

6.243.4.3 int `ServiceResult::totalJobsSoFar`

total jobs processed so far

Definition at line 425 of file `OSResult.h`.

6.243.4.4 std::string `ServiceResult::timeServiceStarted`

the time when the service was started

Definition at line 428 of file `OSResult.h`.

6.243.4.5 double `ServiceResult::serviceUtilization`

service utilization

Definition at line 431 of file `OSResult.h`.

6.243.4.6 OtherResults* `ServiceResult::otherResults`

a pointer to the [OtherResults](#) class

Definition at line 435 of file `OSResult.h`.

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

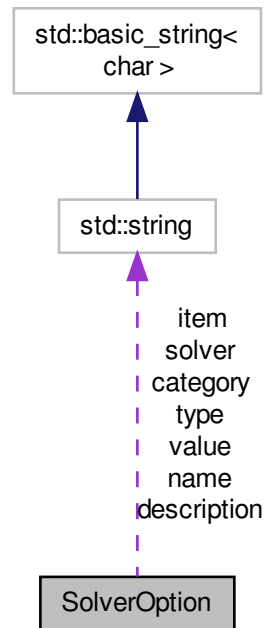
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.244 SolverOption Class Reference

the [SolverOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for SolverOption:



Public Member Functions

- [SolverOption](#) ()
Default constructor.
- [~SolverOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([SolverOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([SolverOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- `std::string` [name](#)
the name of the option
- `std::string` [value](#)
the value of the option
- `std::string` [solver](#)

- the solver to which the option applies*
 - `std::string` [category](#)
 - the category to which the option belongs*
 - `std::string` [type](#)
 - the type of the option value (integer, double, boolean, string)*
 - `std::string` [description](#)
 - the description of the option*
 - `int` [numberOfItems](#)
 - the number of items (additional pieces of data) of the option*
 - `std::string * item`
 - the list of items of the option*

6.244.1 Detailed Description

the [SolverOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3344 of file OSOption.h.

6.244.2 Constructor & Destructor Documentation

6.244.2.1 SolverOption::SolverOption ()

Default constructor.

6.244.2.2 SolverOption::~~SolverOption ()

Class destructor.

6.244.3 Member Function Documentation

6.244.3.1 bool SolverOption::IsEqual (SolverOption * *that*)

A function to check for the equality of two objects.

6.244.3.2 bool SolverOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.244.3.3 bool SolverOption::deepCopyFrom (SolverOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.244.4 Member Data Documentation**6.244.4.1 std::string SolverOption::name**

the name of the option

Definition at line 3349 of file OSOption.h.

6.244.4.2 std::string SolverOption::value

the value of the option

Definition at line 3352 of file OSOption.h.

6.244.4.3 std::string SolverOption::solver

the solver to which the option applies

Definition at line 3355 of file OSOption.h.

6.244.4.4 std::string SolverOption::category

the category to which the option belongs

Definition at line 3358 of file OSOption.h.

6.244.4.5 std::string SolverOption::type

the type of the option value (integer, double, boolean, string)

Definition at line 3361 of file OSOption.h.

6.244.4.6 std::string SolverOption::description

the description of the option

Definition at line 3364 of file OSOption.h.

6.244.4.7 int SolverOption::numberOfItems

the number of items (additional pieces of data) of the option

Definition at line 3367 of file OSOption.h.

6.244.4.8 std::string* SolverOption::item

the list of items of the option

Definition at line 3370 of file OSOption.h.

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

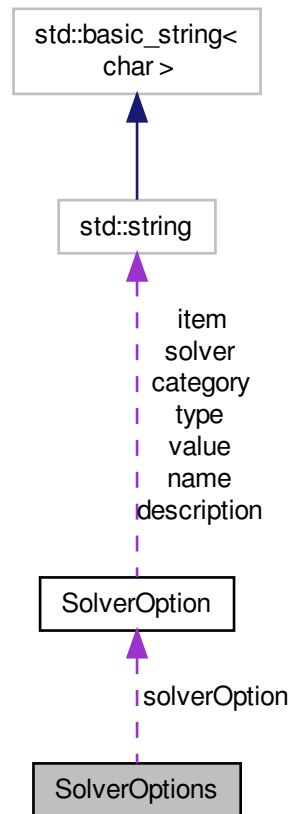
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.245 SolverOptions Class Reference

the [SolverOptions](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for SolverOptions:



Public Member Functions

- [SolverOptions](#) ()
Default constructor.
- [~SolverOptions](#) ()
Class destructor.
- bool [isEqual](#) ([SolverOptions](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([SolverOptions](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setSolverOptions](#) (int numberOfOptions, [SolverOption](#) **solverOption)
A function to set an array of solver options.
- bool [addSolverOption](#) (std::string name, std::string value, std::string solver, std::string category, std::string type, std::string description)

A function to add a solver option.

Public Attributes

- int [numberOfSolverOptions](#)
the number of solver options
- [SolverOption](#) ** [solverOption](#)
the list of solver options

6.245.1 Detailed Description

the [SolverOptions](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 3418 of file OSOption.h.

6.245.2 Constructor & Destructor Documentation

6.245.2.1 SolverOptions::SolverOptions ()

Default constructor.

6.245.2.2 SolverOptions::~~SolverOptions ()

Class destructor.

6.245.3 Member Function Documentation

6.245.3.1 bool SolverOptions::isEqual (SolverOptions * *that*)

A function to check for the equality of two objects.

6.245.3.2 bool SolverOptions::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.245.3.3 bool SolverOptions::deepCopyFrom (SolverOptions * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.245.3.4 bool SolverOptions::setSolverOptions (int *numberOfOptions*, SolverOption ** *solverOption*)

A function to set an array of solver options.

Parameters

<i>numberOfOptions</i> ,:	number of solver options to be set
<i>solverOption</i> ,:	the array of solver options that are to be set

6.245.3.5 bool SolverOptions::addSolverOption (std::string *name*, std::string *value*, std::string *solver*, std::string *category*, std::string *type*, std::string *description*)

A function to add a solver option.

Parameters

<i>name</i> ,:	the name of the solver option (required)
<i>value</i> ,:	a value associated with the option (optional)
<i>solver</i> ,:	the solver to which the option applies (optional)
<i>category</i> ,:	the category (and subcategories) of the option (optional)
<i>type</i> ,:	the type of the option (optional)
<i>description</i> ,:	a description associated with the option (optional)

6.245.4 Member Data Documentation

6.245.4.1 int SolverOptions::numberOfSolverOptions

the number of solver options

Definition at line 3423 of file OSOption.h.

6.245.4.2 SolverOption** SolverOptions::solverOption

the list of solver options

Definition at line 3426 of file OSOption.h.

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

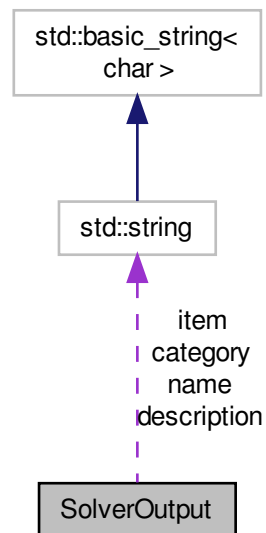
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h](#)

6.246 SolverOutput Class Reference

The [SolverOutput](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for SolverOutput:



Public Member Functions

- [SolverOutput](#) ()
Default constructor.
- [~SolverOutput](#) ()
Class destructor.
- `bool` [isEqual](#) ([SolverOutput](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `std::string` [name](#)

- the name of the result the user is defining*
- std::string [category](#)
this element allows a specific category to be associated with this particular type of result
- std::string [description](#)
a brief description of the type of result
- int [numberOfItems](#)
the number of items contained in this otherSolutionResult
- std::string * [item](#)
an array of items (string-valued)

6.246.1 Detailed Description

The [SolverOutput](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that allows the solver to report an arbitrary result associated with the solution.

Definition at line 2118 of file OSResult.h.

6.246.2 Constructor & Destructor Documentation

6.246.2.1 SolverOutput::SolverOutput ()

Default constructor.

6.246.2.2 SolverOutput::~~SolverOutput ()

Class destructor.

6.246.3 Member Function Documentation

6.246.3.1 bool SolverOutput::isEqual (SolverOutput * *that*)

A function to check for the equality of two objects.

6.246.3.2 bool SolverOutput::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.246.4 Member Data Documentation

6.246.4.1 std::string SolverOutput::name

the name of the result the user is defining

Definition at line 2123 of file OSResult.h.

6.246.4.2 std::string SolverOutput::category

this element allows a specific category to be associated with this particular type of result

Definition at line 2128 of file OSResult.h.

6.246.4.3 std::string SolverOutput::description

a brief description of the type of result

Definition at line 2131 of file OSResult.h.

6.246.4.4 int SolverOutput::numberOfItems

the number of items contained in this otherSolutionResult

Definition at line 2135 of file OSResult.h.

6.246.4.5 std::string* SolverOutput::item

an array of items (string-valued)

Definition at line 2139 of file OSResult.h.

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

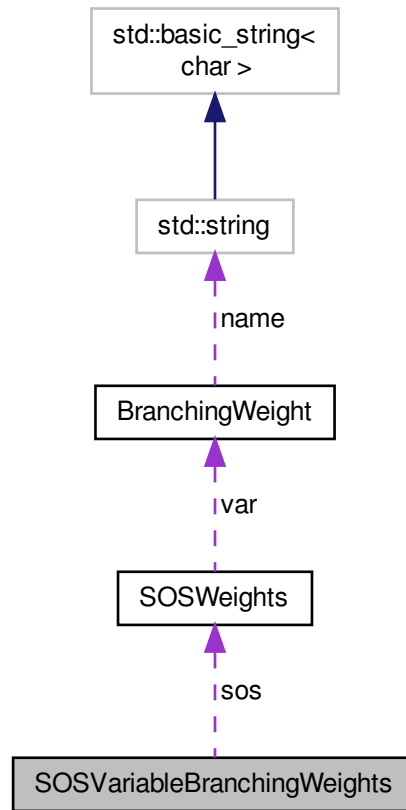
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.247 SOSVariableBranchingWeights Class Reference

the [SOSVariableBranchingWeights](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for SOSVariableBranchingWeights:



Public Member Functions

- [SOSVariableBranchingWeights \(\)](#)
Default constructor.
- [~SOSVariableBranchingWeights \(\)](#)
Class destructor.
- bool [isEqual](#) ([SOSVariableBranchingWeights](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([SOSVariableBranchingWeights](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setSOS](#) (int numberOfSOS, [SOSWeights](#) **sos)
A function to set an array of <sos> elements.
- bool [addSOS](#) (int sosIdx, int nvar, double weight, int *idx, double *value, std::string *name)
A function to add an <sos> element.

Public Attributes

- int [numberOfSOS](#)
number of <sos> children
- [SOSWeights](#) ** [sos](#)
branching weights for the SOS

6.247.1 Detailed Description

the [SOSVariableBranchingWeights](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/11/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1853 of file OSOption.h.

6.247.2 Constructor & Destructor Documentation

6.247.2.1 SOSVariableBranchingWeights::SOSVariableBranchingWeights ()

Default constructor.

6.247.2.2 SOSVariableBranchingWeights::~~SOSVariableBranchingWeights ()

Class destructor.

6.247.3 Member Function Documentation

6.247.3.1 bool SOSVariableBranchingWeights::isEqual (SOSVariableBranchingWeights * *that*)

A function to check for the equality of two objects.

6.247.3.2 bool SOSVariableBranchingWeights::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.247.3.3 bool SOSVariableBranchingWeights::deepCopyFrom (SOSVariableBranchingWeights * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.247.3.4 bool SOSVariableBranchingWeights::setSOS (int *numberOfSOS*, SOSWeights ** *sos*)

A function to set an array of <sos> elements.

Parameters

<i>numberOfSOS</i> ,:	number of <sos> elements to be set
<i>sos</i> ,:	the array of <sos> elements that are to be set

6.247.3.5 bool SOSVariableBranchingWeights::addSOS (int *sosIdx*, int *nvar*, double *weight*, int * *idx*, double * *value*, std::string * *name*)

A function to add an <sos> element.

Parameters

<i>sosIdx</i> ,:	the index of the SOS that is to be added (refer back to OSiL file)
<i>nvar</i> ,:	the number of variables in this SOS that are to be given weights
<i>weight</i> ,:	a selection weight for the entire group of variables
<i>idx</i> ,:	an array of variable indices
<i>value</i> ,:	the array of corresponding selection weights

6.247.4 Member Data Documentation

6.247.4.1 int SOSVariableBranchingWeights::numberOfSOS

number of <sos> children

Definition at line 1858 of file OSOption.h.

6.247.4.2 SOSWeights** SOSVariableBranchingWeights::sos

branching weights for the SOS

Definition at line 1861 of file OSOption.h.

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

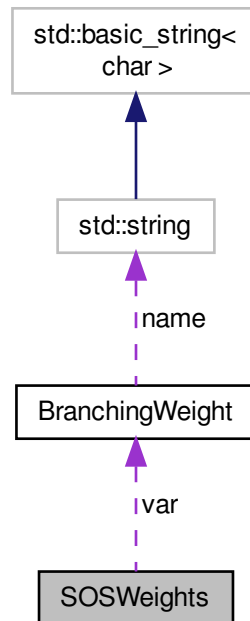
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.248 SOSWeights Class Reference

the [SOSWeights](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for SOSWeights:



Public Member Functions

- [SOSWeights](#) ()
Default constructor.
- [~SOSWeights](#) ()
Class destructor.
- `bool` [IsEqual](#) ([SOSWeights](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([SOSWeights](#) *that)
A function to make a deep copy of an instance of this class.
- `bool` [setVar](#) (int [numberOfVar](#), [BranchingWeight](#) **var)
A function to set an array of elements.
- `bool` [addVar](#) (int idx, double value)
A function to add a element.

Public Attributes

- int [sosIdx](#)
index of the SOS (to match the OSiL file)
- double [groupWeight](#)
branching weight for the entire SOS
- int [numberOfVar](#)
number of children
- [BranchingWeight](#) ** [var](#)
branching weights for individual variables

6.248.1 Detailed Description

the [SOSWeights](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/11/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 1775 of file OSOption.h.

6.248.2 Constructor & Destructor Documentation

6.248.2.1 SOSWeights::SOSWeights ()

Default constructor.

6.248.2.2 SOSWeights::~~SOSWeights ()

Class destructor.

6.248.3 Member Function Documentation

6.248.3.1 bool SOSWeights::isEqual (SOSWeights * *that*)

A function to check for the equality of two objects.

6.248.3.2 bool SOSWeights::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.248.3.3 bool SOSWeights::deepCopyFrom (SOSWeights * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.248.3.4 bool SOSWeights::setVar (int *numberOfVar*, BranchingWeight ** *var*)

A function to set an array of *elements*.

Parameters

<i>numberOfVar</i> ,:	number of <i>elements</i> to be set
<i>var</i> ,:	the array of <i>elements</i> that are to be set

6.248.3.5 bool SOSWeights::addVar (int *idx*, double *value*)

A function to add a *element*.

Parameters

<i>idx</i> ,:	the index of the variable to be given a branching weight
<i>value</i> ,:	the branching weight to be added

6.248.4 Member Data Documentation**6.248.4.1** int SOSWeights::sosIdx

index of the SOS (to match the OSiL file)

Definition at line 1780 of file OSOption.h.

6.248.4.2 double SOSWeights::groupWeight

branching weight for the entire SOS

Definition at line 1783 of file OSOption.h.

6.248.4.3 int SOSWeights::numberOfVar

number of *children*

Definition at line 1786 of file OSOption.h.

6.248.4.4 BranchingWeight** SOSWeights::var

branching weights for individual variables

Definition at line 1789 of file OSOption.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h

6.249 SparseHessianMatrix Class Reference

The in-memory representation of a [SparseHessianMatrix](#).

```
#include <OSGeneral.h>
```

Public Member Functions

- [SparseHessianMatrix](#) ()
Default constructor.
- [SparseHessianMatrix](#) (int startSize, int valueSize)
An Alternative Constructor.
- [~SparseHessianMatrix](#) ()
Default destructor.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- int [hessDimension](#)
hessDimension is the number of nonzeros in each array.
- int * [hessRowIdx](#)
hessRowIdx is an integer array of row indices in the range 0, ..., n - 1.
- int * [hessColIdx](#)
hessColIdx is an integer array of column indices in the range 0, ..., n - 1.
- double * [hessValues](#)
hessValues is a double array of the Hessian values.

6.249.1 Detailed Description

The in-memory representation of a [SparseHessianMatrix](#).

Remarks

Store an upper-triangular Hessian Matrix in sparse format

Assume there are n variables in what follows

Definition at line 376 of file OSGeneral.h.

6.249.2 Constructor & Destructor Documentation

6.249.2.1 SparseHessianMatrix::SparseHessianMatrix ()

Default constructor.

6.249.2.2 SparseHessianMatrix::SparseHessianMatrix (int *startSize*, int *valueSize*)

An Alternative Constructor.

Parameters

<i>startSize</i>	holds the size of the arrays.
<i>valueSize</i>	holds the size of the value and index arrays.

6.249.2.3 SparseHessianMatrix::~SparseHessianMatrix ()

Default destructor.

6.249.3 Member Data Documentation

6.249.3.1 bool SparseHessianMatrix::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 384 of file OSGeneral.h.

6.249.3.2 int SparseHessianMatrix::hessDimension

hessDimension is the number of nonzeros in each array.

Definition at line 389 of file OSGeneral.h.

6.249.3.3 int* SparseHessianMatrix::hessRowIdx

hessRowIdx is an integer array of row indices in the range 0, ..., $n - 1$.

Definition at line 394 of file OSGeneral.h.

6.249.3.4 int* SparseHessianMatrix::hessColIdx

hessColIdx is an integer array of column indices in the range 0, ..., $n - 1$.

Definition at line 399 of file OSGeneral.h.

6.249.3.5 double* SparseHessianMatrix::hessValues

hessValues is a double array of the Hessian values.

Definition at line 404 of file OSGeneral.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h

6.250 SparseIntVector Class Reference

a sparse vector data structure for integer vectors

```
#include <OSGeneral.h>
```

Public Member Functions

- [SparseIntVector](#) (int [number](#))

Constructor.

- [SparseIntVector](#) ()

Default Constructor.

- [~SparseIntVector](#) ()

Default destructor.

Public Attributes

- bool [bDeleteArrays](#)

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

- int [number](#)

number is the number of elements in the indexes and values arrays.

- int * [indexes](#)

indexes holds an integer array of indexes whose corresponding values are listed in the same order in the values array.

- int * [values](#)

values holds an integer array of nonzero values.

6.250.1 Detailed Description

a sparse vector data structure for integer vectors

Definition at line 171 of file OSGeneral.h.

6.250.2 Constructor & Destructor Documentation

6.250.2.1 SparseIntVector::SparseIntVector (int *number*)

Constructor.

Parameters

<i>number</i>	holds the size of the vector.
---------------	-------------------------------

6.250.2.2 SparseIntVector::SparseIntVector ()

Default Constructor.

6.250.2.3 SparseIntVector::~~SparseIntVector ()

Default destructor.

6.250.3 Member Data Documentation

6.250.3.1 bool SparseIntVector::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 198 of file OSGeneral.h.

6.250.3.2 int SparseIntVector::number

number is the number of elements in the indexes and values arrays.

Definition at line 203 of file OSGeneral.h.

6.250.3.3 int* SparseIntVector::indexes

indexes holds an integer array of indexes whose corresponding values are listed in the same order in the values array.

Typically those would be nonzero.

Definition at line 210 of file OSGeneral.h.

6.250.3.4 int* SparseIntVector::values

values holds an integer array of nonzero values.

Definition at line 215 of file OSGeneral.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.251 SparseJacobianMatrix Class Reference

a sparse Jacobian matrix data structure

```
#include <OSGeneral.h>
```

Public Member Functions

- [SparseJacobianMatrix \(\)](#)
Default constructor.
- [SparseJacobianMatrix \(int startSize, int valueSize\)](#)
Constructor.
- [~SparseJacobianMatrix \(\)](#)
Default destructor.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- int [startSize](#)
startSize is the dimension of the starts array – should equal number of rows + 1
- int [valueSize](#)
valueSize is the dimension of the values array
- int * [starts](#)
starts holds an integer array of start elements, each start element points to the start of partials for that row
- int * [conVals](#)
conVals holds an integer array of integers, conVals[i] is the number of constant terms in the gradient for row i.
- int * [indexes](#)
indexes holds an integer array of variable indices.
- double * [values](#)
values holds a double array of nonzero partial derivatives

6.251.1 Detailed Description

a sparse Jacobian matrix data structure

Definition at line 300 of file OSGeneral.h.

6.251.2 Constructor & Destructor Documentation

6.251.2.1 SparseJacobianMatrix::SparseJacobianMatrix ()

Default constructor.

6.251.2.2 SparseJacobianMatrix::SparseJacobianMatrix (int *startSize*, int *valueSize*)

Constructor.

Parameters

<i>startSize</i>	holds the size of the start array.
<i>valueSize</i>	holds the size of the value and index arrays.

6.251.2.3 SparseJacobianMatrix::~~SparseJacobianMatrix ()

Default destructor.

6.251.3 Member Data Documentation

6.251.3.1 bool SparseJacobianMatrix::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 308 of file OSGeneral.h.

6.251.3.2 int SparseJacobianMatrix::startSize

startSize is the dimension of the starts array – should equal number of rows + 1

Definition at line 313 of file OSGeneral.h.

6.251.3.3 int SparseJacobianMatrix::valueSize

valueSize is the dimension of the values array

Definition at line 318 of file OSGeneral.h.

6.251.3.4 int* SparseJacobianMatrix::starts

starts holds an integer array of start elements, each start element points to the start of partials for that row

Definition at line 324 of file OSGeneral.h.

6.251.3.5 int* SparseJacobianMatrix::conVals

conVals holds an integer array of integers, conVals[i] is the number of constant terms in the gradient for row i.

Definition at line 330 of file OSGeneral.h.

6.251.3.6 int* SparseJacobianMatrix::indexes

indexes holds an integer array of variable indices.

Definition at line 335 of file OSGeneral.h.

6.251.3.7 double* SparseJacobianMatrix::values

values holds a double array of nonzero partial derivatives

Definition at line 340 of file OSGeneral.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.252 SparseMatrix Class Reference

a sparse matrix data structure

```
#include <OSGeneral.h>
```

Public Member Functions

- [SparseMatrix](#) ()
Default constructor.
- [SparseMatrix](#) (bool isColumnMajor_, int [startSize](#), int [valueSize](#))
Constructor.
- [~SparseMatrix](#) ()
Default destructor.
- bool [display](#) (int secondaryDim)
This method displays data structure in the matrix format.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- bool [isColumnMajor](#)
isColumnMajor holds whether the coefMatrix (AMatrix) holding linear program data is stored by column.
- int [startSize](#)
startSize is the dimension of the starts array
- int [valueSize](#)
valueSize is the dimension of the indexes and values arrays
- int * [starts](#)
starts holds an integer array of start elements in coefMatrix (AMatrix), which points to the start of a column (row) of nonzero elements in coefMatrix (AMatrix).
- int * [indexes](#)
indexes holds an integer array of rowIdx (or colIdx) elements in coefMatrix (AMatrix).
- double * [values](#)
values holds a double array of value elements in coefMatrix (AMatrix), which contains nonzero elements.

6.252.1 Detailed Description

a sparse matrix data structure

Definition at line 223 of file OSGeneral.h.

6.252.2 Constructor & Destructor Documentation

6.252.2.1 SparseMatrix::SparseMatrix ()

Default constructor.

6.252.2.2 SparseMatrix::SparseMatrix (bool *isColumnMajor*_, int *startSize*, int *valueSize*)

Constructor.

Parameters

<i>isColumnMajor</i>	holds whether the coefMatrix (AMatrix) holding linear program data is stored by column. If false, the matrix is stored by row.
<i>startSize</i>	holds the size of the start array.
<i>valueSize</i>	holds the size of the value and index arrays.

6.252.2.3 SparseMatrix::~~SparseMatrix ()

Default destructor.

6.252.3 Member Function Documentation

6.252.3.1 bool SparseMatrix::display (int *secondaryDim*)

This method displays data structure in the matrix format.

Returns

6.252.4 Member Data Documentation

6.252.4.1 bool SparseMatrix::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 230 of file OSGeneral.h.

6.252.4.2 bool SparseMatrix::isColumnMajor

isColumnMajor holds whether the coefMatrix (AMatrix) holding linear program data is stored by column.

If false, the matrix is stored by row.

Definition at line 236 of file OSGeneral.h.

6.252.4.3 int SparseMatrix::startSize

startSize is the dimension of the starts array

Definition at line 241 of file OSGeneral.h.

6.252.4.4 int SparseMatrix::valueSize

valueSize is the dimension of the indexes and values arrays

Definition at line 246 of file OSGeneral.h.

6.252.4.5 int* SparseMatrix::starts

starts holds an integer array of start elements in coefMatrix (AMatrix), which points to the start of a column (row) of nonzero elements in coefMatrix (AMatrix).

Definition at line 252 of file OSGeneral.h.

6.252.4.6 int* SparseMatrix::indexes

indexes holds an integer array of rowIdx (or colIdx) elements in coefMatrix (AMatrix).

If the matrix is stored by column (row), rowIdx (colIdx) is the array of row (column) indices.

Definition at line 258 of file OSGeneral.h.

6.252.4.7 double* SparseMatrix::values

values holds a double array of value elements in coefMatrix (AMatrix), which contains nonzero elements.

Definition at line 264 of file OSGeneral.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.253 SparseVector Class Reference

a sparse vector data structure

```
#include <OSGeneral.h>
```

Public Member Functions

- [SparseVector](#) (int [number](#))
Constructor.
- [SparseVector](#) ()
Default Constructor.
- [~SparseVector](#) ()
Default destructor.

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- int [number](#)
number is the number of elements in the indexes and values arrays.
- int * [indexes](#)
indexes holds an integer array of indexes whose corresponding values are nonzero.
- double * [values](#)
values holds a double array of nonzero values.

6.253.1 Detailed Description

a sparse vector data structure

Definition at line 122 of file OSGeneral.h.

6.253.2 Constructor & Destructor Documentation

6.253.2.1 SparseVector::SparseVector (int *number*)

Constructor.

Parameters

<i>number</i>	holds the size of the vector.
---------------	-------------------------------

6.253.2.2 SparseVector::SparseVector ()

Default Constructor.

6.253.2.3 SparseVector::~~SparseVector ()

Default destructor.

6.253.3 Member Data Documentation

6.253.3.1 bool SparseVector::bDeleteArrays

bDeleteArrays is true if we delete the arrays in garbage collection set to true by default

Definition at line 149 of file OSGeneral.h.

6.253.3.2 int SparseVector::number

number is the number of elements in the indexes and values arrays.

Definition at line 154 of file OSGeneral.h.

6.253.3.3 int* SparseVector::indexes

indexes holds an integer array of indexes whose corresponding values are nonzero.

Definition at line 159 of file OSGeneral.h.

6.253.3.4 double* SparseVector::values

values holds a double array of nonzero values.

Definition at line 164 of file OSGeneral.h.

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

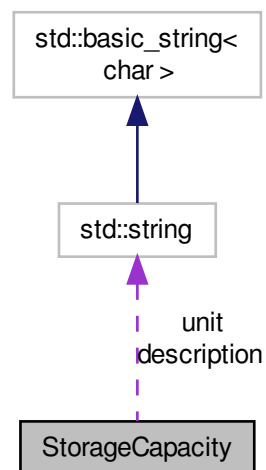
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSGeneral.h](#)

6.254 StorageCapacity Class Reference

the [StorageCapacity](#) class.

```
#include <OSGeneral.h>
```

Collaboration diagram for StorageCapacity:



Public Member Functions

- [StorageCapacity](#) ()
Default constructor.
- [~StorageCapacity](#) ()
Class destructor.
- bool [IsEqual](#) ([StorageCapacity](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([StorageCapacity](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string [unit](#)
the unit in which storage capacity is measured
- std::string [description](#)
additional description about the storage
- double [value](#)
the number of units of storage capacity

6.254.1 Detailed Description

the [StorageCapacity](#) class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

A data structure class that corresponds to an xml element in the OSgL schema.

Definition at line 754 of file OSGeneral.h.

6.254.2 Constructor & Destructor Documentation

6.254.2.1 [StorageCapacity::StorageCapacity](#) ()

Default constructor.

6.254.2.2 [StorageCapacity::~~StorageCapacity](#) ()

Class destructor.

6.254.3 Member Function Documentation

6.254.3.1 bool [StorageCapacity::IsEqual](#) ([StorageCapacity](#) * that)

A function to check for the equality of two objects.

6.254.3.2 bool StorageCapacity::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.254.3.3 bool StorageCapacity::deepCopyFrom (StorageCapacity * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.254.4 Member Data Documentation

6.254.4.1 std::string StorageCapacity::unit

the unit in which storage capacity is measured

Definition at line 759 of file OSGeneral.h.

6.254.4.2 std::string StorageCapacity::description

additional description about the storage

Definition at line 762 of file OSGeneral.h.

6.254.4.3 double StorageCapacity::value

the number of units of storage capacity

Definition at line 765 of file OSGeneral.h.

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

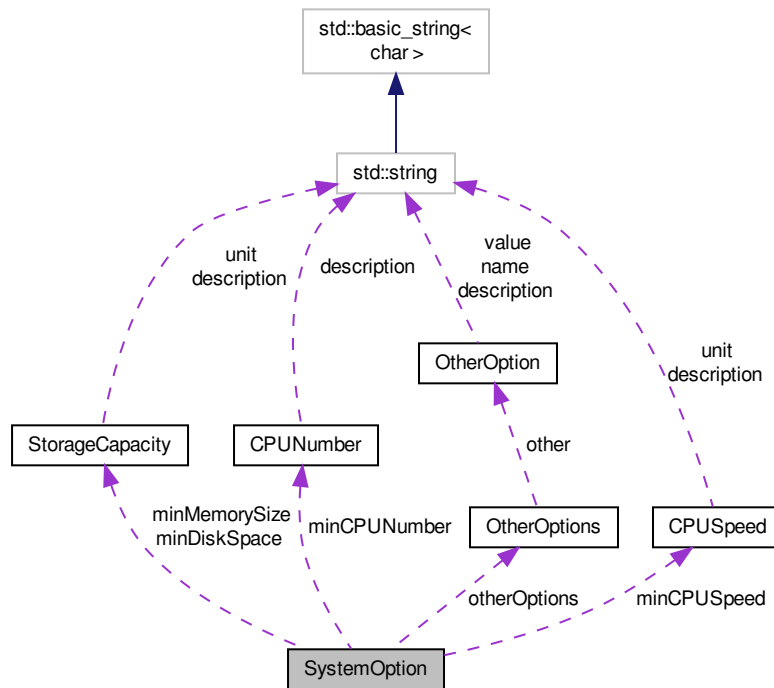
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSGeneral.h](#)

6.255 SystemOption Class Reference

the [SystemOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for SystemOption:



Public Member Functions

- [SystemOption](#) ()
Default constructor.
- [~SystemOption](#) ()
Class destructor.
- `bool` [isEqual](#) ([SystemOption](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- `bool` [deepCopyFrom](#) ([SystemOption](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [StorageCapacity](#) * [minDiskSpace](#)
the minimum disk space required
- [StorageCapacity](#) * [minMemorySize](#)
the minimum memory required
- [CPUSpeed](#) * [minCPUSpeed](#)

- the minimum CPU speed required*
 - `CPUNumber * minCPUNumber`
- the minimum number of processors required*
 - `OtherOptions * otherOptions`
- the list of other system options*

6.255.1 Detailed Description

the [SystemOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 545 of file OSOption.h.

6.255.2 Constructor & Destructor Documentation

6.255.2.1 SystemOption::SystemOption ()

Default constructor.

6.255.2.2 SystemOption::~~SystemOption ()

Class destructor.

6.255.3 Member Function Documentation

6.255.3.1 bool SystemOption::isEqual (SystemOption * *that*)

A function to check for the equality of two objects.

6.255.3.2 bool SystemOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.255.3.3 bool SystemOption::deepCopyFrom (SystemOption * that)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.255.4 Member Data Documentation

6.255.4.1 StorageCapacity* SystemOption::minDiskSpace

the minimum disk space required

Definition at line 550 of file OSOption.h.

6.255.4.2 StorageCapacity* SystemOption::minMemorySize

the minimum memory required

Definition at line 553 of file OSOption.h.

6.255.4.3 CPUSpeed* SystemOption::minCPUSpeed

the minimum CPU speed required

Definition at line 556 of file OSOption.h.

6.255.4.4 CPUNumber* SystemOption::minCPUNumber

the minimum number of processors required

Definition at line 559 of file OSOption.h.

6.255.4.5 OtherOptions* SystemOption::otherOptions

the list of other system options

Definition at line 562 of file OSOption.h.

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

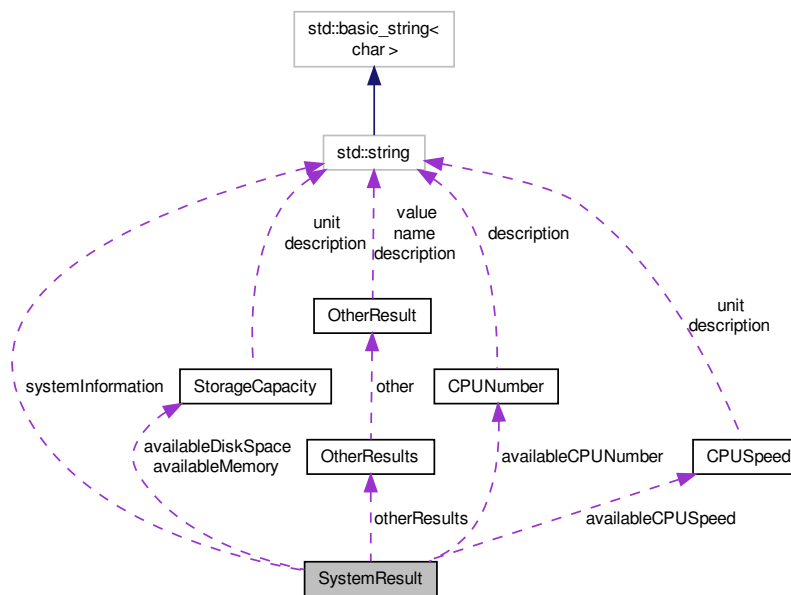
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSOption.h](#)

6.256 SystemResult Class Reference

The [SystemResult](#) Class.

```
#include <OSResult.h>
```


Collaboration diagram for SystemResult:



Public Member Functions

- `SystemResult ()`
Default constructor.
- `~SystemResult ()`
Class destructor.
- `bool IsEqual (SystemResult *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `std::string systemInformation`
a string containing some basic system information
- `StorageCapacity * availableDiskSpace`
a pointer to the `DiskSpace` class
- `StorageCapacity * availableMemory`
a pointer to the `MemorySize` class
- `CPUSpeed * availableCPUSpeed`
a pointer to the `CPUSpeed` class
- `CPUNumber * availableCPUNumber`
a pointer to the `CPUNumber` class
- `OtherResults * otherResults`
a pointer to the `OtherResults` class

6.256.1 Detailed Description

The [SystemResult](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the system information that is defined in the OSrL schema.

Definition at line 348 of file OSResult.h.

6.256.2 Constructor & Destructor Documentation

6.256.2.1 SystemResult::SystemResult ()

Default constructor.

6.256.2.2 SystemResult::~~SystemResult ()

Class destructor.

6.256.3 Member Function Documentation

6.256.3.1 bool SystemResult::isEqual (SystemResult * *that*)

A function to check for the equality of two objects.

6.256.3.2 bool SystemResult::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.256.4 Member Data Documentation

6.256.4.1 `std::string SystemResult::systemInformation`

a string containing some basic system information

Definition at line 353 of file OSResult.h.

6.256.4.2 `StorageCapacity* SystemResult::availableDiskSpace`

a pointer to the DiskSpace class

Definition at line 357 of file OSResult.h.

6.256.4.3 `StorageCapacity* SystemResult::availableMemory`

a pointer to the MemorySize class

Definition at line 361 of file OSResult.h.

6.256.4.4 `CPUSpeed* SystemResult::availableCPUSpeed`

a pointer to the [CPUSpeed](#) class

Definition at line 365 of file OSResult.h.

6.256.4.5 `CPUNumber* SystemResult::availableCPUNumber`

a pointer to the [CPUNumber](#) class

Definition at line 369 of file OSResult.h.

6.256.4.6 `OtherResults* SystemResult::otherResults`

a pointer to the [OtherResults](#) class

Definition at line 373 of file OSResult.h.

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

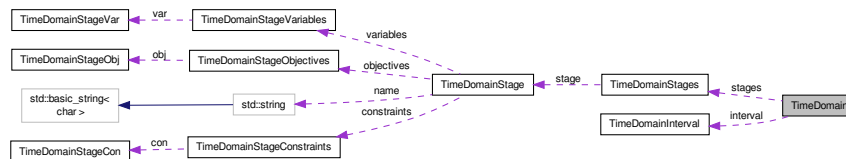
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.257 TimeDomain Class Reference

The in-memory representation of the `<timeDomain>` element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomain:



Public Member Functions

- [TimeDomain](#) ()
The [TimeDomain](#) class constructor.
- [~TimeDomain](#) ()
The [TimeDomain](#) class destructor.

Public Attributes

- [TimeDomainStages](#) * [stages](#)
stages is a pointer to a [Stages](#) object
- [TimeDomainInterval](#) * [interval](#)
interval is a pointer to an [Interval](#) object

6.257.1 Detailed Description

The in-memory representation of the `<timeDomain>` element.

Definition at line 2141 of file `OSInstance.h`.

6.257.2 Constructor & Destructor Documentation

6.257.2.1 TimeDomain::TimeDomain ()

The [TimeDomain](#) class constructor.

6.257.2.2 TimeDomain::~~TimeDomain ()

The [TimeDomain](#) class destructor.

6.257.3 Member Data Documentation

6.257.3.1 TimeDomainStages* TimeDomain::stages

`stages` is a pointer to a [Stages](#) object

Definition at line 2153 of file `OSInstance.h`.

6.257.3.2 TimeDomainInterval* TimeDomain::interval

`interval` is a pointer to an [Interval](#) object

Definition at line 2157 of file `OSInstance.h`.

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

- `/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h`

6.258 TimeDomainInterval Class Reference

```
#include <OSInstance.h>
```

Public Member Functions

- [TimeDomainInterval](#) ()
The [Interval](#) class constructor.
- [~TimeDomainInterval](#) ()
The [Interval](#) class destructor.

Public Attributes

- double [start](#)
start is the start of the planning period in the [<interval>](#) element.
- double [horizon](#)
horizon is the end of the planning period in the [<interval>](#) element.

6.258.1 Detailed Description

Definition at line 2116 of file OSInstance.h.

6.258.2 Constructor & Destructor Documentation

6.258.2.1 TimeDomainInterval::TimeDomainInterval ()

The [Interval](#) class constructor.

6.258.2.2 TimeDomainInterval::~~TimeDomainInterval ()

The [Interval](#) class destructor.

6.258.3 Member Data Documentation

6.258.3.1 double TimeDomainInterval::start

start is the start of the planning period in the [<interval>](#) element.

Definition at line 2129 of file OSInstance.h.

6.258.3.2 double TimeDomainInterval::horizon

horizon is the end of the planning period in the [<interval>](#) element.

Definition at line 2134 of file OSInstance.h.

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

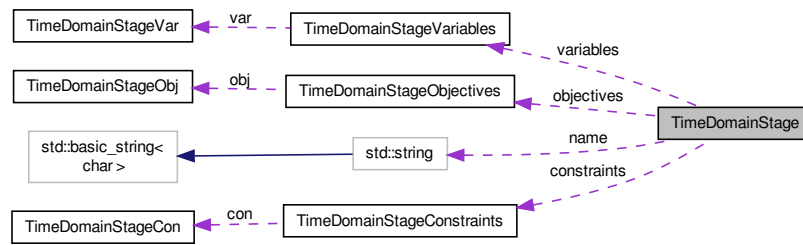
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.259 TimeDomainStage Class Reference

The in-memory representation of the [<stage>](#) element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomainStage:



Public Member Functions

- [TimeDomainStage \(\)](#)
The *TimeDomainStage* class constructor.
- [~TimeDomainStage \(\)](#)
The *TimeDomainStage* class destructor.

Public Attributes

- `std::string` [name](#)
name corresponds to the optional attribute that holds the name of the stage; the default value is empty
- [TimeDomainStageVariables](#) * [variables](#)
variables is a pointer to a *TimeDomainVariables* object
- [TimeDomainStageConstraints](#) * [constraints](#)
constraints is a pointer to a *TimeDomainConstraints* object
- [TimeDomainStageObjectives](#) * [objectives](#)
objectives is a pointer to a *TimeDomainObjectives* object

6.259.1 Detailed Description

The in-memory representation of the `<stage>` element.

Definition at line 2064 of file `OSInstance.h`.

6.259.2 Constructor & Destructor Documentation

6.259.2.1 TimeDomainStage::TimeDomainStage ()

The *TimeDomainStage* class constructor.

6.259.2.2 TimeDomainStage::~~TimeDomainStage ()

The *TimeDomainStage* class destructor.

6.259.3 Member Data Documentation

6.259.3.1 `std::string TimeDomainStage::name`

name corresponds to the optional attribute that holds the name of the stage; the default value is empty

Definition at line 2077 of file OSInstance.h.

6.259.3.2 `TimeDomainStageVariables* TimeDomainStage::variables`

variables is a pointer to a TimeDomainVariables object

Definition at line 2080 of file OSInstance.h.

6.259.3.3 `TimeDomainStageConstraints* TimeDomainStage::constraints`

constraints is a pointer to a TimeDomainConstraints object

Definition at line 2083 of file OSInstance.h.

6.259.3.4 `TimeDomainStageObjectives* TimeDomainStage::objectives`

objectives is a pointer to a TimeDomainObjectives object

Definition at line 2086 of file OSInstance.h.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.260 TimeDomainStageCon Class Reference

The in-memory representation of the `<con>` element.

```
#include <OSInstance.h>
```

Public Member Functions

- [TimeDomainStageCon \(\)](#)
The TimeDomainStageCon class constructor.
- [~TimeDomainStageCon \(\)](#)
The TimeDomainStageCon class destructor.

Public Attributes

- [int idx](#)
idx gives the index of this constraint

6.260.1 Detailed Description

The in-memory representation of the `<con>` element.

Definition at line 1978 of file OSInstance.h.

6.260.2 Constructor & Destructor Documentation

6.260.2.1 TimeDomainStageCon::TimeDomainStageCon ()

The [TimeDomainStageCon](#) class constructor.

6.260.2.2 TimeDomainStageCon::~~TimeDomainStageCon ()

The [TimeDomainStageCon](#) class destructor.

6.260.3 Member Data Documentation

6.260.3.1 int TimeDomainStageCon::idx

idx gives the index of this constraint

Definition at line 1989 of file OSInstance.h.

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

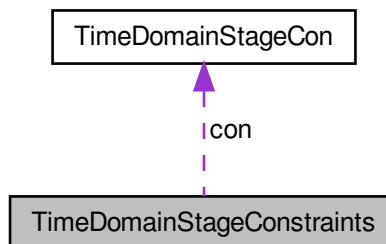
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.261 TimeDomainStageConstraints Class Reference

The in-memory representation of the **<constraints>** child of the **<stage>** element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomainStageConstraints:



Public Member Functions

- [TimeDomainStageConstraints](#) ()
The [TimeDomainStageConstraints](#) class constructor.
- [~TimeDomainStageConstraints](#) ()
The [TimeDomainStageConstraints](#) class destructor.

Public Attributes

- int [numberOfConstraints](#)
numberOfConstraints gives the number of constraints contained in this stage
- int [startIdx](#)
startIdx gives the number of the first constraint contained in this stage
- [TimeDomainStageCon](#) ** [con](#)
con is a pointer to an array of [TimeDomainStageCon](#) object pointers

6.261.1 Detailed Description

The in-memory representation of the **<constraints>** child of the **<stage>** element.

Definition at line 1996 of file OSInstance.h.

6.261.2 Constructor & Destructor Documentation

6.261.2.1 TimeDomainStageConstraints::TimeDomainStageConstraints ()

The [TimeDomainStageConstraints](#) class constructor.

6.261.2.2 TimeDomainStageConstraints::~TimeDomainStageConstraints ()

The [TimeDomainStageConstraints](#) class destructor.

6.261.3 Member Data Documentation

6.261.3.1 int TimeDomainStageConstraints::numberOfConstraints

numberOfConstraints gives the number of constraints contained in this stage

Definition at line 2007 of file OSInstance.h.

6.261.3.2 int TimeDomainStageConstraints::startIdx

startIdx gives the number of the first constraint contained in this stage

Definition at line 2010 of file OSInstance.h.

6.261.3.3 TimeDomainStageCon** TimeDomainStageConstraints::con

con is a pointer to an array of [TimeDomainStageCon](#) object pointers

Definition at line 2013 of file OSInstance.h.

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

- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.262 TimeDomainStageObj Class Reference

The in-memory representation of the **<obj>** element.

```
#include <OSInstance.h>
```

Public Member Functions

- [TimeDomainStageObj \(\)](#)
The [TimeDomainStageObj](#) class constructor.
- [~TimeDomainStageObj \(\)](#)
The [TimeDomainStageObj](#) class destructor.

Public Attributes

- [int idx](#)
idx gives the index of this variable

6.262.1 Detailed Description

The in-memory representation of the <**obj**> element.
Definition at line 2021 of file OSInstance.h.

6.262.2 Constructor & Destructor Documentation

6.262.2.1 TimeDomainStageObj::TimeDomainStageObj ()

The [TimeDomainStageObj](#) class constructor.

6.262.2.2 TimeDomainStageObj::~~TimeDomainStageObj ()

The [TimeDomainStageObj](#) class destructor.

6.262.3 Member Data Documentation

6.262.3.1 int TimeDomainStageObj::idx

idx gives the index of this variable

Definition at line 2032 of file OSInstance.h.

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

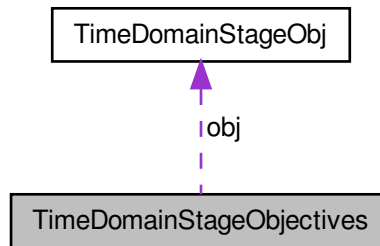
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.263 TimeDomainStageObjectives Class Reference

The in-memory representation of the <**objectives**> child of the <stage> element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomainStageObjectives:



Public Member Functions

- [TimeDomainStageObjectives](#) ()
The *TimeDomainStageObjectives* class constructor.
- [~TimeDomainStageObjectives](#) ()
The *TimeDomainStageObjectives* class destructor.

Public Attributes

- int [numberOfObjectives](#)
numberOfObjectives gives the number of objectives contained in this stage
- int [startIdx](#)
startIdx gives the number of the first objective contained in this stage
- [TimeDomainStageObj](#) ** [obj](#)
obj is a pointer to an array of *TimeDomainStageObj* object pointers

6.263.1 Detailed Description

The in-memory representation of the <**objectives**> child of the <stage> element.
Definition at line 2039 of file OSInstance.h.

6.263.2 Constructor & Destructor Documentation

6.263.2.1 TimeDomainStageObjectives::TimeDomainStageObjectives ()

The [TimeDomainStageObjectives](#) class constructor.

6.263.2.2 TimeDomainStageObjectives::~~TimeDomainStageObjectives ()

The [TimeDomainStageObjectives](#) class destructor.

6.263.3 Member Data Documentation

6.263.3.1 int TimeDomainStageObjectives::numberOfObjectives

numberOfObjectives gives the number of objectives contained in this stage

Definition at line 2050 of file OSInstance.h.

6.263.3.2 int TimeDomainStageObjectives::startIdx

startIdx gives the number of the first objective contained in this stage

Definition at line 2053 of file OSInstance.h.

6.263.3.3 TimeDomainStageObj** TimeDomainStageObjectives::obj

obj is a pointer to an array of [TimeDomainStageObj](#) object pointers

Definition at line 2056 of file OSInstance.h.

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

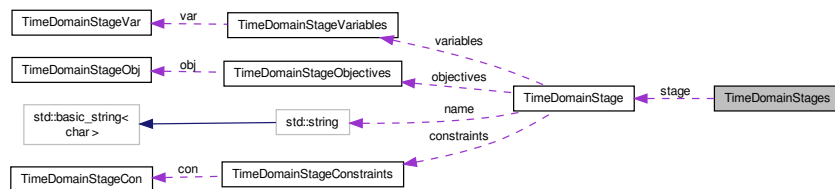
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.264 TimeDomainStages Class Reference

The in-memory representation of the **<stages>** element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomainStages:



Public Member Functions

- [TimeDomainStages \(\)](#)
The Stages class constructor.
- [~TimeDomainStages \(\)](#)
The Stages class destructor.

Public Attributes

- int [numberOfStages](#)
*numberOfStages is the number of stages in the **<stages>** element.*
- [TimeDomainStage ** stage](#)
stage is pointer to an array of stage object pointers

6.264.1 Detailed Description

The in-memory representation of the `<stages>` element.

Definition at line 2093 of file `OSInstance.h`.

6.264.2 Constructor & Destructor Documentation

6.264.2.1 TimeDomainStages::TimeDomainStages ()

The Stages class constructor.

6.264.2.2 TimeDomainStages::~~TimeDomainStages ()

The Stages class destructor.

6.264.3 Member Data Documentation

6.264.3.1 int TimeDomainStages::numberOfStages

numberOfStages is the number of stages in the `<stages>` element.

Definition at line 2106 of file `OSInstance.h`.

6.264.3.2 TimeDomainStage** TimeDomainStages::stage

stage is pointer to an array of stage object pointers

Definition at line 2109 of file `OSInstance.h`.

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

- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.265 TimeDomainStageVar Class Reference

The in-memory representation of the *element*.

```
#include <OSInstance.h>
```

Public Member Functions

- [TimeDomainStageVar \(\)](#)
The TimeDomainStageVar class constructor.
- [~TimeDomainStageVar \(\)](#)
The TimeDomainStageVar class destructor.

Public Attributes

- [int idx](#)
idx gives the index of this variable

6.265.1 Detailed Description

The in-memory representation of the *element*.

Definition at line 1935 of file OSInstance.h.

6.265.2 Constructor & Destructor Documentation

6.265.2.1 TimeDomainStageVar::TimeDomainStageVar ()

The [TimeDomainStageVar](#) class constructor.

6.265.2.2 TimeDomainStageVar::~TimeDomainStageVar ()

The [TimeDomainStageVar](#) class destructor.

6.265.3 Member Data Documentation

6.265.3.1 int TimeDomainStageVar::idx

idx gives the index of this variable

Definition at line 1946 of file OSInstance.h.

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

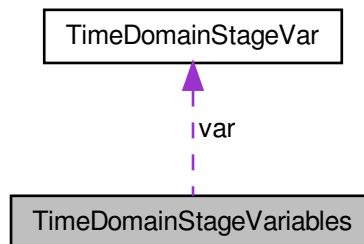
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h](#)

6.266 TimeDomainStageVariables Class Reference

The in-memory representation of the **<variables>** child of the **<stage>** element.

```
#include <OSInstance.h>
```

Collaboration diagram for TimeDomainStageVariables:



Public Member Functions

- [TimeDomainStageVariables](#) ()
The [TimeDomainStageVariables](#) class constructor.
- [~TimeDomainStageVariables](#) ()
The [TimeDomainStageVariables](#) class destructor.

Public Attributes

- int [numberOfVariables](#)
numberOfVariables gives the number of variables contained in this stage
- int [startIdx](#)
startIdx gives the number of the first variable contained in this stage
- [TimeDomainStageVar](#) ** [var](#)
var is a pointer to an array of [TimeDomainStageVar](#) object pointers

6.266.1 Detailed Description

The in-memory representation of the <**variables**> child of the <stage> element.

Definition at line 1953 of file OSInstance.h.

6.266.2 Constructor & Destructor Documentation

6.266.2.1 TimeDomainStageVariables::TimeDomainStageVariables ()

The [TimeDomainStageVariables](#) class constructor.

6.266.2.2 TimeDomainStageVariables::~~TimeDomainStageVariables ()

The [TimeDomainStageVariables](#) class destructor.

6.266.3 Member Data Documentation

6.266.3.1 int TimeDomainStageVariables::numberOfVariables

numberOfVariables gives the number of variables contained in this stage

Definition at line 1964 of file OSInstance.h.

6.266.3.2 int TimeDomainStageVariables::startIdx

startIdx gives the number of the first variable contained in this stage

Definition at line 1967 of file OSInstance.h.

6.266.3.3 TimeDomainStageVar** TimeDomainStageVariables::var

var is a pointer to an array of [TimeDomainStageVar](#) object pointers

Definition at line 1970 of file OSInstance.h.

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

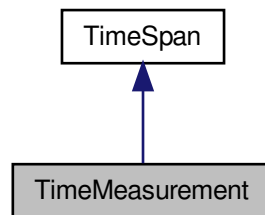
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.267 TimeMeasurement Class Reference

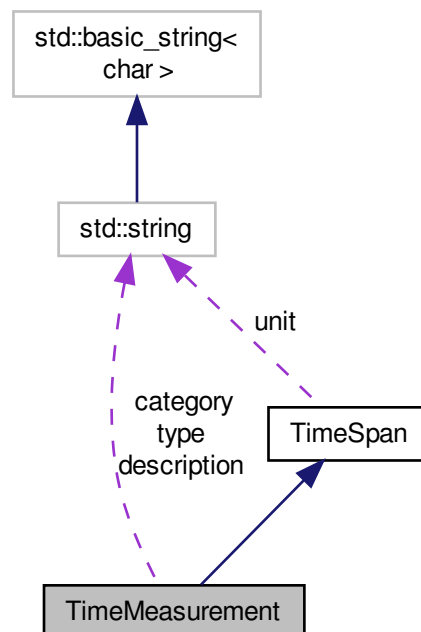
The [TimeMeasurement](#) Class.

```
#include <OSResult.h>
```

Inheritance diagram for TimeMeasurement:



Collaboration diagram for TimeMeasurement:



Public Member Functions

- [TimeMeasurement](#) ()
Default constructor.
- [~TimeMeasurement](#) ()
Class destructor.
- bool [isEqual](#) ([TimeMeasurement](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- std::string [type](#)
The type of timer used (cpuTime/elapsedTime/other)
- std::string [category](#)
The category of time (total/input/preprocessing/optimization/postprocessing/output/other)
- std::string [description](#)
Further description on the timer used.

6.267.1 Detailed Description

The [TimeMeasurement](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

A class that provides an individual time measurement as defined in the OSrL schema. Extends the class [TimeSpan](#) defined in [OSGeneral.h](#) by adding three elements type, category and description. This class supersedes the old class Time since version 2.3.

Definition at line 545 of file OSResult.h.

6.267.2 Constructor & Destructor Documentation

6.267.2.1 [TimeMeasurement::TimeMeasurement](#) ()

Default constructor.

6.267.2.2 [TimeMeasurement::~~TimeMeasurement](#) ()

Class destructor.

6.267.3 Member Function Documentation

6.267.3.1 bool [TimeMeasurement::isEqual](#) ([TimeMeasurement](#) * *that*)

A function to check for the equality of two objects.

6.267.3.2 bool TimeMeasurement::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

Reimplemented from [TimeSpan](#).

6.267.4 Member Data Documentation

6.267.4.1 std::string TimeMeasurement::type

The type of timer used (cpuTime/elapsedTime/other)

Definition at line 552 of file OSResult.h.

6.267.4.2 std::string TimeMeasurement::category

The category of time (total/input/preprocessing/optimization/postprocessing/output/other)

Definition at line 557 of file OSResult.h.

6.267.4.3 std::string TimeMeasurement::description

Further description on the timer used.

Definition at line 562 of file OSResult.h.

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

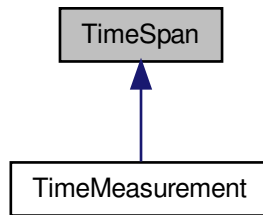
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSResult.h](#)

6.268 TimeSpan Class Reference

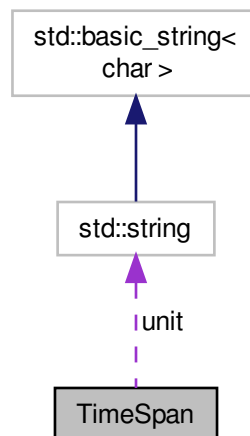
the [TimeSpan](#) class.

```
#include <OSGeneral.h>
```

Inheritance diagram for TimeSpan:



Collaboration diagram for TimeSpan:



Public Member Functions

- [TimeSpan](#) ()
Default constructor.
- [~TimeSpan](#) ()
Class destructor.
- bool [IsEqual](#) ([TimeSpan](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

- bool `deepCopyFrom` (`TimeSpan` *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- std::string `unit`
the unit in which time is measured
- double `value`
the number of units

6.268.1 Detailed Description

the `TimeSpan` class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

A data structure class that corresponds to an xml element in the OSgL schema.

Definition at line 924 of file OSGeneral.h.

6.268.2 Constructor & Destructor Documentation

6.268.2.1 `TimeSpan::TimeSpan ()`

Default constructor.

6.268.2.2 `TimeSpan::~~TimeSpan ()`

Class destructor.

6.268.3 Member Function Documentation

6.268.3.1 `bool TimeSpan::isEqual (TimeSpan * that)`

A function to check for the equality of two objects.

6.268.3.2 `bool TimeSpan::setRandom (double density, bool conformant)`

A function to make a random instance of this class.

Parameters

<i>density,:</i>	corresponds to the probability that a particular child element is created
<i>conformant,:</i>	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

Reimplemented in `TimeMeasurement`.

6.268.3.3 bool TimeSpan::deepCopyFrom (TimeSpan * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.268.4 Member Data Documentation

6.268.4.1 std::string TimeSpan::unit

the unit in which time is measured

Definition at line 929 of file OSGeneral.h.

6.268.4.2 double TimeSpan::value

the number of units

Definition at line 932 of file OSGeneral.h.

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

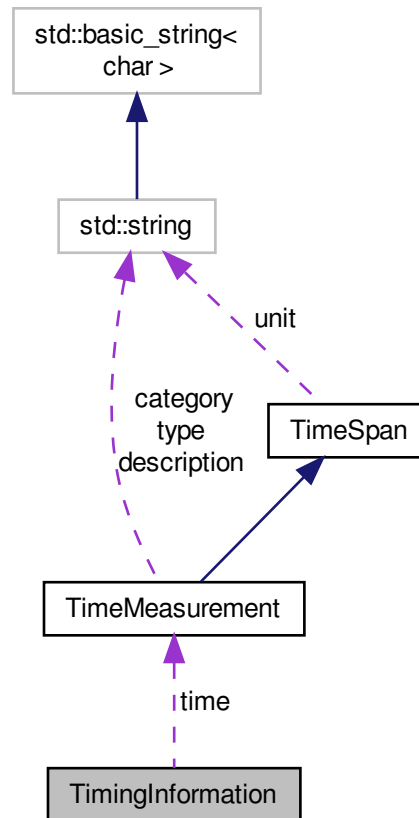
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSGeneral.h](#)

6.269 TimingInformation Class Reference

The [TimingInformation](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for TimingInformation:



Public Member Functions

- [TimingInformation](#) ()
Default constructor.
- [~TimingInformation](#) ()
Class destructor.
- `bool` [isEqual](#) ([TimingInformation](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [numberOfTimes](#)
The number of elements in the time array.

- [TimeMeasurement](#) ** *time*

An array of time measurements.

6.269.1 Detailed Description

The [TimingInformation](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that provides the timer information that is defined in the OSrL schema.

Definition at line 603 of file OSResult.h.

6.269.2 Constructor & Destructor Documentation

6.269.2.1 TimingInformation::TimingInformation ()

Default constructor.

6.269.2.2 TimingInformation::~~TimingInformation ()

Class destructor.

6.269.3 Member Function Documentation

6.269.3.1 bool TimingInformation::isEqual (TimingInformation * *that*)

A function to check for the equality of two objects.

6.269.3.2 bool TimingInformation::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.269.4 Member Data Documentation

6.269.4.1 int TimingInformation::numberOfTimes

The number of elements in the time array.

Definition at line 611 of file OSResult.h.

6.269.4.2 TimeMeasurement** TimingInformation::time

An array of time measurements.

Definition at line 617 of file OSResult.h.

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

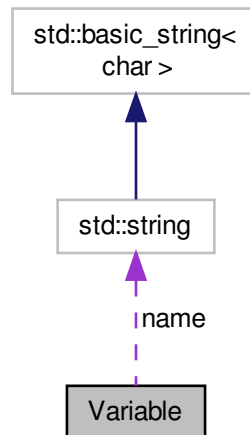
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.270 Variable Class Reference

The in-memory representation of the **variable** element.

```
#include <OSInstance.h>
```

Collaboration diagram for Variable:



Public Member Functions

- `Variable()`
The `Variable` class constructor.
- `~Variable()`
The `Variable` class destructor.
- `bool IsEqual(Variable *that)`

A function to check for the equality of two objects.

Public Attributes

- double [lb](#)
lb corresponds to the optional attribute that holds the variable lower bound.
- double [ub](#)
ub corresponds to the optional attribute that holds the variable upper bound.
- char [type](#)
type corresponds to the attribute that holds the variable type: C (Continuous), B (binary), I (general integer), or S (string).
- std::string [name](#)
name corresponds to the optional attribute that holds the variable name, the default value is empty

6.270.1 Detailed Description

The in-memory representation of the **variable** element.

Definition at line 44 of file OSInstance.h.

6.270.2 Constructor & Destructor Documentation

6.270.2.1 [Variable::Variable](#) ()

The [Variable](#) class constructor.

6.270.2.2 [Variable::~~Variable](#) ()

The [Variable](#) class destructor.

6.270.3 Member Function Documentation

6.270.3.1 [bool Variable::isEqual](#) ([Variable](#) * *that*)

A function to check for the equality of two objects.

6.270.4 Member Data Documentation

6.270.4.1 [double Variable::lb](#)

lb corresponds to the optional attribute that holds the variable lower bound.

The default value is 0

Definition at line 56 of file OSInstance.h.

6.270.4.2 [double Variable::ub](#)

ub corresponds to the optional attribute that holds the variable upper bound.

The default value is OSINFINITY

Definition at line 61 of file OSInstance.h.

6.270.4.3 char Variable::type

type corresponds to the attribute that holds the variable type: C (Continuous), B (binary), I (general integer), or S (string).
The default is C

Definition at line 66 of file OSInstance.h.

6.270.4.4 std::string Variable::name

name corresponds to the optional attribute that holds the variable name, the default value is empty

Definition at line 71 of file OSInstance.h.

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

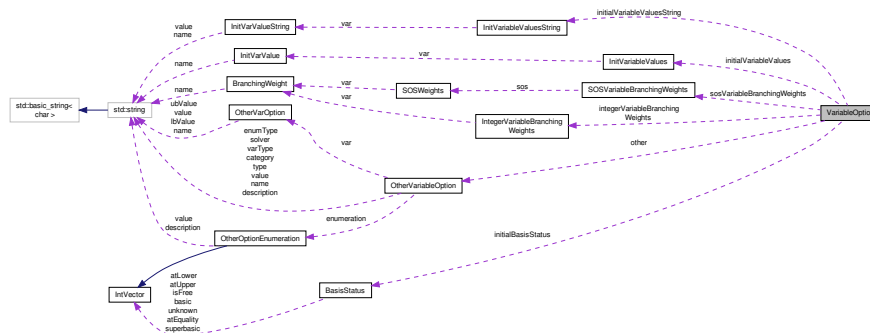
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSInstance.h

6.271 VariableOption Class Reference

the [VariableOption](#) class.

```
#include <OSOption.h>
```

Collaboration diagram for VariableOption:



Public Member Functions

- [VariableOption](#) ()
Default constructor.
- [~VariableOption](#) ()
Class destructor.
- bool [IsEqual](#) ([VariableOption](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([VariableOption](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setOther](#) (int numberOfOptions, [OtherVariableOption](#) **other)
A function to set an array of <other> elements.

- bool [addOther](#) ([OtherVariableOption](#) *other)
A function to add an <other> element.

Public Attributes

- int [numberOfOtherVariableOptions](#)
number of <other> child elements
- [InitVariableValues](#) * [initialVariableValues](#)
initial values for the variables
- [InitVariableValuesString](#) * [initialVariableValuesString](#)
initial values for string-valued variables
- [BasisStatus](#) * [initialBasisStatus](#)
initial basis information
- [IntegerVariableBranchingWeights](#) * [integerVariableBranchingWeights](#)
branching weights for integer variables
- [SOSVariableBranchingWeights](#) * [sosVariableBranchingWeights](#)
branching weights for SOS variables and groups
- [OtherVariableOption](#) ** other
other variable options

6.271.1 Detailed Description

the [VariableOption](#) class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 21/07/2008

Since

OS 1.1

Remarks

A data structure class that corresponds to an xml element in the OSoL schema.

Definition at line 2096 of file OSOption.h.

6.271.2 Constructor & Destructor Documentation

6.271.2.1 [VariableOption::VariableOption](#) ()

Default constructor.

6.271.2.2 VariableOption::~~VariableOption ()

Class destructor.

6.271.3 Member Function Documentation

6.271.3.1 bool VariableOption::isEqual (VariableOption * *that*)

A function to check for the equality of two objects.

6.271.3.2 bool VariableOption::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.271.3.3 bool VariableOption::deepCopyFrom (VariableOption * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.271.3.4 bool VariableOption::setOther (int *numberOfOptions*, OtherVariableOption ** *other*)

A function to set an array of <other> elements.

Parameters

<i>numberOfOptions</i> ,:	number of <other> elements to be set
<i>other</i> ,:	the array of <other> elements that are to be set

6.271.3.5 bool VariableOption::addOther (OtherVariableOption * *other*)

A function to add an <other> element.

Parameters

<i>other</i> ,:	the content of the <other> element to be added
-----------------	--

6.271.4 Member Data Documentation

6.271.4.1 int VariableOption::numberOfOtherVariableOptions

number of <other> child elements

Definition at line 2101 of file OOption.h.

6.271.4.2 InitVariableValues* VariableOption::initialVariableValues

initial values for the variables

Definition at line 2104 of file OOption.h.

6.271.4.3 InitVariableValuesString* VariableOption::initialVariableValuesString

initial values for string-valued variables

Definition at line 2107 of file OOption.h.

6.271.4.4 BasisStatus* VariableOption::initialBasisStatus

initial basis information

Definition at line 2110 of file OOption.h.

6.271.4.5 IntegerVariableBranchingWeights* VariableOption::integerVariableBranchingWeights

branching weights for integer variables

Definition at line 2113 of file OOption.h.

6.271.4.6 SOSVariableBranchingWeights* VariableOption::sosVariableBranchingWeights

branching weights for SOS variables and groups

Definition at line 2116 of file OOption.h.

6.271.4.7 OtherVariableOption VariableOption::other**

other variable options

Definition at line 2119 of file OOption.h.

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

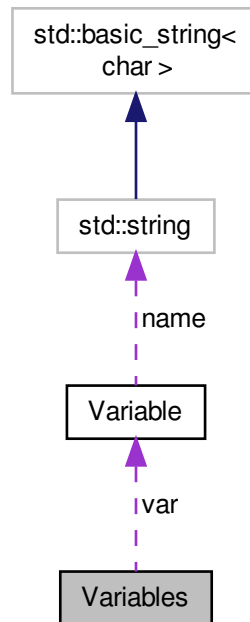
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OOption.h](#)

6.272 Variables Class Reference

The in-memory representation of the **variables** element.

```
#include <OSInstance.h>
```

Collaboration diagram for Variables:



Public Member Functions

- `Variables ()`
The `Variables` class constructor.
- `~Variables ()`
The `Variables` class destructor.
- `bool IsEqual (Variables *that)`
A function to check for the equality of two objects.

Public Attributes

- `int numberOfVariables`
numberOfVariables is the number of variables in the instance
- `Variable ** var`
Here we define a pointer to an array of var pointers.

6.272.1 Detailed Description

The in-memory representation of the **variables** element.

Definition at line 83 of file `OSInstance.h`.

6.272.2 Constructor & Destructor Documentation

6.272.2.1 Variables::Variables ()

The [Variables](#) class constructor.

6.272.2.2 Variables::~~Variables ()

The [Variables](#) class destructor.

6.272.3 Member Function Documentation

6.272.3.1 bool Variables::IsEqual (Variables * *that*)

A function to check for the equality of two objects.

6.272.4 Member Data Documentation

6.272.4.1 int Variables::numberOfVariables

numberOfVariables is the number of variables in the instance

Definition at line 94 of file OSInstance.h.

6.272.4.2 Variable** Variables::var

Here we define a pointer to an array of var pointers.

Definition at line 97 of file OSInstance.h.

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

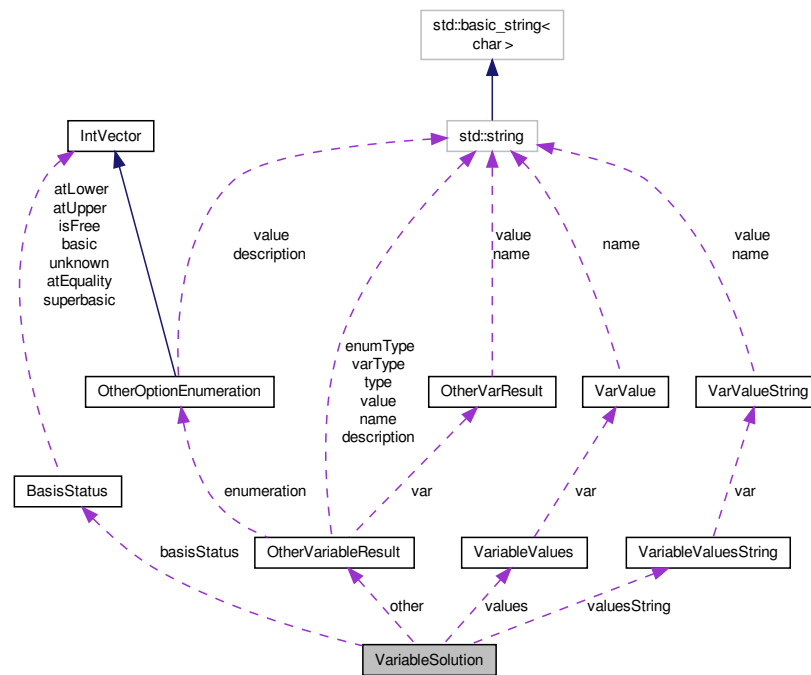
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSInstance.h](#)

6.273 VariableSolution Class Reference

The [VariableSolution](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for VariableSolution:



Public Member Functions

- `VariableSolution ()`
Default constructor.
- `~VariableSolution ()`
Class destructor.
- `bool IsEqual (VariableSolution *that)`
A function to check for the equality of two objects.
- `bool setRandom (double density, bool conformant)`
A function to make a random instance of this class.

Public Attributes

- `int numberOfOtherVariableResults`
the number of types of variable results other than the value of the variable
- `VariableValues * values`
a pointer to a `VariableValues` object
- `VariableValuesString * valuesString`
a pointer to a `VariableValuesString` object
- `BasisStatus * basisStatus`
a pointer to a `BasisStatus` object
- `OtherVariableResult ** other`
a pointer to an array of other pointer objects for variables

6.273.1 Detailed Description

The [VariableSolution](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class for reporting all of the types of solution values associated with variables.

Definition at line 1209 of file OSResult.h.

6.273.2 Constructor & Destructor Documentation

6.273.2.1 VariableSolution::VariableSolution ()

Default constructor.

6.273.2.2 VariableSolution::~~VariableSolution ()

Class destructor.

6.273.3 Member Function Documentation

6.273.3.1 bool VariableSolution::IsEqual (VariableSolution * that)

A function to check for the equality of two objects.

6.273.3.2 bool VariableSolution::setRandom (double density, bool conformant)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.273.4 Member Data Documentation

6.273.4.1 `int VariableSolution::numberOfOtherVariableResults`

the number of types of variable results other than the value of the variable

Definition at line 1216 of file OSResult.h.

6.273.4.2 `VariableValues* VariableSolution::values`

a pointer to a [VariableValues](#) object

Definition at line 1219 of file OSResult.h.

6.273.4.3 `VariableValuesString* VariableSolution::valuesString`

a pointer to a [VariableValuesString](#) object

Definition at line 1222 of file OSResult.h.

6.273.4.4 `BasisStatus* VariableSolution::basisStatus`

a pointer to a [BasisStatus](#) object

Definition at line 1225 of file OSResult.h.

6.273.4.5 `OtherVariableResult** VariableSolution::other`

a pointer to an array of other pointer objects for variables

Definition at line 1230 of file OSResult.h.

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

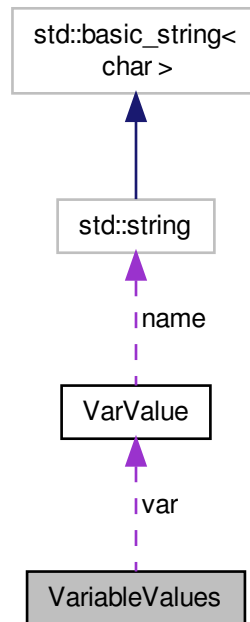
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.274 VariableValues Class Reference

The [VariableValues](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for VariableValues:



Public Member Functions

- [VariableValues](#) ()
Default constructor.
- [~VariableValues](#) ()
Class destructor.
- `bool` [IsEqual](#) ([VariableValues](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [numberOfVar](#)
the number of variable values that are in the solution
- [VarValue](#) ** [var](#)
a vector of [VarValue](#) objects, there will be one for each variable in the solution

6.274.1 Detailed Description

The [VariableValues](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that contains values for all the variables

Definition at line 899 of file OSResult.h.

6.274.2 Constructor & Destructor Documentation

6.274.2.1 VariableValues::VariableValues ()

Default constructor.

6.274.2.2 VariableValues::~~VariableValues ()

Class destructor.

6.274.3 Member Function Documentation

6.274.3.1 bool VariableValues::isEqual (VariableValues * *that*)

A function to check for the equality of two objects.

6.274.3.2 bool VariableValues::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.274.4 Member Data Documentation

6.274.4.1 int VariableValues::numberOfVar

the number of variable values that are in the solution

Definition at line 905 of file OSResult.h.

6.274.4.2 VarValue** VariableValues::var

a vector of [VarValue](#) objects, there will be one for each variable in the solution

Definition at line 910 of file OSResult.h.

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

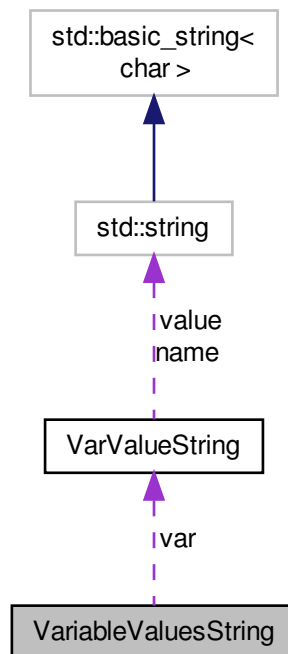
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.275 VariableValuesString Class Reference

The [VariableValuesString](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for VariableValuesString:



Public Member Functions

- [VariableValuesString](#) ()

Default constructor.

- [~VariableValuesString](#) ()

Class destructor.

- bool [isEqual](#) (VariableValuesString *that)

A function to check for the equality of two objects.

- bool [setRandom](#) (double density, bool conformant)

A function to make a random instance of this class.

Public Attributes

- int [numberOfVar](#)

the number of string-valued variable values that are in the solution

- [VarValueString](#) ** [var](#)

a vector of [VarValueString](#) objects, there will be one for each variable in the solution

6.275.1 Detailed Description

The [VariableValuesString](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that contains values for all the string-valued variables

Definition at line 1009 of file OSResult.h.

6.275.2 Constructor & Destructor Documentation

6.275.2.1 VariableValuesString::VariableValuesString ()

Default constructor.

6.275.2.2 VariableValuesString::~~VariableValuesString ()

Class destructor.

6.275.3 Member Function Documentation

6.275.3.1 bool VariableValuesString::isEqual (VariableValuesString * that)

A function to check for the equality of two objects.

6.275.3.2 bool VariableValuesString::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.275.4 Member Data Documentation

6.275.4.1 int VariableValuesString::numberOfVar

the number of string-valued variable values that are in the solution

Definition at line 1015 of file OSResult.h.

6.275.4.2 VarValueString** VariableValuesString::var

a vector of [VarValueString](#) objects, there will be one for each variable in the solution

Definition at line 1020 of file OSResult.h.

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

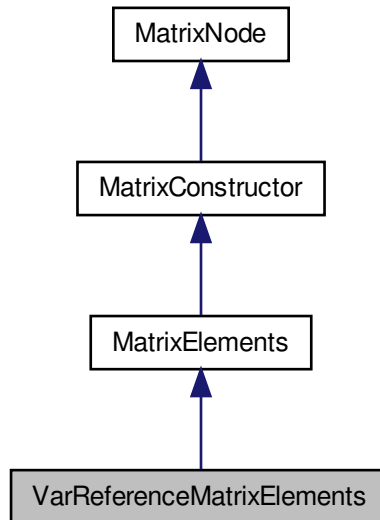
- [/home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h](#)

6.276 VarReferenceMatrixElements Class Reference

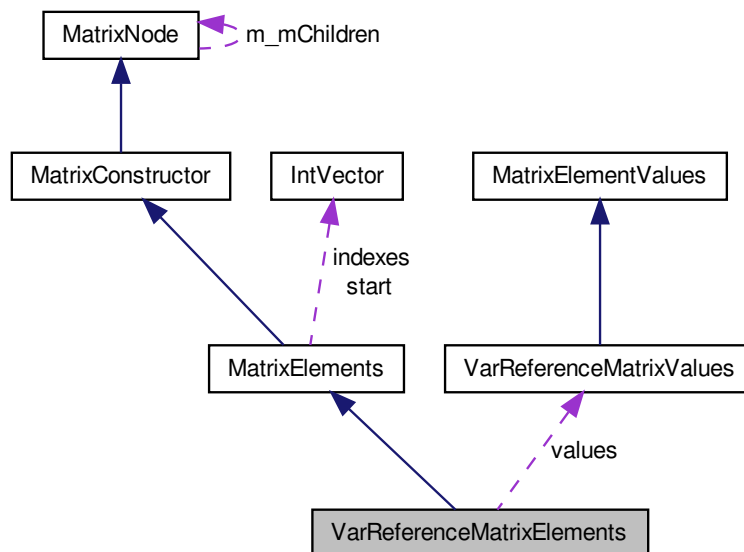
a data structure to represent variable reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a variable

```
#include <OSMatrix.h>
```

Inheritance diagram for VarReferenceMatrixElements:



Collaboration diagram for VarReferenceMatrixElements:



Public Member Functions

- [VarReferenceMatrixElements](#) ()
- [~VarReferenceMatrixElements](#) ()
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual std::string [getMatrixNodeInXML](#) ()
- virtual bool [alignsOnBlockBoundary](#) (int firstRow, int firstColumn, int nRows, int nCols)
Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.
- virtual [VarReferenceMatrixElements](#) * [cloneMatrixNode](#) ()
- bool [isEqual](#) ([VarReferenceMatrixElements](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([VarReferenceMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [VarReferenceMatrixValues](#) * [values](#)
The variable references (indexes of core variables) of the elements.

6.276.1 Detailed Description

a data structure to represent variable reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a variable

Definition at line 836 of file OSMatrix.h.

6.276.2 Constructor & Destructor Documentation

6.276.2.1 [VarReferenceMatrixElements::VarReferenceMatrixElements](#) ()

6.276.2.2 [VarReferenceMatrixElements::~~VarReferenceMatrixElements](#) ()

6.276.3 Member Function Documentation

6.276.3.1 virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [VarReferenceMatrixElements::getNodeType](#) () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

6.276.3.2 virtual **ENUM_MATRIX_TYPE** VarReferenceMatrixElements::getMatrixType () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

6.276.3.3 virtual std::string VarReferenceMatrixElements::getNodeName () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

6.276.3.4 virtual std::string VarReferenceMatrixElements::getMatrixNodeInXML () [virtual]

The following method writes a matrix node in OSgL format. it is used by OSgLWriter to write a <matrix> element.

Returns

the [MatrixNode](#) and its children as an OSgL string.

Implements [MatrixNode](#).

6.276.3.5 virtual bool VarReferenceMatrixElements::alignsOnBlockBoundary (int *firstRow*, int *firstColumn*, int *nRows*, int *nCols*) [virtual]

Check whether a submatrix aligns with the block partition of a matrix or block or other constructor.

Parameters

<i>firstRow</i>	gives the number of the first row in the submatrix (zero-based)
<i>firstColumn</i>	gives the number of the first column in the submatrix (zero-based)
<i>nRows</i>	gives the number of rows in the submatrix
<i>nColumns</i>	gives the number of columns in the submatrix

Returns

true if the submatrix aligns with the boundaries of a block This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.276.3.6 virtual VarReferenceMatrixElements* VarReferenceMatrixElements::cloneMatrixNode () [virtual]

Create or clone a node of this type. This is an abstract method which is required to be implemented by the concrete operator nodes that derive or extend from this class.

Implements [MatrixNode](#).

6.276.3.7 bool VarReferenceMatrixElements::isEqual (VarReferenceMatrixElements * *that*)

A function to check for the equality of two objects.

6.276.3.8 bool VarReferenceMatrixElements::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

Reimplemented from [MatrixNode](#).

6.276.3.9 bool VarReferenceMatrixElements::deepCopyFrom (VarReferenceMatrixElements * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.276.4 Member Data Documentation

6.276.4.1 VarReferenceMatrixValues* VarReferenceMatrixElements::values

The variable references (indexes of core variables) of the elements.

Definition at line 840 of file OSMatrix.h.

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

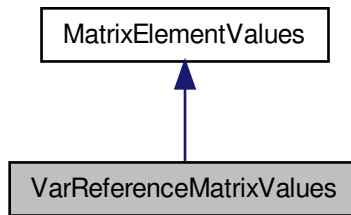
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSMatrix.h](#)

6.277 VarReferenceMatrixValues Class Reference

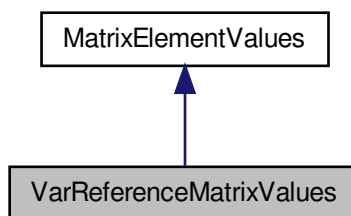
an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.

```
#include <OSMatrix.h>
```

Inheritance diagram for VarReferenceMatrixValues:



Collaboration diagram for VarReferenceMatrixValues:



Public Member Functions

- [VarReferenceMatrixValues](#) ()
- [~VarReferenceMatrixValues](#) ()
- bool [IsEqual](#) ([VarReferenceMatrixValues](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant, int iMin, int iMax)
A function to make a random instance of this class.
- bool [deepCopyFrom](#) ([VarReferenceMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int * [el](#)
Each el is a reference to a constraint defined in the <constraints> section of the OSiL file.

6.277.1 Detailed Description

an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.

Definition at line 564 of file OSMatrix.h.

6.277.2 Constructor & Destructor Documentation

6.277.2.1 VarReferenceMatrixValues::VarReferenceMatrixValues ()

6.277.2.2 VarReferenceMatrixValues::~~VarReferenceMatrixValues ()

6.277.3 Member Function Documentation

6.277.3.1 bool VarReferenceMatrixValues::isEqual (VarReferenceMatrixValues * *that*)

A function to check for the equality of two objects.

6.277.3.2 bool VarReferenceMatrixValues::setRandom (double *density*, bool *conformant*, int *iMin*, int *iMax*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)
<i>iMin</i> ,:	lowest index value (inclusive) that a variable reference in this matrix can take
<i>iMax</i> ,:	greatest index value (inclusive) that a variable reference in this matrix can take

6.277.3.3 bool VarReferenceMatrixValues::deepCopyFrom (VarReferenceMatrixValues * *that*)

A function to make a deep copy of an instance of this class.

Parameters

<i>that</i> ,:	the instance from which information is to be copied
----------------	---

Returns

whether the copy was created successfully

6.277.4 Member Data Documentation

6.277.4.1 int* VarReferenceMatrixValues::el

Each el is a reference to a constraint defined in the <constraints> section of the OSiL file.

Definition at line 570 of file OSMatrix.h.

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

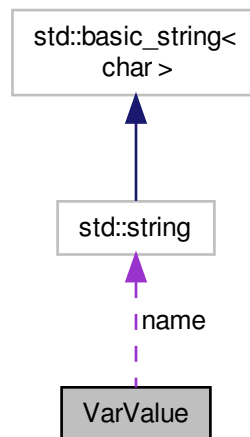
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/[OSMatrix.h](#)

6.278 VarValue Class Reference

[VarValue](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for VarValue:



Public Member Functions

- [VarValue](#) ()
Default constructor.
- [~VarValue](#) ()
Class destructor.
- `bool` [isEqual](#) ([VarValue](#) *that)
A function to check for the equality of two objects.
- `bool` [setRandom](#) (double density, `bool` conformant)
A function to make a random instance of this class.

Public Attributes

- `int` [idx](#)
idx is the index on variable in the solution
- `std::string` [name](#)
optional name
- `double` [value](#)

6.278.1 Detailed Description

[VarValue](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that is used to provide the value and index associated with the variables in the solution.

Definition at line 845 of file OSResult.h.

6.278.2 Constructor & Destructor Documentation

6.278.2.1 VarValue::VarValue ()

Default constructor.

6.278.2.2 VarValue::~~VarValue ()

Class destructor.

6.278.3 Member Function Documentation

6.278.3.1 bool VarValue::IsEqual (VarValue * *that*)

A function to check for the equality of two objects.

6.278.3.2 bool VarValue::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.278.4 Member Data Documentation

6.278.4.1 int VarValue::idx

idx is the index on variable in the solution

Definition at line 850 of file OSResult.h.

6.278.4.2 std::string VarValue::name

optional name

Definition at line 853 of file OSResult.h.

6.278.4.3 double VarValue::value

Definition at line 858 of file OSResult.h.

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

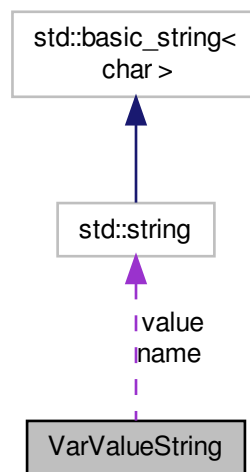
- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.279 VarValueString Class Reference

[VarValueString](#) Class.

```
#include <OSResult.h>
```

Collaboration diagram for VarValueString:



Public Member Functions

- [VarValueString](#) ()
Default constructor.

- [~VarValueString](#) ()
Class destructor.
- bool [isEqual](#) ([VarValueString](#) *that)
A function to check for the equality of two objects.
- bool [setRandom](#) (double density, bool conformant)
A function to make a random instance of this class.

Public Attributes

- int [idx](#)
idx is the index on variable in the solution
- std::string [name](#)
optional name
- std::string [value](#)

6.279.1 Detailed Description

[VarValueString](#) Class.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

A class that is used to provide the value and index associated with the string-valued variables in the solution.

Definition at line 954 of file OSResult.h.

6.279.2 Constructor & Destructor Documentation

6.279.2.1 [VarValueString::VarValueString](#) ()

Default constructor.

6.279.2.2 [VarValueString::~~VarValueString](#) ()

Class destructor.

6.279.3 Member Function Documentation

6.279.3.1 bool [VarValueString::isEqual](#) ([VarValueString](#) * that)

A function to check for the equality of two objects.

6.279.3.2 bool VarValueString::setRandom (double *density*, bool *conformant*)

A function to make a random instance of this class.

Parameters

<i>density</i> ,:	corresponds to the probability that a particular child element is created
<i>conformant</i> ,:	if true enforces side constraints not enforceable in the schema (e.g., agreement of "numberOfXX-X" attributes and <XXX> children)

6.279.4 Member Data Documentation

6.279.4.1 int VarValueString::idx

idx is the index on variable in the solution

Definition at line 959 of file OSResult.h.

6.279.4.2 std::string VarValueString::name

optional name

Definition at line 962 of file OSResult.h.

6.279.4.3 std::string VarValueString::value

Definition at line 967 of file OSResult.h.

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

- </home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSResult.h>

6.280 WSUtil Class Reference

Used by [OSSolverAgent](#) client for help in invoking a remote solver.

```
#include <OSWSUtil.h>
```

Public Member Functions

- [WSUtil](#) ()
Default constructor.
- [~WSUtil](#) ()
Class destructor.

Static Public Member Functions

- static std::string [sendSOAPMessage](#) (std::string theSOAP, std::string serviceIP, unsigned int servicePort-Number)
open a socket and send a SOAP message to the solver Web Service
- static std::string [SOAPify](#) (std::string theXmlString, bool useCDATA)
prepare XML to be put into a SOAP envelop, replace < with < replace > with > replace " and ' with
- static std::string [deSOAPify](#) (std::string theXmlString, bool useCDATA)

take the XML from a SOAP envelop and replace < with < replace > with > replace " with ";

- static std::string [createSOAPMessage](#) (int numInputs, std::string solverAddress, std::string postURI, std::string smethod, std::string *msInputs, std::string *msInputNames, std::string sSoapAction)

create the SOAP message that is send to the solver Web Service

- static std::string [createFormDataURL](#) (std::string solverAddress, std::string postURI, std::string fileName, std::string theFile, std::string boundaryName)

create the SOAP message that is sent to the solver Web Service

- static std::string [getOSxL](#) (std::string soapstring, std::string serviceMethod)

extract the appropriate OSxL protocol from the SOAP envelop

6.280.1 Detailed Description

Used by [OSSolverAgent](#) client for help in invoking a remote solver.

Remarks

The following key utilities invoked:

1. Open a TCP socket and send a message
2. Modify XML to use in a SOAP message
3. Modify the result of a SOAP message to be valid XML
4. Extract an OSxL from the SOAP

Definition at line 42 of file OSWSUtil.h.

6.280.2 Constructor & Destructor Documentation

6.280.2.1 WSUtil::WSUtil ()

Default constructor.

Parameters

<i>solverURI</i>	is the location of remote solver or scheduler
------------------	---

6.280.2.2 WSUtil::~WSUtil ()

Class destructor.

6.280.3 Member Function Documentation

6.280.3.1 static std::string WSUtil::sendSOAPMessage (std::string *theSOAP*, std::string *serviceIP*, unsigned int *servicePortNumber*) [static]

open a socket and send a SOAP message to the solver Web Service

Parameters

<i>theSOAP</i>	is a string that SOAP message sent to the Web service
<i>servIP</i>	is a string with IP address or domain name of the server
<i>solverPort- Number</i>	is a string with the port number of Web server (assume 80 by default)

Returns

the reply from the Web service in a SOAP message.

6.280.3.2 static std::string WSUtil::SOAPify (std::string *theXmlString*, bool *useCDATA*) [static]

prepare XML to be put into a SOAP envelop, replace < with < replace > with > replace " and ' with

Parameters

<i>theXmlString</i>	is the string to modify to out in the SOAP envelop
<i>useCDATA</i>	is true if just encase the XML in a CDATA statement

Returns

the XML string that goes into the SOAP envelop.

6.280.3.3 static std::string WSUtil::deSOAPify (std::string *theXmlString*, bool *useCDATA*) [static]

take the XML from a SOAP envelop and replace < with < replace > with > replace " with ";

Parameters

<i>theXmlString</i>	is the string from the SOAP envelop to modify
<i>useCDATA</i>	is true if just encasing the XML in a CDATA statement

Returns

the resulting XML string.

6.280.3.4 static std::string WSUtil::createSOAPMessage (int *numInputs*, std::string *solverAddress*, std::string *postURI*, std::string *smethod*, std::string * *msInputs*, std::string * *msInputNames*, std::string *sSoapAction*) [static]

create the SOAP message that is send to the solver Web Service

Parameters

<i>numInputs</i>	is the number of OSxL protocols (e.g. osil, osol) in the SOAP message
<i>solverAddress</i>	is the address of the scheduler or solver used
<i>postURI</i>	is the path to the solver that follows the first / in the solverAddress
<i>smethod</i>	is the method invoked, e.g. solve, kill, send, etc.
<i>msInputs</i>	is string pointer to an array of strings are the OSxL protocols protocols that go into the message, e.g. osil, osol
<i>msInputNames</i>	is a string pointer to an array of string names of the OSxL protocols
<i>sSoapAction</i>	is the name of the solver service plus the method, e.g. OSSolverService::solve

Returns

the resulting XML string that is the SOAP message.

6.280.3.5 static std::string WSUtil::createFormDataUpload (std::string *solverAddress*, std::string *postURI*, std::string *fileName*, std::string *theFile*, std::string *boundaryName*) [static]

create the SOAP message that is sent to the solver Web Service

Parameters

<i>numInputs</i>	is the number of OSxL protocols (e.g. osil, osol) in the SOAP message
<i>solverAddress</i>	is the address of the scheduler or solver used
<i>postURI</i>	is the path to the solver that follows the first / in the solverAddress
<i>smethod</i>	is the method invoked, e.g. solve, kill, send, etc.
<i>msInputs</i>	is string pointer to an array of strings are the OSxL protocols protocols that go into the message, e.g. osil, osol
<i>msInputNames</i>	is string pointer to an array of string names of the OSxL protocols
<i>sSoapAction</i>	is the name of the solver service plus the method, e.g. OSSolverService::solve

Returns

the resulting XML string that is the SOAP message.

6.280.3.6 static std::string WSUtil::getOSxL (std::string *soapstring*, std::string *serviceMethod*) [static]

extract the appropriate OSxL protocol from the SOAP envelop

Parameters

<i>soapstring</i>	the soap envelop returned from the Web service
<i>serviceMethod</i>	– extract the string between the <serviceMethodReturn> and </serviceMethodReturn> tags.

Returns

the resulting protocol.

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

- /home/ted/COIN/trunk/OS/src/OSAgent/[OSWSUtil.h](#)

6.281 YYLTYPE Struct Reference

```
#include <OSParseosil.tab.hpp>
```

Public Attributes

- int [first_line](#)
- int [first_column](#)
- int [last_line](#)
- int [last_column](#)

6.281.1 Detailed Description

Definition at line 732 of file `OSParseosil.tab.hpp`.

6.281.2 Member Data Documentation

6.281.2.1 `int YYSTYPE::first_line`

Definition at line 734 of file `OSParseosil.tab.hpp`.

6.281.2.2 `int YYSTYPE::first_column`

Definition at line 735 of file `OSParseosil.tab.hpp`.

6.281.2.3 `int YYSTYPE::last_line`

Definition at line 736 of file `OSParseosil.tab.hpp`.

6.281.2.4 `int YYSTYPE::last_column`

Definition at line 737 of file `OSParseosil.tab.hpp`.

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

- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosil.tab.hpp](#)
- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosol.tab.hpp](#)
- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosrl.tab.hpp](#)

6.282 YYSTYPE Union Reference

```
#include <OSParseosil.tab.hpp>
```

Public Attributes

- `double dval`
- `int ival`
- `char * sval`

6.282.1 Detailed Description

Definition at line 712 of file `OSParseosil.tab.hpp`.

6.282.2 Member Data Documentation

6.282.2.1 `double YYSTYPE::dval`

Definition at line 716 of file `OSParseosil.tab.hpp`.

6.282.2.2 `int YYSTYPE::ival`

Definition at line 717 of file `OSParseosil.tab.hpp`.

6.282.2.3 `char * YYSTYPE::sval`

Definition at line 718 of file `OSParseosil.tab.hpp`.

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

- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosil.tab.hpp](#)
- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosol.tab.hpp](#)
- [/home/ted/COIN/trunk/OS/src/OSParsers/OSParseosrl.tab.hpp](#)

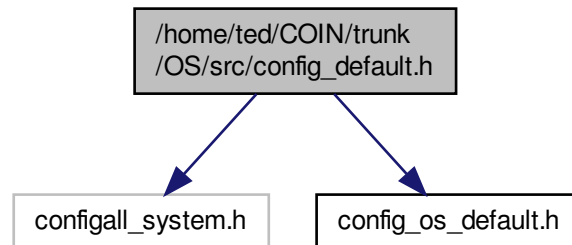
7 File Documentation

7.1 `/home/ted/COIN/trunk/OS/src/config_default.h` File Reference

```
#include "configall_system.h"
```

```
#include "config_os_default.h"
```

Include dependency graph for `config_default.h`:



Macros

- `#define COIN_OS_CHECKLEVEL 0`
- `#define COIN_OS_VERBOSITY 0`
- `#define COIN_HAS_CBC 1`
- `#define COIN_HAS_CGL 1`
- `#define COIN_HAS_CLP 1`
- `#define COIN_HAS_COINUTILS 1`
- `#define COIN_HAS_OSI 1`
- `#define COIN_HAS_SYMPHONY 1`
- `#define COIN_HAS_VOL 1`

7.1.1 Macro Definition Documentation

7.1.1.1 `#define COIN_OS_CHECKLEVEL 0`

Definition at line 14 of file `config_default.h`.

7.1.1.2 `#define COIN_OS_VERBOSITY 0`

Definition at line 17 of file config_default.h.

7.1.1.3 `#define COIN_HAS_CBC 1`

Definition at line 29 of file config_default.h.

7.1.1.4 `#define COIN_HAS_CGL 1`

Definition at line 32 of file config_default.h.

7.1.1.5 `#define COIN_HAS_CLP 1`

Definition at line 35 of file config_default.h.

7.1.1.6 `#define COIN_HAS_COINUTILS 1`

Definition at line 38 of file config_default.h.

7.1.1.7 `#define COIN_HAS_OSI 1`

Definition at line 74 of file config_default.h.

7.1.1.8 `#define COIN_HAS_SYMPHONY 1`

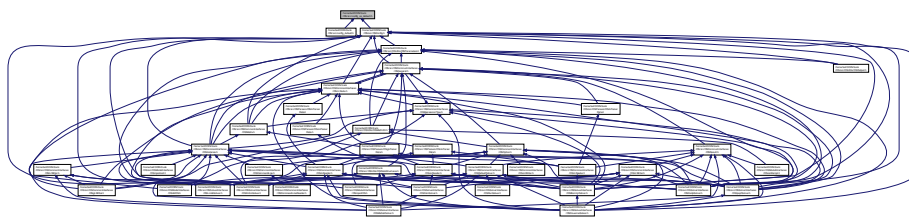
Definition at line 80 of file config_default.h.

7.1.1.9 `#define COIN_HAS_VOL 1`

Definition at line 83 of file config_default.h.

7.2 /home/ted/COIN/trunk/OS/src/config_os_default.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define OS_VERSION "trunk"`
- `#define OS_VERSION_MAJOR 9999`
- `#define OS_VERSION_MINOR 9999`
- `#define OS_VERSION_RELEASE 9999`
- `#define OS_HAS_CPPAD`
- `#define OS_HAS_IPOPT`
- `#define OS_HAS_BONMIN`

- `#define OS_HAS_COUENNE`
- `#define OS_HAS_SYMPHONY`
- `#define OS_HAS_DYLP`
- `#define OS_HAS_VOL`
- `#define OS_HAS_GLPK`
- `#define OS_HAS_ASX`

7.2.1 Macro Definition Documentation

7.2.1.1 `#define OS_VERSION "trunk"`

Definition at line 8 of file config_os_default.h.

7.2.1.2 `#define OS_VERSION_MAJOR 9999`

Definition at line 11 of file config_os_default.h.

7.2.1.3 `#define OS_VERSION_MINOR 9999`

Definition at line 14 of file config_os_default.h.

7.2.1.4 `#define OS_VERSION_RELEASE 9999`

Definition at line 17 of file config_os_default.h.

7.2.1.5 `#define OS_HAS_CPPAD`

Definition at line 20 of file config_os_default.h.

7.2.1.6 `#define OS_HAS_IPOPT`

Definition at line 23 of file config_os_default.h.

7.2.1.7 `#define OS_HAS_BONMIN`

Definition at line 26 of file config_os_default.h.

7.2.1.8 `#define OS_HAS_COUENNE`

Definition at line 29 of file config_os_default.h.

7.2.1.9 `#define OS_HAS_SYMPHONY`

Definition at line 32 of file config_os_default.h.

7.2.1.10 `#define OS_HAS_DYLP`

Definition at line 35 of file config_os_default.h.

7.2.1.11 `#define OS_HAS_VOL`

Definition at line 38 of file config_os_default.h.

7.2.1.12 `#define OS_HAS_GLPK`

Definition at line 41 of file config_os_default.h.

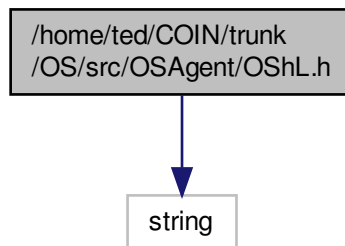
7.2.1.13 #define OS_HAS_AS_L

Definition at line 44 of file config_os_default.h.

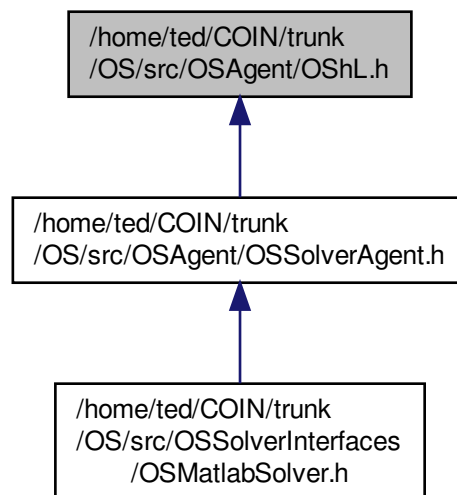
7.3 /home/ted/COIN/trunk/OS/src/OSAgent/OShL.h File Reference

```
#include <string>
```

Include dependency graph for OShL.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OShL](#)

An interface that specified virtual methods to be implemented by agents.

7.3.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

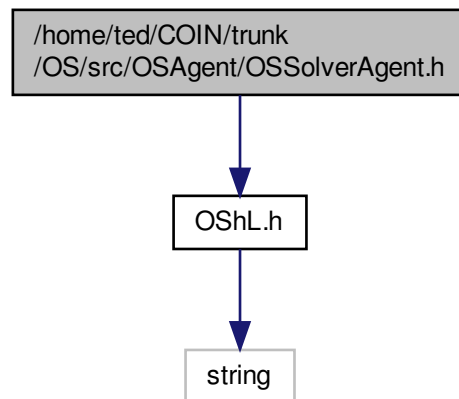
Copyright (C) 2005-2011, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OShL.h](#).

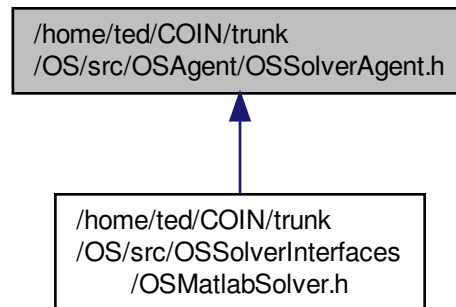
7.4 /home/ted/COIN/trunk/OS/src/OSAgent/OSSolverAgent.h File Reference

```
#include "OShL.h"
```

Include dependency graph for OSSolverAgent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSSolverAgent](#)
Used by a client to invoke a remote solver.

7.4.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

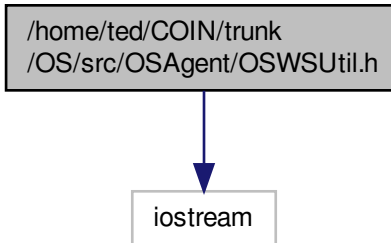
Copyright (C) 2005-2011, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSSolverAgent.h](#).

7.5 /home/ted/COIN/trunk/OS/src/OSAgent/OSWSUtil.h File Reference

```
#include <iostream>
```

Include dependency graph for OSWSUtil.h:



Classes

- class [WSUtil](#)
Used by [OSSolverAgent](#) client for help in invoking a remote solver.

Macros

- `#define RCVBUFSIZE 1024 /* Size of receive buffer */`

7.5.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2011, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSWSUtil.h](#).

7.5.2 Macro Definition Documentation

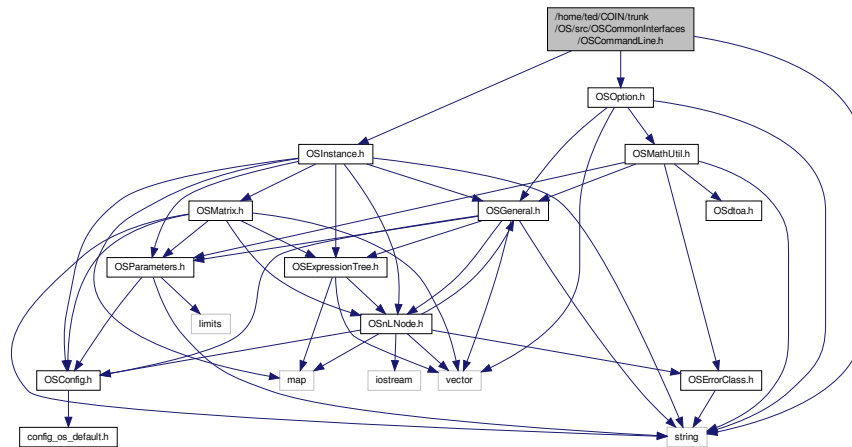
7.5.2.1 `#define RCVBUFSIZE 1024 /* Size of receive buffer */`

Definition at line 25 of file `OSWSUtil.h`.

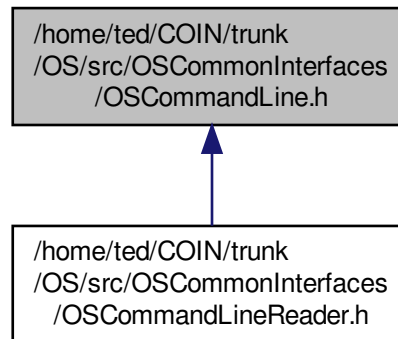
7.6 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLine.h File Reference

```
#include "OSInstance.h"
#include "OSOption.h"
#include <string>
```

Include dependency graph for OSCommandLine.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSCommandLine](#)

This class is used to store command line options for the OSSolverService executable and to provide methods to manipulate them.

7.6.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

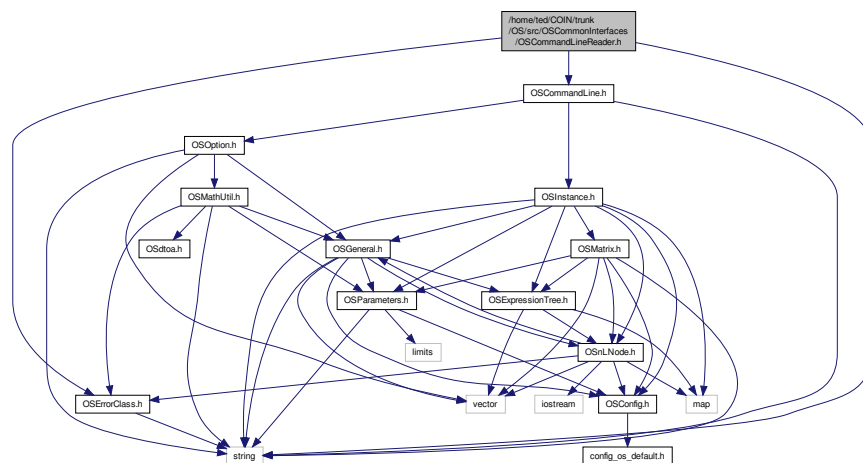
Copyright (C) 2011-2012, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSCommandLine.h](#).

7.7 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSCommandLineReader.h File Reference

```
#include "OSCommandLine.h"
#include "OSErrorClass.h"
#include <string>
```

Include dependency graph for OSCommandLineReader.h:



Classes

- class [OSCommandLineReader](#)
The *OSCommandLineReader* Class.

7.7.1 Detailed Description

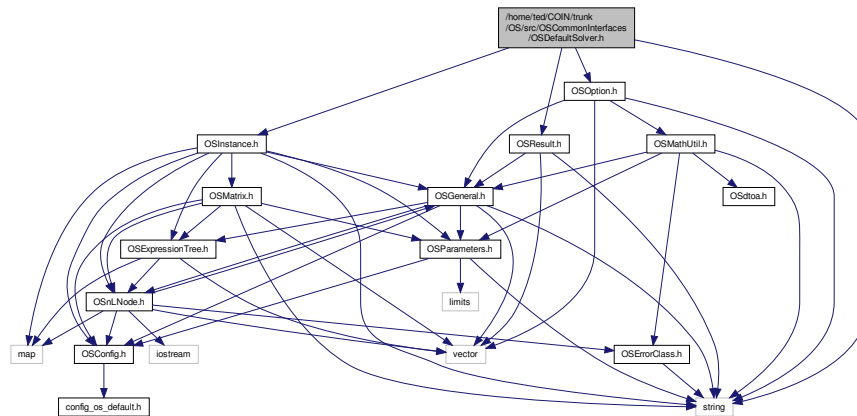
Author

Horand Gassmann, Jun Ma, Kipp Martin

Copyright (C) 2011-2013, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

7.8 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSDefaultSolver.h File Reference

Include dependency graph for OSDefaultSolver.h:

[illegible]

- class **DefaultSolver**
The Default Solver Class.

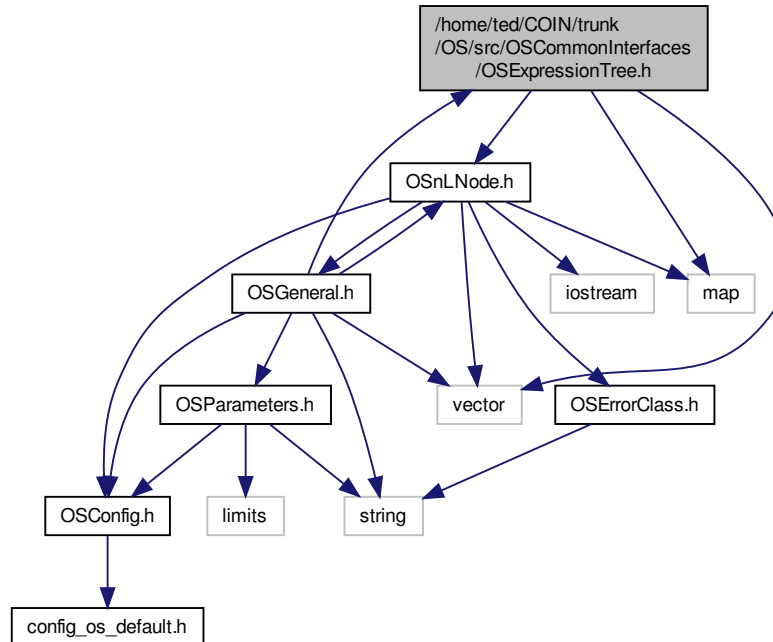
```
#include "OSnLNode.h"
```



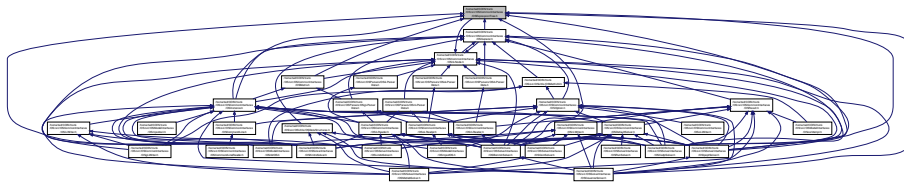
```
#include <vector>
```

```
#include <map>
```

Include dependency graph for OSExpressionTree.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSExpressionTree](#)
Used to hold the instance in memory.
- class [ScalarExpressionTree](#)
Used to hold part of the instance in memory.
- class [MatrixExpressionTree](#)
Used to hold the instance in memory.

7.9.1 Detailed Description

Classes

- class [GeneralFileHeader](#)
a data structure that holds general information about files that conform to one of the OSxL schemas
- class [SparseVector](#)
a sparse vector data structure
- class [SparseIntVector](#)
a sparse vector data structure for integer vectors
- class [SparseMatrix](#)
a sparse matrix data structure
- class [SparseJacobianMatrix](#)
a sparse Jacobian matrix data structure
- class [SparseHessianMatrix](#)
The in-memory representation of a [SparseHessianMatrix](#).
- class [QuadraticTerms](#)
a data structure for holding quadratic terms
- class [IntVector](#)
an integer Vector data structure
- class [OtherOptionEnumeration](#)
brief an integer vector data structure used in [OSOption](#) and [OSResult](#)
- class [DoubleVector](#)
a double vector data structure
- struct [IndexValuePair](#)
A commonly used structure holding an index-value pair.
- class [BasisStatus](#)
a data structure to represent an LP basis on both input and output
- class [StorageCapacity](#)
the [StorageCapacity](#) class.
- class [CPUSpeed](#)
the [CPUSpeed](#) class.
- class [CPUNumber](#)
the [CPUNumber](#) class.
- class [TimeSpan](#)
the [TimeSpan](#) class.
- class [OSGeneral](#)

Functions

- bool [OSIsEqual](#) (double x, double y)

7.10.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

- std::string [writeDbfVectorData](#) ([DoubleVector](#) *v, bool addWhiteSpace, bool writeBase64)
Take a [DoubleVector](#) object and write a string that validates against the OSgL schema.
- std::string [writeBasisStatus](#) ([BasisStatus](#) *bs, bool addWhiteSpace, bool writeBase64)
Take a [BasisStatus](#) object and write a string that validates against the OSgL schema.

7.11.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

Version

1.0, 22/Oct/2010

Since

OS2.2

Remarks

Copyright (C) 2005-2010, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSgLWriter.h](#).

7.11.2 Function Documentation

7.11.2.1 std::string writeIntVectorData ([IntVector](#) * v, bool addWhiteSpace, bool writeBase64)

Take an [IntVector](#) object and write a string that validates against the OSgL schema.

Parameters

v	is the IntVector to be output
addWhiteSpace	controls whether whitespace (i.e., line feed) is to be added
writeBase64	controls whether the IntVector is to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

Take a [DoubleVector](#) object and write a string that validates against the OSgL schema.

Parameters

v	is the DoubleVector to be output
addWhiteSpace	controls whether whitespace (i.e., line feed) is to be added
writeBase64	controls whether the IntVector is to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

Take a [BasisStatus](#) object and write a string that validates against the OSgL schema.

Parameters

<i>bs</i>	is the basisStatus object to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added
<i>writeBase64</i>	controls whether the IntVectors contained in the enumerations are to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

Take an [IntVector](#) object and write a string that validates against the OSgL schema.

Parameters

<i>v</i>	is the IntVector to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added
<i>writeBase64</i>	controls whether the IntVector is to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

7.11.2.2 std::string writeGeneralFileHeader ([GeneralFileHeader](#) * *v*, bool *addWhiteSpace*)

Take a [GeneralFileHeader](#) object and write a string that validates against the OSgL schema.

Parameters

<i>v</i>	is the header to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added

7.11.2.3 std::string writeOtherOptionEnumeration ([OtherOptionEnumeration](#) * *e*, bool *addWhiteSpace*, bool *writeBase64*)

Take an [OtherOptionEnumeration](#) object and write a string that validates against the OSgL schema.

Parameters

<i>e</i>	is the OtherOptionEnumeration to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added
<i>writeBase64</i>	controls whether the embedded integer array is to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

7.11.2.4 std::string writeDblVectorData ([DoubleVector](#) * *v*, bool *addWhiteSpace*, bool *writeBase64*)

Take a [DoubleVector](#) object and write a string that validates against the OSgL schema.

Parameters

<i>v</i>	is the DoubleVector to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added
<i>writeBase64</i>	controls whether the IntVector is to be output in base64 format or as a sequence of <el> (including mult and incr attributes)

7.11.2.5 std::string writeBasisStatus ([BasisStatus](#) * *bs*, bool *addWhiteSpace*, bool *writeBase64*)

Take a [BasisStatus](#) object and write a string that validates against the OSgL schema.

Parameters

<i>bs</i>	is the basisStatus object to be output
<i>addWhiteSpace</i>	controls whether whitespace (i.e., line feed) is to be added

<i>writeBase64</i>	controls whether the IntVectors contained in the enumerations are to be output in base64 format or as a sequence of <code><el></code> (including mult and incr attributes)
--------------------	--

```
#include "OSInstance.h"
#include "OSiLParserData.h"
#include "OSgLParserData.h"
#include "OSnLParserData.h"
#include "OSErrorClass.h"
#include <string>
```

```

graph TD
    OSInstance[OSInstance.h] --> OSMatrix[OSMatrix.h]
    OSInstance --> OSGeneral[OSGeneral.h]
    OSInstance --> OSParserData[OSParserData.h]
    OSInstance --> OSParameters[OSParameters.h]
    OSInstance --> OSExpressionTree[OSExpressionTree.h]
    OSInstance --> OSILNode[OSILNode.h]
    OSInstance --> OSILParserData[OSILParserData.h]
    OSInstance --> OSConfig[OSConfig.h]
    OSInstance --> OSErrorClass[OSErrorClass.h]
    
    OSMatrix --> OSGeneral
    OSMatrix --> OSILNode
    
    OSGeneral --> OSParserData
    OSGeneral --> OSILNode
    OSGeneral --> OSILParserData
    
    OSParserData --> stdio[stdio.h]
    OSParserData --> OSILNode
    OSParserData --> OSILParserData
    
    OSParameters --> OSILNode
    OSParameters --> OSILParserData
    
    OSExpressionTree --> OSILNode
    
    OSILNode --> limits[limits]
    OSILNode --> map[map]
    OSILNode --> istream[istream]
    OSILNode --> vector[vector]
    OSILNode --> OSILParserData
    
    OSILParserData --> OSConfig
    OSILParserData --> OSErrorClass
    
    OSConfig --> config_os_default[config_os_default.h]
    OSConfig --> string[string]
    
    OSErrorClass --> string

```

```

graph TD
    A["/home/ed/COIN/trunk  
OS/src/OSCommonInterfaces  
/OSILReader.h"]
    B["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSBommiSolver.h"]
    C["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSIpoptSolver.h"]
    D["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSCoinSolver.h"]
    E["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSCdqpSolver.h"]
    F["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSKnitroSolver.h"]
    G["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSLindoSolver.h"]
    H["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSCouenneSolver.h"]
    I["/home/ed/COIN/trunk  
/OS/src/OSSolverInterfaces  
/OSMathSolver.h"]

    B --> A
    C --> A
    D --> A
    E --> A
    F --> A
    G --> A
    H --> A
    I --> A
  
```

The diagram illustrates the COIN-OR C++ interface structure. At the top is a central node representing the main interface, which includes `/home/ed/COIN/trunk`, `OS/src/OSCommonInterfaces`, and `/OSILReader.h`. Below this central node are several other nodes, each representing a specific solver interface. These nodes are organized into two rows. The top row contains six nodes, and the bottom row contains two nodes. Each node is labeled with its file path and the interfaces it provides. Arrows point from each of these bottom nodes to the central top node, indicating that they all implement or inherit from the common interface.

- Top Row (Central Node):**
 - `/home/ed/COIN/trunk`
 - `OS/src/OSCommonInterfaces`
 - `/OSILReader.h`
- Bottom Row (Solver Interfaces):**
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSBommiSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSIpoptSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSCoinSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSCdqpSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSKnitroSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSLindoSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSCouenneSolver.h`
 - `/home/ed/COIN/trunk`, `/OS/src/OSSolverInterfaces`, `/OSMathSolver.h`

- class **OSiLReader**

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

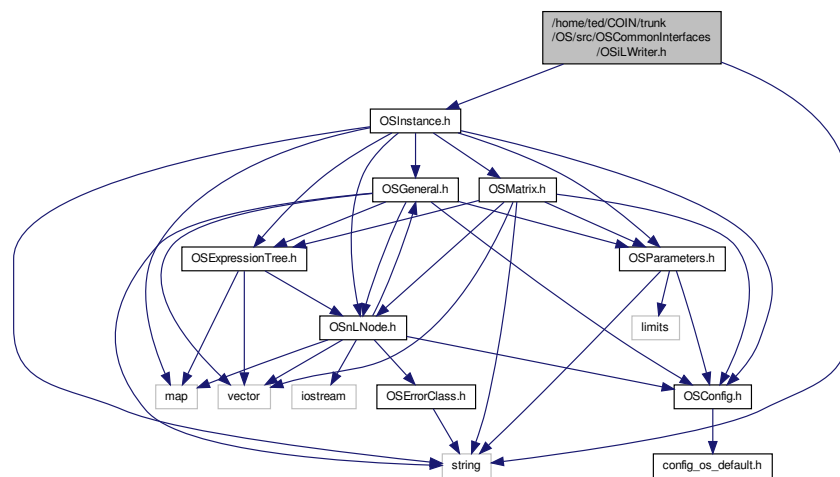
Remarks

Copyright (C) 2005-2011, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

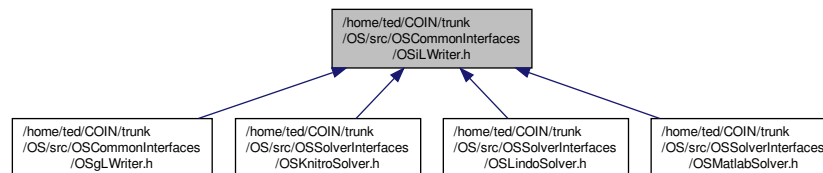
Definition in file [OSiLReader.h](#).

7.13 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSiLWriter.h File Reference

```
#include <string>
#include "OSInstance.h"
Include dependency graph for OSiLWriter.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [OSiLWriter](#)

Take an [OSInstance](#) object and write a string that validates against the OSiL schema.

Classes

- class [Variable](#)
The in-memory representation of the **variable** element.
- class [Variables](#)
The in-memory representation of the **variables** element.
- class [ObjCoef](#)
The in-memory representation of the objective function **<coef>** element.
- class [Objective](#)
The in-memory representation of the **<obj>** element.
- class [Objectives](#)
The in-memory representation of the **<objectives>** element.
- class [Constraint](#)
The in-memory representation of the **<con>** element.
- class [Constraints](#)
The in-memory representation of the **<constraints>** element.
- class [LinearConstraintCoefficients](#)
The in-memory representation of the **<linearConstraintCoefficients>** element.
- class [QuadraticTerm](#)
The in-memory representation of the **<qTerm>** element.
- class [QuadraticCoefficients](#)
The in-memory representation of the **<quadraticCoefficients>** element.
- class [NI](#)
The in-memory representation of the **<nl>** element.
- class [NonlinearExpressions](#)
The in-memory representation of the **<nonlinearExpressions>** element.
- class [Matrices](#)
The in-memory representation of the **<matrices>** element.
- class [Cone](#)
The in-memory representation of a generic cone Specific cone types are derived from this generic class.
- class [NonnegativeCone](#)
The [NonnegativeCone](#) Class.
- class [NonpositiveCone](#)
The [NonpositiveCone](#) Class.
- class [OrthantCone](#)
The [OrthantCone](#) Class.
- class [PolyhedralCone](#)
The in-memory representation of a polyhedral cone.
- class [QuadraticCone](#)
The in-memory representation of a quadratic cone.
- class [RotatedQuadraticCone](#)
The in-memory representation of a rotated quadratic cone.
- class [SemidefiniteCone](#)
The in-memory representation of a cone of semidefinite matrices.
- class [CopositiveMatricesCone](#)
The [CopositiveMatricesCone](#) Class.
- class [CompletelyPositiveMatricesCone](#)

- The *CompletelyPositiveMatricesCone* Class.
- class [ProductCone](#)
 - The in-memory representation of a product cone.
- class [IntersectionCone](#)
 - The in-memory representation of an intersection cone.
- class [DualCone](#)
 - The in-memory representation of a dual cone.
- class [PolarCone](#)
 - The in-memory representation of a polar cone.
- class [Cones](#)
 - The in-memory representation of the `<cones>` element.
- class [MatrixVar](#)
 - The in-memory representation of the `<matrixVar>` element.
- class [MatrixVariables](#)
 - The in-memory representation of the `<matrixVariables>` element.
- class [MatrixObj](#)
 - The in-memory representation of the `<matrixObj>` element.
- class [MatrixObjectives](#)
 - The in-memory representation of the `<matrixObjectives>` element.
- class [MatrixCon](#)
 - The in-memory representation of the `<matrixCon>` element.
- class [MatrixConstraints](#)
 - The in-memory representation of the `<matrixConstraints>` element.
- class [MatrixExpression](#)
 - The in-memory representation of the `<expr>` element, which is like a nonlinear expression, but since it involves matrices, the expression could be linear, so a "shape" attribute is added to distinguish linear and nonlinear expressions.
- class [MatrixExpressions](#)
 - The in-memory representation of the `<matrixExpressions>` element.
- class [MatrixProgramming](#)
 - The in-memory representation of the `<matrixProgramming>` element.
- class [TimeDomainStageVar](#)
 - The in-memory representation of the `element`.
- class [TimeDomainStageVariables](#)
 - The in-memory representation of the `<variables>` child of the `<stage>` element.
- class [TimeDomainStageCon](#)
 - The in-memory representation of the `<con>` element.
- class [TimeDomainStageConstraints](#)
 - The in-memory representation of the `<constraints>` child of the `<stage>` element.
- class [TimeDomainStageObj](#)
 - The in-memory representation of the `<obj>` element.
- class [TimeDomainStageObjectives](#)
 - The in-memory representation of the `<objectives>` child of the `<stage>` element.
- class [TimeDomainStage](#)
 - The in-memory representation of the `<stage>` element.
- class [TimeDomainStages](#)
 - The in-memory representation of the `<stages>` element.
- class [TimeDomainInterval](#)

- class [TimeDomain](#)
The in-memory representation of the <timeDomain> element.
- class [InstanceData](#)
The in-memory representation of the <instanceData> element.
- class [OSInstance](#)
The in-memory representation of an OSiL instance.

7.14.1 Detailed Description

This file defines the [OSInstance](#) class along with its supporting classes.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2012, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

1. Elements become objects of class type (the ComplexType is the class)
2. The attributes, children of the element, and text correspond to members of the class.
(Note text does not have a name and becomes .value)
3. Model groups such as choice and sequence and all correspond to arrays

Exceptions:

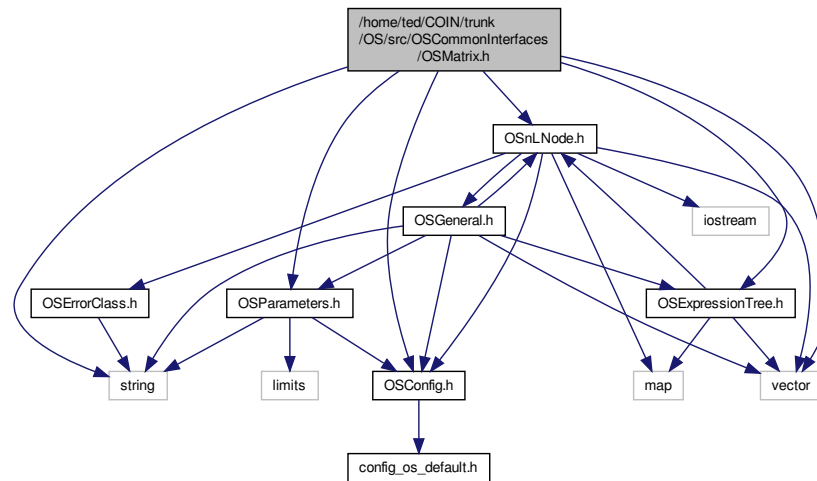
1. anything specific to XML such as base64, multi, incr does not go into classes
2. The root [OSnLNode](#) of each <nl> element is called ExpressionTree
3. Root is not called osil; it is called osinstance

Definition in file [OSInstance.h](#).

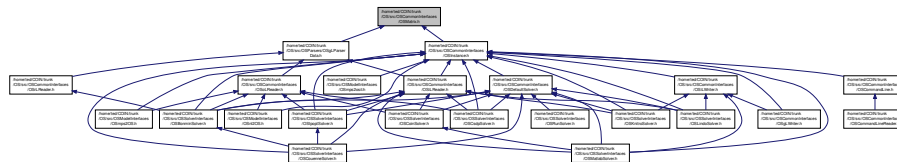
7.15 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSMatrix.h File Reference

```
#include "OSConfig.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSExpressionTree.h"
#include <string>
#include <vector>
```

Include dependency graph for OSMatrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `MatrixNode`
a generic class from which we derive matrix constructors (`BaseMatrix`, `MatrixElements`, `MatrixTransformation` and `MatrixBlocks`) as well as matrix types (`OSMatrix` and `MatrixBlock`).
- class `MatrixConstructor`
a data structure to describe one step in the construction of a matrix.
- class `MatrixElements`
an abstract class to help represent the elements in a `MatrixType` object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.
- class `MatrixElementValues`
an abstract class to help represent the elements in a `MatrixType` object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.
- class `LinearMatrixElementTerm`
a data structure to represent a term in a `linearMatrix` element A term has the form $c * x_{\{k\}}$, where c defaults to 1 and k is a valid index for a variable This is essentially an index-value pair, but with the presence of a default value
- class `LinearMatrixElement`
a data structure to represent an expression in a `linearMatrix` element A `LinearMatrixElement` is a (finite) sum of `LinearMatrixElementTerms`, with an optional additive constant

- class [ConReferenceMatrixElement](#)
a data structure to represent an entry in a [conReferenceMatrix](#) element, which consists of a constraint reference as well as a value type.
- class [ConstantMatrixValues](#)
to represent the nonzeros in a [constantMatrix](#) element
- class [VarReferenceMatrixValues](#)
an abstract class to help represent the elements in a [MatrixType](#) object From this we derive concrete classes that are used to store specific types of values, such as constant values, variable references, general nonlinear expressions, etc.
- class [LinearMatrixValues](#)
a data structure to represent the linear expressions in a [LinearMatrixElement](#) object
- class [GeneralMatrixValues](#)
a data structure to represent the nonzeros in a [generalMatrix](#) element
- class [ObjReferenceMatrixValues](#)
to represent the nonzeros in an [objReferenceMatrix](#) element
- class [ConReferenceMatrixValues](#)
a data structure to represent the nonzeros in a [conReferenceMatrix](#) element
- class [ConstantMatrixElements](#)
a data structure to represent the constant elements in a [MatrixType](#) object
- class [VarReferenceMatrixElements](#)
a data structure to represent variable reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a variable
- class [LinearMatrixElements](#)
a data structure to represent the nonzero values in a [linearMatrix](#) element
- class [GeneralMatrixElements](#)
a data structure to represent the nonzero values in a [generalMatrix](#) element
- class [ObjReferenceMatrixElements](#)
a data structure to represent objective reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of an objective (i.e., less than zero)
- class [ConReferenceMatrixElements](#)
a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element is of the form $x_{\{k\}}$ where k is the index of a constraint
- class [RowReferenceMatrixElements](#)
a data structure to represent row reference elements in a [MatrixType](#) object Each nonzero element references a row (if index is negative) or constraint (otherwise) This class allows the combining of row and constraint references in a single matrix constructor.
- class [MatrixTransformation](#)
a data structure to represent the nonzeros of a matrix by transformation from other (previously defined) matrices
- class [MatrixBlocks](#)
a data structure to represent the nonzeros of a matrix in a blockwise fashion.
- class [BaseMatrix](#)
a data structure to represent a point of departure for constructing a matrix by modifying parts of a previously defined matrix
- class [GeneralSparseMatrix](#)
a sparse matrix data structure for matrices that can hold nonconstant values
- class [ExpandedMatrixBlocks](#)
a sparse matrix data structure for matrices that can hold nonconstant values and have block structure In addition it is assumed that all nesting of blocks has been resolved.
- class [MatrixType](#)
a data structure to represent a [MatrixType](#) object (from which we derive [OSMatrix](#) and [MatrixBlock](#))
- class [OSMatrix](#)

a data structure to represent a matrix object (derived from [MatrixType](#))

- class [MatrixBlock](#)

a data structure to represent a [MatrixBlock](#) object (derived from [MatrixType](#))

7.15.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2010-2015, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

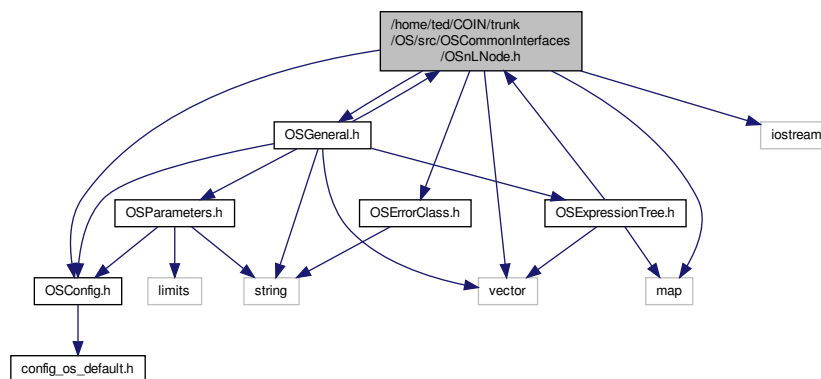
Definition in file [OSMatrix.h](#).

7.16 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSnLNode.h File Reference

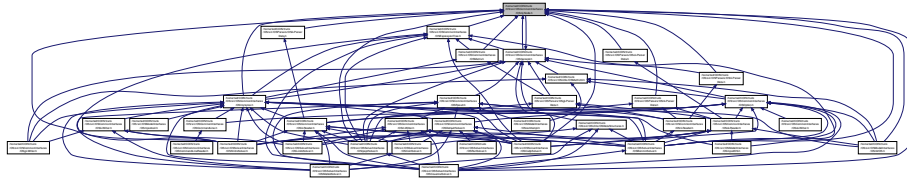
This file defines the [OSnLNode](#) class along with its derived classes.

```
#include "OSConfig.h"
#include "OSGeneral.h"
#include "OSErrorClass.h"
#include <iostream>
#include <vector>
#include <map>
```

Include dependency graph for OSnLNode.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ExprNode](#)
A generic class from which we derive both [OSnLNode](#) and [OSnLMNode](#).
- class [OSnLNode](#)
The [OSnLNode](#) Class for nonlinear expressions.
- class [OSnLNodePlus](#)
The [OSnLNodePlus](#) Class.
- class [OSnLNodeSum](#)
The [OSnLNodeSum](#) Class.
- class [OSnLNodeMax](#)
The [OSnLNodeMax](#) Class.
- class [OSnLNodeMin](#)
The [OSnLNodeMin](#) Class.
- class [OSnLNodeMinus](#)
The [OSnLNodeMinus](#) Class.
- class [OSnLNodeNegate](#)
The [OSnLNodeNegate](#) Class.
- class [OSnLNodeTimes](#)
The [OSnLNodeTimes](#) Class.
- class [OSnLNodeDivide](#)
The [OSnLNodeDivide](#) Class.
- class [OSnLNodePower](#)
The [OSnLNodePower](#) Class.
- class [OSnLNodeProduct](#)
The [OSnLNodeProduct](#) Class.
- class [OSnLNodeLn](#)
The [OSnLNodeLn](#) Class.
- class [OSnLNodeSqrt](#)
The [OSnLNodeSqrt](#) Class.
- class [OSnLNodeSquare](#)
The [OSnLNodeSquare](#) Class.
- class [OSnLNodeCos](#)
The [OSnLNodeCos](#) Class.
- class [OSnLNodeSin](#)
The [OSnLNodeSin](#) Class.
- class [OSnLNodeExp](#)
The [OSnLNodeExp](#) Class.

- class [OSnLNodeAbs](#)
The *OSnLNodeAbs* Class.
- class [OSnLNodeErf](#)
The *OSnLNodeErf* Class.
- class [OSnLNodeIf](#)
The *OSnLNodeIf* Class.
- class [OSnLNodeNumber](#)
The *OSnLNodeNumber* Class.
- class [OSnLNodeE](#)
The *OSnLNodeE* Class.
- class [OSnLNodePI](#)
The *OSnLNodePI* Class.
- class [OSnLNodeVariable](#)
The *OSnLNodeVariable* Class.
- class [OSnLNodeAllDiff](#)
The *OSnLNodeAllDiff* Class.
- class [OSnLNodeMatrixDeterminant](#)
The next few nodes evaluate to a scalar even though one or more of its arguments are matrices.
- class [OSnLNodeMatrixTrace](#)
The *OSnLNodeMatrixTrace* Class.
- class [OSnLNodeMatrixToScalar](#)
The *OSnLNodeMatrixTrace* Class.
- class [OSnLMNode](#)
The *OSnLMNode* Class for nonlinear expressions involving matrices.
- class [OSnLMNodeMatrixPlus](#)
- class [OSnLMNodeMatrixSum](#)
- class [OSnLMNodeMatrixMinus](#)
- class [OSnLMNodeMatrixNegate](#)
- class [OSnLMNodeMatrixTimes](#)
- class [OSnLMNodeMatrixInverse](#)
- class [OSnLMNodeMatrixTranspose](#)
- class [OSnLMNodeMatrixScalarTimes](#)
- class [OSnLMNodeMatrixDotTimes](#)
- class [OSnLMNodeIdentityMatrix](#)
- class [OSnLMNodeMatrixLowerTriangle](#)
- class [OSnLMNodeMatrixUpperTriangle](#)
- class [OSnLMNodeMatrixDiagonal](#)
- class [OSnLMNodeDiagonalMatrixFromVector](#)
- class [OSnLMNodeMatrixSubmatrixAt](#)
- class [OSnLMNodeMatrixReference](#)
- class [OSnLMNodeMatrixVar](#)
- class [OSnLMNodeMatrixObj](#)
- class [OSnLMNodeMatrixCon](#)
- class [OSnLMNodeMatrixProduct](#)
The *OSnLMNodeMatrixProduct* Class.

Typedefs

- typedef double [ADdouble](#)
- typedef std::vector< [ADdouble](#) > [ADvector](#)

7.16.1 Detailed Description

This file defines the [OSnLNode](#) class along with its derived classes.

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2014, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

In this file we define classes for a subset of the nodes defined in the OSnL schema. These nodes fall into two broad classes: Those that evaluate to scalar values (which inherit from [OSnLNode](#)), and those that evaluate to matrices (and inherit from [OSnLMNode](#)). OSnLNodes can have [OSnLMNode](#) children (e.g., [matrixDeterminant](#)) and vice versa (e.g., [matrixScalarTimes](#)). Both [OSnLNode](#) and [OSnLMNode](#) inherit from the base class [ExprNode](#).

Definition in file [OSnLNode.h](#).

7.16.2 Typedef Documentation

7.16.2.1 typedef double [ADdouble](#)

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

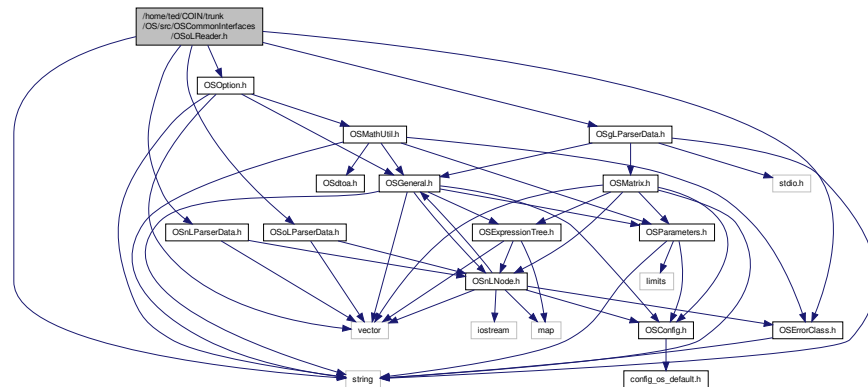
7.16.2.2 typedef std::vector<[ADdouble](#)> [ADvector](#)

Definition at line 40 of file [OSnLNode.h](#).

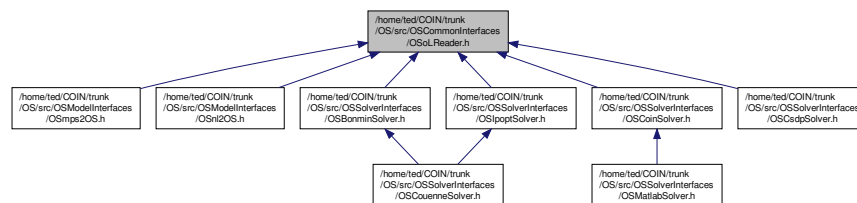
7.17 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLReader.h File Reference

```
#include <string>
#include "OSErrorClass.h"
#include "OSOption.h"
#include "OSoLParserData.h"
#include "OSgLParserData.h"
#include "OSnLParserData.h"
```

Include dependency graph for OSoLReader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSoLReader](#)
Used to read an OSoL string.

7.17.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

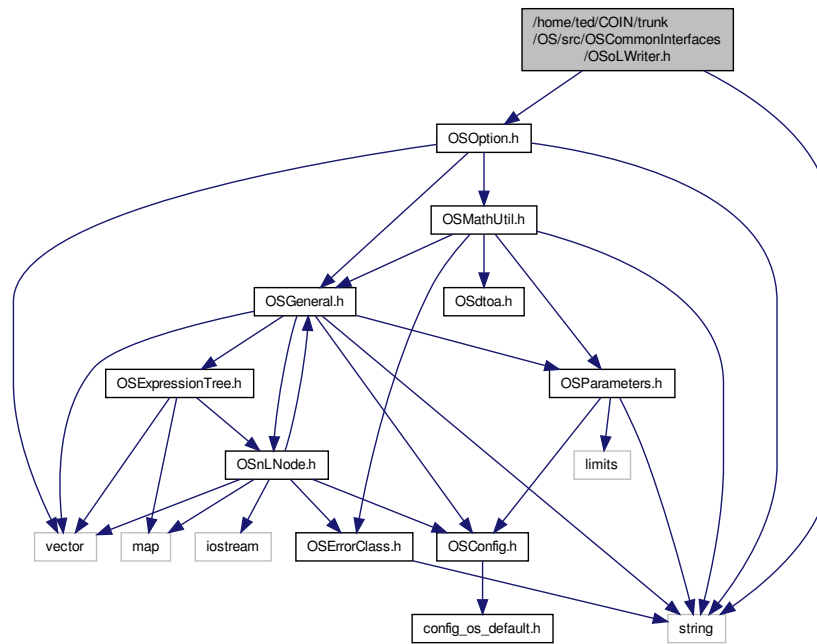
Definition in file [OSoLReader.h](#).

7.18 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSoLWriter.h File Reference

```
#include "OSOption.h"
```

```
#include <string>
```

Include dependency graph for OSoLWriter.h:



Classes

- class [OSoLWriter](#)

Take an [OSOption](#) object and write a string that validates against the OSoL schema.

7.18.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

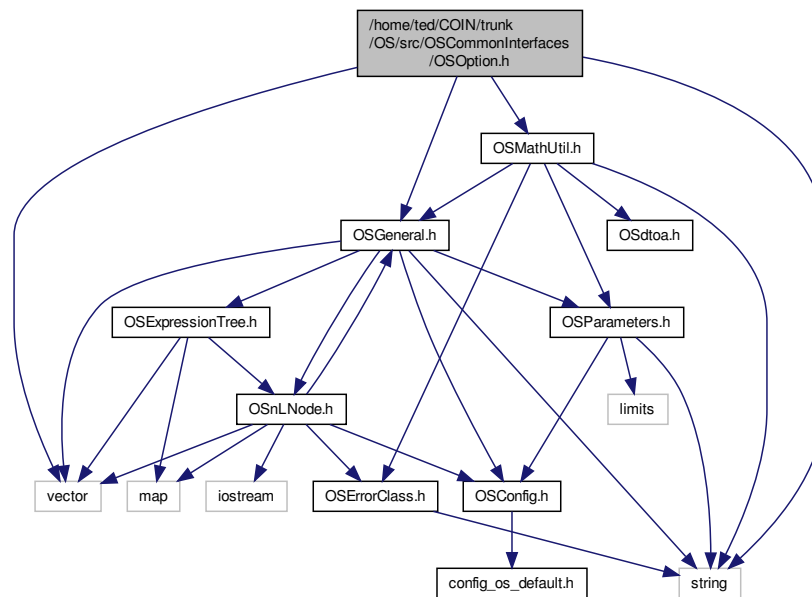
Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSoLWriter.h](#).

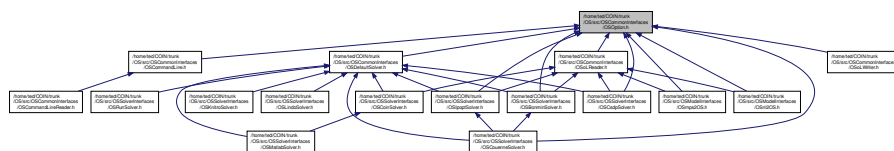
7.19 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSOption.h File Reference

```
#include <string>
#include <vector>
#include "OSGeneral.h"
#include "OSMathUtil.h"
```

Include dependency graph for OSOption.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [InstanceLocationOption](#)
the *InstanceLocationOption* class.
- class [ContactOption](#)
the *ContactOption* class.
- class [OtherOption](#)
the *OtherOption* class.
- class [OtherOptions](#)
the *OtherOptions* class.

- class [GeneralOption](#)
 The GeneralOption Class.
- class [MinDiskSpace](#)
 the MinDiskSpace class.
- class [MinMemorySize](#)
 the MinMemorySize class.
- class [MinCPUSpeed](#)
 the MinCPUSpeed class.
- class [MinCPUNumber](#)
 the MinCPUNumber class.
- class [SystemOption](#)
 the SystemOption class.
- class [ServiceOption](#)
 the ServiceOption class.
- class [MaxTime](#)
 the MaxTime class.
- class [JobDependencies](#)
 the JobDependencies class.
- class [DirectoriesAndFiles](#)
 the DirectoriesAndFiles class.
- class [PathPair](#)
 the PathPair class.
- class [PathPairs](#)
 the PathPairs class.
- class [Processes](#)
 the Processes class.
- class [JobOption](#)
 the JobOption class.
- class [InitVarValue](#)
 the InitVarValue class.
- class [InitVariableValues](#)
 the InitVariableValues class.
- class [InitVarValueString](#)
 the InitVarValueString class.
- class [InitVariableValuesString](#)
 the InitVariableValuesString class.
- class [InitBasStatus](#)
 the InitBasStatus class.
- class [InitialBasisStatus](#)
 the InitialBasisStatus class.
- class [BranchingWeight](#)
 the BranchingWeight class.
- class [IntegerVariableBranchingWeights](#)
 the IntegerVariableBranchingWeights class.
- class [SOSWeights](#)
 the SOSWeights class.
- class [SOSVariableBranchingWeights](#)

- the *SOSVariableBranchingWeights* class.
- class [OtherVarOption](#)
 - the *OtherVarOption* class.
- class [OtherVariableOption](#)
 - the *OtherVariableOption* class.
- class [VariableOption](#)
 - the *VariableOption* class.
- class [InitObjValue](#)
 - the *InitObjValue* class.
- class [InitObjectiveValues](#)
 - the *InitObjectiveValues* class.
- class [InitObjBound](#)
 - the *InitObjBound* class.
- class [InitObjectiveBounds](#)
 - the *InitObjectiveBounds* class.
- class [OtherObjOption](#)
 - the *OtherObjOption* class.
- class [OtherObjectiveOption](#)
 - the *OtherObjectiveOption* class.
- class [ObjectiveOption](#)
 - the *ObjectiveOption* class.
- class [InitConValue](#)
 - the *InitConValue* class.
- class [InitConstraintValues](#)
 - the *InitConstraintValues* class.
- class [InitDualVarValue](#)
 - the *InitDualVarValue* class.
- class [InitDualVariableValues](#)
 - the *InitDualVariableValues* class.
- class [OtherConOption](#)
 - the *OtherConOption* class.
- class [OtherConstraintOption](#)
 - the *OtherConstraintOption* class.
- class [ConstraintOption](#)
 - the *ConstraintOption* class.
- class [SolverOption](#)
 - the *SolverOption* class.
- class [SolverOptions](#)
 - the *SolverOptions* class.
- class [OptimizationOption](#)
 - the *OptimizationOption* class.
- class [OSOption](#)
 - The *Option* Class.

Classes

- struct [IndexStringPair](#)
A commonly used structure holding an index-string pair This definition is based on the definition of [IndexValuePair](#) in [OSGeneral.h](#).
- class [GeneralSubstatus](#)
The [GeneralSubstatus](#) Class.
- class [GeneralStatus](#)
The [GeneralStatus](#) Class.
- class [OtherResult](#)
The [OtherResult](#) Class.
- class [OtherResults](#)
The [OtherResults](#) Class.
- class [GeneralResult](#)
The [GeneralResult](#) Class.
- class [SystemResult](#)
The [SystemResult](#) Class.
- class [ServiceResult](#)
The [ServiceResult](#) Class.
- class [TimeMeasurement](#)
The [TimeMeasurement](#) Class.
- class [TimingInformation](#)
The [TimingInformation](#) Class.
- class [JobResult](#)
The [JobResult](#) Class.
- class [OptimizationSolutionSubstatus](#)
The [OptimizationSolutionSubstatus](#) Class.
- class [OptimizationSolutionStatus](#)
The [OptimizationSolutionStatus](#) Class.
- class [VarValue](#)
[VarValue](#) Class.
- class [VariableValues](#)
The [VariableValues](#) Class.
- class [VarValueString](#)
[VarValueString](#) Class.
- class [VariableValuesString](#)
The [VariableValuesString](#) Class.
- class [OtherVarResult](#)
[OtherVarResult](#) Class.
- class [OtherVariableResult](#)
The [OtherVariableResult](#) Class.
- class [VariableSolution](#)
The [VariableSolution](#) Class.
- class [ObjValue](#)
The [ObjValue](#) Class.
- class [ObjectiveValues](#)
The [ObjectiveValues](#) Class.
- class [OtherObjResult](#)

- The [OtherObjResult](#) Class.*
- class [OtherObjectiveResult](#)
The [OtherObjectiveResult](#) Class.
- class [ObjectiveSolution](#)
The [ObjectiveSolution](#) Class.
- class [DualVarValue](#)
The [DualVarValue](#) Class.
- class [DualVariableValues](#)
The [DualVariableValues](#) Class.
- class [OtherConResult](#)
The [OtherConResult](#) Class.
- class [OtherConstraintResult](#)
The [OtherConstraintResult](#) Class.
- class [ConstraintSolution](#)
The [ConstraintSolution](#) Class.
- class [OtherSolutionResult](#)
The [OtherSolutionResult](#) Class.
- class [OtherSolutionResults](#)
The [OtherSolutionResults](#) Class.
- class [OptimizationSolution](#)
The [OptimizationSolution](#) Class.
- class [SolverOutput](#)
The [SolverOutput](#) Class.
- class [OtherSolverOutput](#)
The [OtherSolverOutput](#) Class.
- class [OptimizationResult](#)
The [OptimizationResult](#) Class.
- class [OSResult](#)
The [Result](#) Class.

7.20.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Dalhousie University, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSResult.h](#).

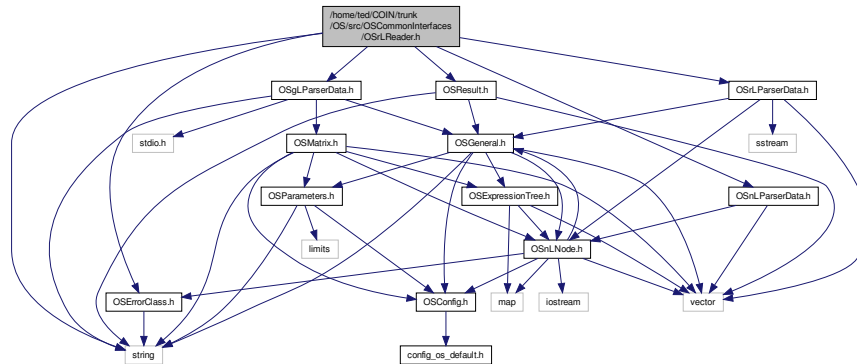
7.21 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLReader.h File Reference

```

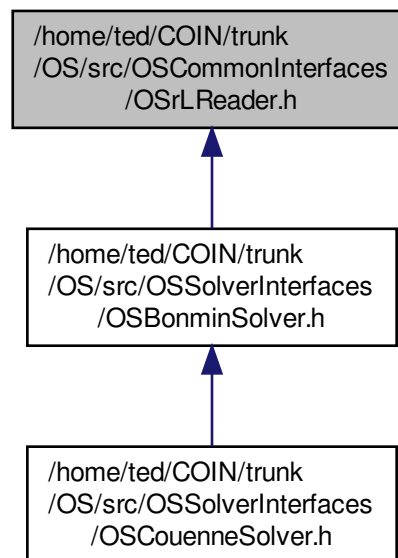
#include "OSResult.h"
#include "OSrLParserData.h"
#include "OSgLParserData.h"
#include "OSnLParserData.h"
#include "OSErrorClass.h"
#include <string>

```

Include dependency graph for OSrLReader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSrLReader](#)
The *OSrLReader* Class.

7.21.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Dalhousie University, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

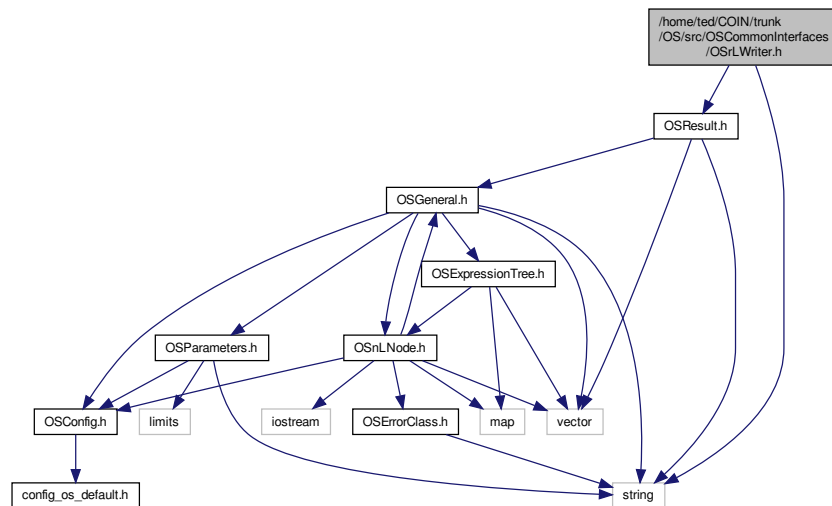
Definition in file [OSrLReader.h](#).

7.22 /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSrLWriter.h File Reference

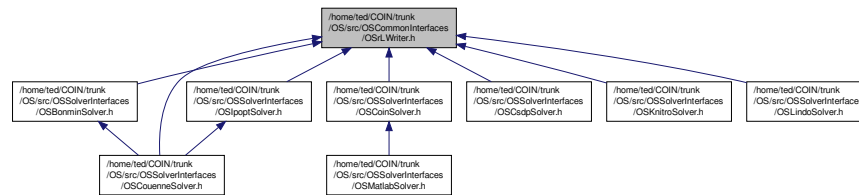
```
#include "OSResult.h"
```

```
#include <string>
```

Include dependency graph for OSrLWriter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OSrLWriter

Take an `OSResult` object and write a string that validates against `OSrL`.

7.22.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

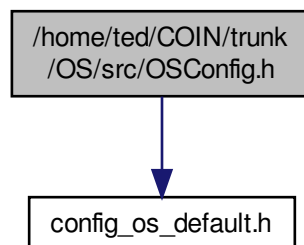
Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Dalhousie University, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSrLWriter.h](#).

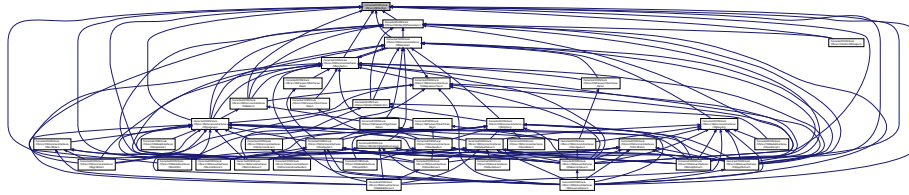
7.23 /home/ted/COIN/trunk/OS/src/OSConfig.h File Reference

```
#include "config_os_default.h"
```

Include dependency graph for OSConfig.h:



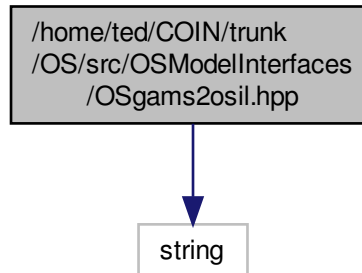
This graph shows which files directly or indirectly include this file:



7.24 /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSgams2osil.hpp File Reference

```
#include <string>
```

Include dependency graph for OSgams2osil.hpp:



Classes

- class [OSgams2osil](#)

Creating a [OSInstance](#) from a GAMS model given as GAMS Modeling Object (GMO).

7.25 /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSmps2OS.h File Reference

```
#include <CoinMpsIO.hpp>
#include <CoinPackedMatrix.hpp>
#include <string>
#include "OSInstance.h"
#include "OSOption.h"
#include "OSoLReader.h"
```

- class `OSmps2OS`
The `OSmps2OS` Class.

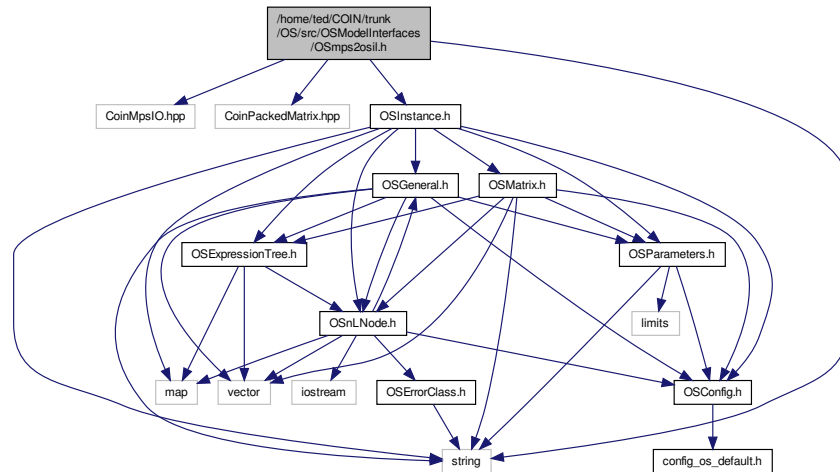
Author

Remarks

Definition in file [OSmps2OS.h](#).

```
#include <CoinMpsIO.hpp>
#include <CoinPackedMatrix.hpp>
#include <string>
#include "OSInstance.h"
```

Include dependency graph for OSmps2osil.h:



Classes

- class [OSmps2osil](#)
The *OSmps2osil* Class.

7.26.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2011, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSmps2osil.h](#).

7.27 /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSnI2OS.h File Reference

```

#include "OSoLReader.h"
#include "OSOption.h"
#include "OSInstance.h"
#include "OSnLNode.h"
#include "OSMathUtil.h"
#include <string>
#include <vector>

```


- struct `OS_AMPL_SUFFIX`
- class `OSnl2OS`

- enum `OS_AMPL_SUFFIX_TYPE` { `OS_AMPL_SUFFIX_TYPE_integer`, `OS_AMPL_SUFFIX_TYPE_double` }
- enum `OS_AMPL_SUFFIX_SCOPE` { `OS_AMPL_SUFFIX_SCOPE_variables`, `OS_AMPL_SUFFIX_SCOPE_constraints`, `OS_AMPL_SUFFIX_SCOPE_objectives`, `OS_AMPL_SUFFIX_SCOPE_problems` }
- enum `OS_AMPL_SUFFIX_DIRECTION` { `OS_AMPL_SUFFIX_DIRECTION_toSolver`, `OS_AMPL_SUFFIX_DIRECTION_toAMPL`, `OS_AMPL_SUFFIX_DIRECTION_both`, `OS_AMPL_SUFFIX_DIRECTION_local` }

OS_AMPL_SUFFIX_TYPE_integer
OS_AMPL_SUFFIX_TYPE_double

OS_AMPL_SUFFIX_SCOPE_variables
OS_AMPL_SUFFIX_SCOPE_constraints
OS_AMPL_SUFFIX_SCOPE_objectives
OS_AMPL_SUFFIX_SCOPE_problems

Generated on Mon Mar 16 2015 21:53:41 for OS by Doxygen

7.27.1.3 enum OS_AMPL_SUFFIX_DIRECTION

Enumerator:

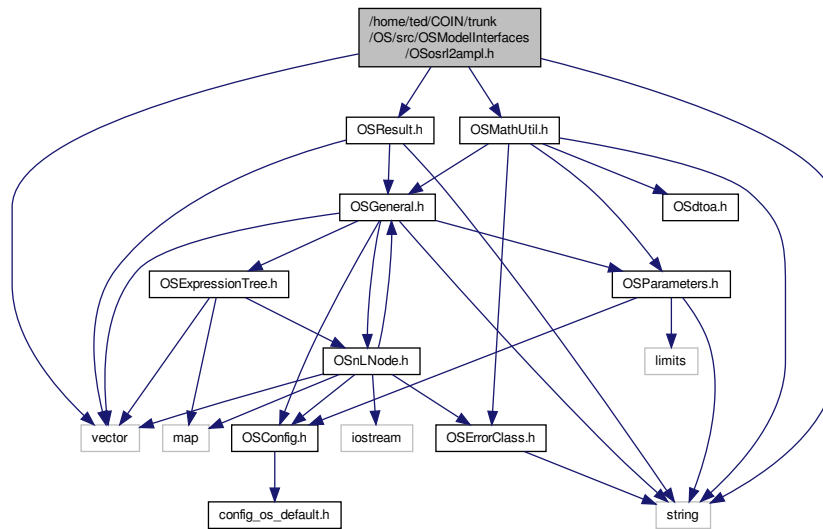
OS_AMPL_SUFFIX_DIRECTION_toSolver
OS_AMPL_SUFFIX_DIRECTION_toAMPL
OS_AMPL_SUFFIX_DIRECTION_both
OS_AMPL_SUFFIX_DIRECTION_local

Definition at line 62 of file OSnl2OS.h.

7.28 /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosl2ampl.h File Reference

```
#include "OSResult.h"
#include "OSMathUtil.h"
#include <string>
#include <vector>
```

Include dependency graph for OSosl2ampl.h:



Classes

- class [OSosl2ampl](#)
 The *OSosl2ampl* Class.

7.28.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

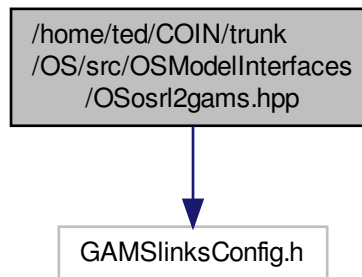
Copyright (C) 2012, Horand Gassmann, Jun Ma, Kipp Martin, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSosl2ampl.h](#).

7.29 /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSosl2gams.hpp File Reference

```
#include "GAMSLinksConfig.h"
```

Include dependency graph for OSosl2gams.hpp:



Classes

- class [OSrL2Gams](#)

Reads an optimization result and stores result and solution in a Gams Modeling Object.

7.30 /home/ted/COIN/trunk/OS/src/OSParsers/OSglParserData.h File Reference

```
#include "OSGeneral.h"
#include "OSMatrix.h"
#include <stdio.h>
#include <string>
```

```

graph TD
    Root["normalised CONTrunk  
(OS/ncv/OSParameters/OSParamData.h)"]
    Root --> Node1["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSComm.h)"]
    Root --> Node2["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommReader.h)"]
    Node1 --> Node3["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node4["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node5["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node6["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node7["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node8["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node1 --> Node9["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommSolver.h)"]
    Node2 --> Node10["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommReader.h)"]
    Node2 --> Node11["normalised CONTrunk  
(OS/ncv/OSCommInterfaces/OSCommReader.h)"]
  
```

- class `OSgLParseData`
The `OSgLParseData` Class.

- void `osgl_empty_vectors` (OSGLParserData *osglData)

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSgLParserData.h](#).

7.30.2 Function Documentation

7.30.2.1 `void osgl_empty_vectors (OSgLParserData * osglData)` `[inline]`

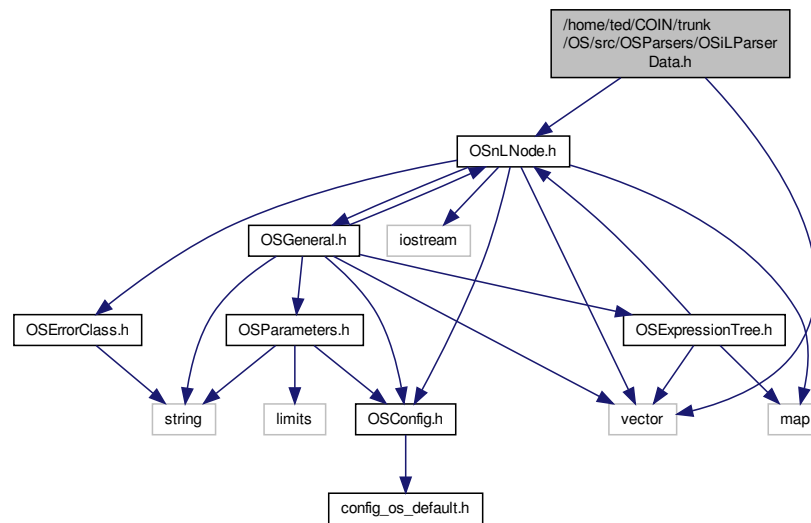
Definition at line 173 of file OSgLParserData.h.

7.31 /home/ted/COIN/trunk/OS/src/OSParsers/OSiLParserData.h File Reference

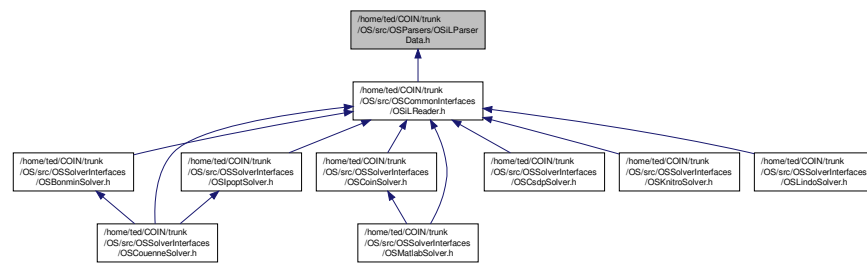
```
#include "OSnLNode.h"
```

```
#include <vector>
```

Include dependency graph for OSiLParserData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **OSiLParserData**

The *OSiLParserData* Class, used to store parser data.

7.31.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2014, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSiLParserData.h](#).

7.32 /home/ted/COIN/trunk/OS/src/OSParsers/OSnLParserData.h File Reference

```
#include "OSnLNode.h"
#include <vector>
```

```

graph TD
    A["/home/ted/COIN/trunk /OS/src/OSnLParser/OSnLParserData.h"] --> B["OSnLNode.h"]
    B --> C["OSGeneral.h"]
    B --> D["iostream"]
    B --> E["OSErrorClass.h"]
    B --> F["OSParameters.h"]
    B --> G["OSConfig.h"]
    B --> H["OSExpressionTree.h"]
    B --> I["vector"]
    B --> J["map"]
    C --> F
    C --> G
    F --> G
    G --> K["config_os_default.h"]
    H --> I
    H --> J
    I --> B
    J --> B
  
```

[illegible]

- class `OSnLParserData`
The `OSnLParserData` Class.

- void `osnl_empty_vectors` (`OSnLParserData *osnlData`)

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2014, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSnLParserData.h](#).

7.32.2 Function Documentation

7.32.2.1 `void osnl_empty_vectors (OSnLParserData * osnlData)` [inline]

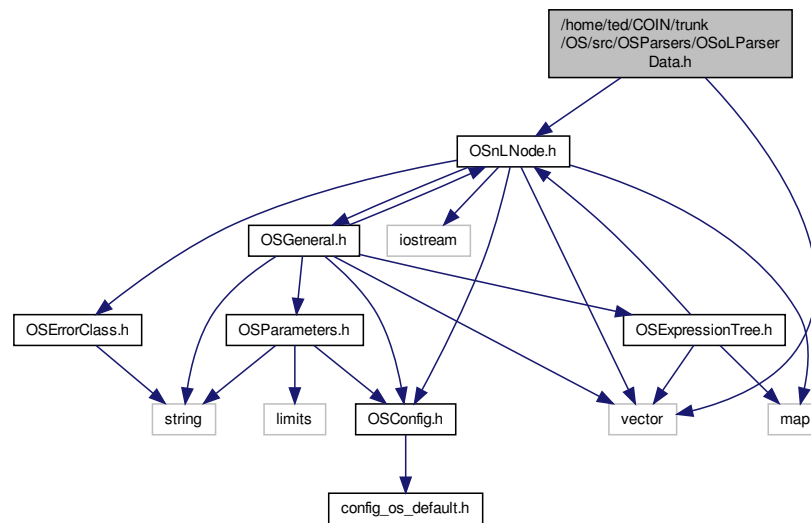
Definition at line 222 of file OSnLParserData.h.

7.33 /home/ted/COIN/trunk/OS/src/OSParsers/OSoLParserData.h File Reference

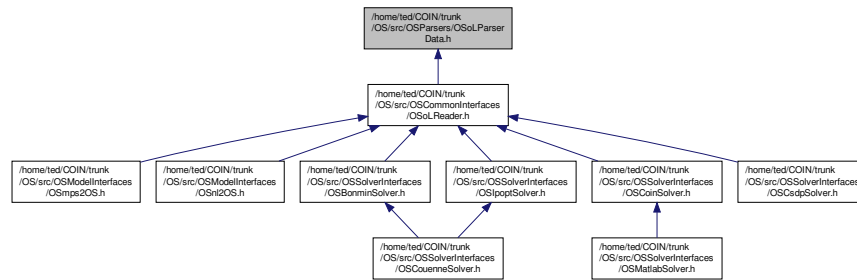
```
#include "OSnLNode.h"
```

```
#include <vector>
```

Include dependency graph for OSoLParserData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [OSoLParserData](#)
The [OSoLParserData](#) Class.

7.33.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

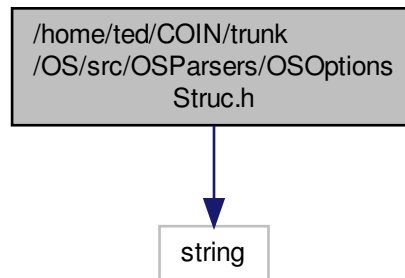
Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSoLParserData.h](#).

7.34 /home/ted/COIN/trunk/OS/src/OSParers/OSOptionsStruc.h File Reference

```
#include <string>
```

Include dependency graph for OSOptionsStruc.h:



Classes

- struct [osOptionsStruc](#)

This structure is used to store options for the OSSolverService executable.

7.34.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2012, Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSOptionsStruc.h](#).

7.35 /home/ted/COIN/trunk/OS/src/OSParsers/OSParseosil.tab.hpp File Reference

Classes

- union [YYSTYPE](#)
- struct [YYLTYPE](#)

Macros

- #define [QUOTE](#) 258
- #define [ATTRIBUTETEXT](#) 259
- #define [ELEMENTTEXT](#) 260

- #define [ITEMTEXT](#) 261
- #define [INTEGER](#) 262
- #define [DOUBLE](#) 263
- #define [TWOQUOTES](#) 264
- #define [ENDOFELEMENT](#) 265
- #define [GREATERTHAN](#) 266
- #define [OSILEND](#) 267
- #define [INSTANCEDATAEND](#) 268
- #define [INSTANCEDATASTARTEND](#) 269
- #define [EMPTYIDATT](#) 270
- #define [IDXONEATT](#) 271
- #define [IDXTWOATT](#) 272
- #define [VALUEATT](#) 273
- #define [QUADRATICCOEFFICIENTSSTART](#) 274
- #define [QUADRATICCOEFFICIENTSEND](#) 275
- #define [NUMBEROFQTERMSATT](#) 276
- #define [QTERMSTART](#) 277
- #define [QTERMEND](#) 278
- #define [MATRICESSTART](#) 279
- #define [MATRICESEND](#) 280
- #define [NUMBEROFMATRICESATT](#) 281
- #define [CONESSTART](#) 282
- #define [CONESEND](#) 283
- #define [NUMBEROFCONESATT](#) 284
- #define [NONNEGATIVECONESTART](#) 285
- #define [NONNEGATIVECONEEND](#) 286
- #define [NONPOSITIVECONESTART](#) 287
- #define [NONPOSITIVECONEEND](#) 288
- #define [ORTHANTCONESTART](#) 289
- #define [ORTHANTCONEEND](#) 290
- #define [POLYHEDRALCONESTART](#) 291
- #define [POLYHEDRALCONEEND](#) 292
- #define [QUADRATICCONESTART](#) 293
- #define [QUADRATICCONEEND](#) 294
- #define [ROTATEDQUADRATICCONESTART](#) 295
- #define [ROTATEDQUADRATICCONEEND](#) 296
- #define [SEMIDEFINITECONESTART](#) 297
- #define [SEMIDEFINITECONEEND](#) 298
- #define [PRODUCTCONESTART](#) 299
- #define [PRODUCTCONEEND](#) 300
- #define [INTERSECTIONCONESTART](#) 301
- #define [INTERSECTIONCONEEND](#) 302
- #define [DUALCONESTART](#) 303
- #define [DUALCONEEND](#) 304
- #define [POLARCONESTART](#) 305
- #define [POLARCONEEND](#) 306
- #define [DIRECTIONSTART](#) 307
- #define [DIRECTIONEND](#) 308
- #define [FACTORSSTART](#) 309
- #define [FACTORSEND](#) 310
- #define [COMPONENTSSTART](#) 311

- #define COMPONENTSEND 312
- #define NORMSCALEFACTORATT 313
- #define DISTORTIONMATRIXIDXATT 314
- #define AXISDIRECTIONATT 315
- #define FIRSTAXISDIRECTIONATT 316
- #define SECONDAXISDIRECTIONATT 317
- #define EMPTYSEMIDEFINITENESSATT 318
- #define SEMIDEFINITENESSATT 319
- #define REFERENCEMATRIXIDXATT 320
- #define MATRIXPROGRAMMINGSTART 321
- #define MATRIXPROGRAMMINGEND 322
- #define VARTYPEATT 323
- #define MATRIXVARIABLESSTART 324
- #define MATRIXVARIABLESEND 325
- #define NUMBEROFMATRIXVARATT 326
- #define MATRIXVARSTART 327
- #define MATRIXVAREND 328
- #define MATRIXOBJECTIVESSTART 329
- #define MATRIXOBJECTIVESEND 330
- #define NUMBEROFMATRIXOBJATT 331
- #define MATRIXOBJSTART 332
- #define MATRIXOBJEND 333
- #define MATRIXCONSTRAINTSSTART 334
- #define MATRIXCONSTRAINTSEND 335
- #define NUMBEROFMATRIXCONATT 336
- #define MATRIXCONSTART 337
- #define MATRIXCONEND 338
- #define MATRIXIDXATT 339
- #define LBMATRIXIDXATT 340
- #define LBCONEIDXATT 341
- #define UBMATRIXIDXATT 342
- #define UBCONEIDXATT 343
- #define TEMPLATEMATRIXIDXATT 344
- #define VARREFERENCEMATRIXIDXATT 345
- #define OBJREFERENCEMATRIXIDXATT 346
- #define CONREFERENCEMATRIXIDXATT 347
- #define ORDERCONEIDXATT 348
- #define CONSTANTMATRIXIDXATT 349
- #define TIMEDOMAINSTART 350
- #define TIMEDOMAINEND 351
- #define STAGESSTART 352
- #define STAGESEND 353
- #define STAGESTART 354
- #define STAGEEND 355
- #define NUMBEROFSTAGESATT 356
- #define HORIZONATT 357
- #define STARTATT 358
- #define VARIABLESSTART 359
- #define CONSTRAINTSSTART 360
- #define OBJECTIVESSTART 361
- #define VARIABLESEND 362

- #define [CONSTRAINTSEND](#) 363
- #define [OBJECTIVESEND](#) 364
- #define [NUMBEROFVARIABLESATT](#) 365
- #define [NUMBEROFCONSTRAINTSATT](#) 366
- #define [NUMBEROFOBJECTIVESATT](#) 367
- #define [STARTIDXATT](#) 368
- #define [VARSTART](#) 369
- #define [VAREND](#) 370
- #define [CONSTART](#) 371
- #define [CONEND](#) 372
- #define [OBJSTART](#) 373
- #define [OBJEND](#) 374
- #define [INTERVALSTART](#) 375
- #define [INTERVALEND](#) 376
- #define [HEADERSTART](#) 377
- #define [HEADEREND](#) 378
- #define [FILENAMESTART](#) 379
- #define [FILENAMEEND](#) 380
- #define [FILENAMEEMPTY](#) 381
- #define [FILENAMESTARTANDEND](#) 382
- #define [FILESOURCESTART](#) 383
- #define [FILESOURCEEND](#) 384
- #define [FILESOURCEEMPTY](#) 385
- #define [FILESOURCESTARTANDEND](#) 386
- #define [FILEDESCRIPTIONSTART](#) 387
- #define [FILEDESCRIPTIONEND](#) 388
- #define [FILEDESCRIPTIONEMPTY](#) 389
- #define [FILEDESCRIPTIONSTARTANDEND](#) 390
- #define [FILECREATORSTART](#) 391
- #define [FILECREATOREND](#) 392
- #define [FILECREATOREMPTY](#) 393
- #define [FILECREATORSTARTANDEND](#) 394
- #define [FILELICENCESTART](#) 395
- #define [FILELICENCEEND](#) 396
- #define [FILELICENCEEMPTY](#) 397
- #define [FILELICENCESTARTANDEND](#) 398
- #define [ENUMERATIONSTART](#) 399
- #define [ENUMERATIONEND](#) 400
- #define [NUMBEROFELATT](#) 401
- #define [ITEMEMPTY](#) 402
- #define [ITEMSTART](#) 403
- #define [ITEMEND](#) 404
- #define [ITEMSTARTANDEND](#) 405
- #define [BASE64START](#) 406
- #define [BASE64END](#) 407
- #define [INCRATT](#) 408
- #define [MULTATT](#) 409
- #define [SIZEOFATT](#) 410
- #define [ELSTART](#) 411
- #define [ELEND](#) 412
- #define [MATRIXSTART](#) 413

- #define [MATRIXEND](#) 414
- #define [BASEMATRIXEND](#) 415
- #define [BASEMATRIXSTART](#) 416
- #define [BLOCKSTART](#) 417
- #define [BLOCKEND](#) 418
- #define [BLOCKSSTART](#) 419
- #define [BLOCKSEND](#) 420
- #define [EMPTYNAMEATT](#) 421
- #define [NAMEATT](#) 422
- #define [EMPTYTYPEATT](#) 423
- #define [TYPEATT](#) 424
- #define [EMPTYSHAPEATT](#) 425
- #define [SHAPEATT](#) 426
- #define [EMPTYSYMMETRYATT](#) 427
- #define [SYMMETRYATT](#) 428
- #define [EMPTYNEGATIVEPATTERNATT](#) 429
- #define [NEGATIVEPATTERNATT](#) 430
- #define [CONSTANTATT](#) 431
- #define [NUMBEROFBLOCKSATT](#) 432
- #define [NUMBEROFCOLUMNSATT](#) 433
- #define [NUMBEROFROWSATT](#) 434
- #define [NUMBEROFVALUESATT](#) 435
- #define [NUMBEROFVARIDXATT](#) 436
- #define [IDXATT](#) 437
- #define [COEFATT](#) 438
- #define [BASEMATRIXIDXATT](#) 439
- #define [TARGETMATRIXFIRSTROWATT](#) 440
- #define [TARGETMATRIXFIRSTCOLATT](#) 441
- #define [BASEMATRIXSTARTROWATT](#) 442
- #define [BASEMATRIXSTARTCOLATT](#) 443
- #define [BASEMATRIXENDROWATT](#) 444
- #define [BASEMATRIXENDCOLATT](#) 445
- #define [SCALARMULTIPLIERATT](#) 446
- #define [EMPTYBASETRANSPOSEATT](#) 447
- #define [BASETRANSPOSEATT](#) 448
- #define [ELEMENTSSTART](#) 449
- #define [ELEMENTSEND](#) 450
- #define [CONSTANTELEMENTSSTART](#) 451
- #define [CONSTANTELEMENTSEND](#) 452
- #define [STARTVECTORSTART](#) 453
- #define [STARTVECTOREND](#) 454
- #define [NONZEROSSTART](#) 455
- #define [NONZEROSEND](#) 456
- #define [INDEXESSTART](#) 457
- #define [INDEXESEND](#) 458
- #define [VALUESSTART](#) 459
- #define [VALUESEND](#) 460
- #define [VARREFERENCEELEMENTSSTART](#) 461
- #define [VARREFERENCEELEMENTSEND](#) 462
- #define [LINEARELEMENTSSTART](#) 463
- #define [LINEARELEMENTSEND](#) 464

- #define [GENERALELEMENTSSTART](#) 465
- #define [GENERALELEMENTSEND](#) 466
- #define [CONREFERENCEELEMENTSSTART](#) 467
- #define [CONREFERENCEELEMENTSEND](#) 468
- #define [VALUETYPEATT](#) 469
- #define [OBJREFERENCEELEMENTSSTART](#) 470
- #define [OBJREFERENCEELEMENTSEND](#) 471
- #define [PATTERNELEMENTSSTART](#) 472
- #define [PATTERNELEMENTSEND](#) 473
- #define [VARIDXSTART](#) 474
- #define [VARIDXEND](#) 475
- #define [TRANSFORMATIONSTART](#) 476
- #define [TRANSFORMATIONEND](#) 477
- #define [COLOFFSETSSTART](#) 478
- #define [COLOFFSETSEND](#) 479
- #define [ROWOFFSETSSTART](#) 480
- #define [ROWOFFSETSEND](#) 481
- #define [EMPTYROWMAJORATT](#) 482
- #define [ROWMAJORATT](#) 483
- #define [BLOCKROWIDXATT](#) 484
- #define [BLOCKCOLIDXATT](#) 485
- #define [DUMMY](#) 486
- #define [NONLINEAREXPRESSIONSSTART](#) 487
- #define [NONLINEAREXPRESSIONSEND](#) 488
- #define [NUMBEROFNONLINEAREXPRESSIONS](#) 489
- #define [NLSTART](#) 490
- #define [NLEND](#) 491
- #define [MATRIXEXPRESSIONSSTART](#) 492
- #define [MATRIXEXPRESSIONSEND](#) 493
- #define [NUMBEROFEXPR](#) 494
- #define [EXPRSTART](#) 495
- #define [EXPREND](#) 496
- #define [NUMBEROFMATRIXTERMSATT](#) 497
- #define [MATRIXTERMSTART](#) 498
- #define [MATRIXTERMEND](#) 499
- #define [POWERSTART](#) 500
- #define [POWEREND](#) 501
- #define [PLUSSTART](#) 502
- #define [PLUSEND](#) 503
- #define [MINUSSTART](#) 504
- #define [MINUSEND](#) 505
- #define [DIVIDESTART](#) 506
- #define [DIVIDEEND](#) 507
- #define [LNSTART](#) 508
- #define [LNEND](#) 509
- #define [SQRTSTART](#) 510
- #define [SQRTEND](#) 511
- #define [SUMSTART](#) 512
- #define [SUMEND](#) 513
- #define [PRODUCTSTART](#) 514
- #define [PRODUCTEND](#) 515

- #define [EXPSTART](#) 516
- #define [EXPEND](#) 517
- #define [NEGATESTART](#) 518
- #define [NEGATEEND](#) 519
- #define [IFSTART](#) 520
- #define [IFEND](#) 521
- #define [SQUARESTART](#) 522
- #define [SQUAREEND](#) 523
- #define [COSSTART](#) 524
- #define [COSEND](#) 525
- #define [SINSTART](#) 526
- #define [SINEND](#) 527
- #define [VARIABLESTART](#) 528
- #define [VARIABLEEND](#) 529
- #define [ABSSTART](#) 530
- #define [ABSEND](#) 531
- #define [ERFSTART](#) 532
- #define [ERFEND](#) 533
- #define [MAXSTART](#) 534
- #define [MAXEND](#) 535
- #define [ALLDIFFSTART](#) 536
- #define [ALLDIFFEND](#) 537
- #define [MINSTART](#) 538
- #define [MINEND](#) 539
- #define [ESTART](#) 540
- #define [EEND](#) 541
- #define [PISTART](#) 542
- #define [PIEND](#) 543
- #define [TIMESSTART](#) 544
- #define [TIMESEND](#) 545
- #define [NUMBERSTART](#) 546
- #define [NUMBEREND](#) 547
- #define [MATRIXDETERMINANTSTART](#) 548
- #define [MATRIXDETERMINANTEND](#) 549
- #define [MATRIXTRACESTART](#) 550
- #define [MATRIXTRACEEND](#) 551
- #define [MATRIXTOSCALARSTART](#) 552
- #define [MATRIXTOSCALAREND](#) 553
- #define [MATRIXDIAGONALSTART](#) 554
- #define [MATRIXDIAGONALEND](#) 555
- #define [MATRIXDOTTIMESSTART](#) 556
- #define [MATRIXDOTTIMESEND](#) 557
- #define [MATRIXLOWERTRIANGLESTART](#) 558
- #define [MATRIXLOWERTRIANGLEEND](#) 559
- #define [MATRIXUPPERTRIANGLESTART](#) 560
- #define [MATRIXUPPERTRIANGLEEND](#) 561
- #define [MATRIXMERGESTART](#) 562
- #define [MATRIXMERGEEND](#) 563
- #define [MATRIXMINUSSTART](#) 564
- #define [MATRIXMINUSEND](#) 565
- #define [MATRIXNEGATESTART](#) 566

- #define [MATRIXNEGATEEND](#) 567
- #define [MATRIXPLUSSTART](#) 568
- #define [MATRIXPLUSEND](#) 569
- #define [MATRIXTIMESSTART](#) 570
- #define [MATRIXTIMSEEND](#) 571
- #define [MATRIXPRODUCTSTART](#) 572
- #define [MATRIXPRODUCTEND](#) 573
- #define [MATRIXSCALARTIMESSTART](#) 574
- #define [MATRIXSCALARTIMSEEND](#) 575
- #define [MATRIXSUBMATRIXATSTART](#) 576
- #define [MATRIXSUBMATRIXATEND](#) 577
- #define [MATRIXTRANSPOSESTART](#) 578
- #define [MATRIXTRANSPOSEEND](#) 579
- #define [MATRIXREFERENCESTART](#) 580
- #define [MATRIXREFERENCEEND](#) 581
- #define [IDENTITYMATRIXSTART](#) 582
- #define [IDENTITYMATRIXEND](#) 583
- #define [MATRIXINVERSESTART](#) 584
- #define [MATRIXINVERSEEND](#) 585
- #define [EMPTYINCLUDEDIAGONALATT](#) 586
- #define [INCLUDEDIAGONALATT](#) 587
- #define [IDATT](#) 588
- #define [YYSTYPE_IS_TRIVIAL](#) 1
- #define [yystype YYSYPE](#) /* obsolescent; will be withdrawn */
- #define [YYSTYPE_IS_DECLARED](#) 1
- #define [yyltype YYLTYPE](#) /* obsolescent; will be withdrawn */
- #define [YYLTYPE_IS_DECLARED](#) 1
- #define [YYLTYPE_IS_TRIVIAL](#) 1

Typedefs

- typedef union [YYSTYPE](#) YYSYPE
- typedef struct [YYLTYPE](#) YYLTYPE

Enumerations

- enum [yytokentype](#) {
[QUOTE](#) = 258, [ATTRIBUTETEXT](#) = 259, [ELEMENTTEXT](#) = 260, [ITEMTEXT](#) = 261,
[INTEGER](#) = 262, [DOUBLE](#) = 263, [TWOQUOTES](#) = 264, [ENDOFELEMENT](#) = 265,
[GREATERTHAN](#) = 266, [OSILEND](#) = 267, [INSTANCEDATAEND](#) = 268, [INSTANCEDATASTARTEND](#) = 269,
[EMPTYIDATT](#) = 270, [IDXONEATT](#) = 271, [IDXTWOATT](#) = 272, [VALUEATT](#) = 273,
[QUADRATICCOEFFICIENTSSTART](#) = 274, [QUADRATICCOEFFICIENTSEND](#) = 275, [NUMBEROFQTERMSA-](#)
[TT](#) = 276, [QTERMSTART](#) = 277,
[QTERMEND](#) = 278, [MATRICESSTART](#) = 279, [MATRICESEND](#) = 280, [NUMBEROFMATRICESATT](#) = 281,
[CONESSTART](#) = 282, [CONESEND](#) = 283, [NUMBEROFCONESATT](#) = 284, [NONNEGATIVECONESTART](#) =

285,
 NONNEGATIVECONEEND = 286, NONPOSITIVECONESTART = 287, NONPOSITIVECONEEND = 288, ORTHANTCONESTART = 289,
 ORTHANTCONEEND = 290, POLYHEDRALCONESTART = 291, POLYHEDRALCONEEND = 292, QUADRATICCONESTART = 293,
 QUADRATICCONEEND = 294, ROTATEDQUADRATICCONESTART = 295, ROTATEDQUADRATICCONEEND = 296, SEMIDEFINITECONESTART = 297,
 SEMIDEFINITECONEEND = 298, PRODUCTCONESTART = 299, PRODUCTCONEEND = 300, INTERSECTIONCONESTART = 301,
 INTERSECTIONCONEEND = 302, DUALCONESTART = 303, DUALCONEEND = 304, POLARCONESTART = 305,
 POLARCONEEND = 306, DIRECTIONSTART = 307, DIRECTIONEND = 308, FACTORSSTART = 309, FACTORSEND = 310, COMPONENTSSTART = 311, COMPONENTSEND = 312, NORMSCALEFACTORATT = 313,
 DISTORTIONMATRIXIDXATT = 314, AXISDIRECTIONATT = 315, FIRSTAXISDIRECTIONATT = 316, SECONDAXISDIRECTIONATT = 317,
 EMPTYSEMIDEFINITENESSATT = 318, SEMIDEFINITENESSATT = 319, REFERENCEMATRIXIDXATT = 320, MATRIXPROGRAMMINGSTART = 321,
 MATRIXPROGRAMMINGEND = 322, VARTYPEATT = 323, MATRIXVARIABLESSTART = 324, MATRIXVARIABLESEND = 325,
 NUMBEROFMATRIXVARATT = 326, MATRIXVARSTART = 327, MATRIXVAREND = 328, MATRIXOBJECTIVESSTART = 329,
 MATRIXOBJECTIVESEND = 330, NUMBEROFMATRIXOBJATT = 331, MATRIXOBJSTART = 332, MATRIXOBJEND = 333,
 MATRIXCONSTRAINTSSTART = 334, MATRIXCONSTRAINTSEND = 335, NUMBEROFMATRIXCONATT = 336, MATRIXCONSTART = 337,
 MATRIXCONEND = 338, MATRIXIDXATT = 339, LBMATRIXIDXATT = 340, LBCONEIDXATT = 341, UBMATRIXIDXATT = 342, UBCONEIDXATT = 343, TEMPLATEMATRIXIDXATT = 344, VARREFERENCEMATRIXIDXATT = 345,
 OBJREFERENCEMATRIXIDXATT = 346, CONREFERENCEMATRIXIDXATT = 347, ORDERCONEIDXATT = 348, CONSTANTMATRIXIDXATT = 349,
 TIMEDOMAINSTART = 350, TIMEDOMAINEND = 351, STAGESSTART = 352, STAGESEND = 353, STAGESTART = 354, STAGEEND = 355, NUMBEROFSTAGESATT = 356, HORIZONATT = 357,
 STARTATT = 358, VARIABLESSTART = 359, CONSTRAINTSSTART = 360, OBJECTIVESSTART = 361, VARIABLESEND = 362, CONSTRAINTSEND = 363, OBJECTIVESEND = 364, NUMBEROFVARIABLESATT = 365,
 NUMBEROFCONSTRAINTSATT = 366, NUMBEROFOBJECTIVESATT = 367, STARTIDXATT = 368, VARSTART = 369,
 VAREND = 370, CONSTART = 371, CONEND = 372, OBJSTART = 373, OBJEND = 374, INTERVALSTART = 375, INTERVALEND = 376, HEADERSTART = 377,
 HEADEREND = 378, FILENAMESTART = 379, FILENAMEEND = 380, FILENAMEEMPTY = 381, FILENAMESTARTANDEND = 382, FILESOURCESTART = 383, FILESOURCEEND = 384, FILESOURCEEMPTY = 385,
 FILESOURCESTARTANDEND = 386, FILEDESCRIPTIONSTART = 387, FILEDESCRIPTIONEND = 388, FILEDESCRIPTIONEMPTY = 389,
 FILEDESCRIPTIONSTARTANDEND = 390, FILECREATORSTART = 391, FILECREATOREND = 392, FILECREATOREMPTY = 393,
 FILECREATORSTARTANDEND = 394, FILELICENCESTART = 395, FILELICENCEEND = 396, FILELICENCEEMPTY = 397,
 FILELICENCESTARTANDEND = 398, ENUMERATIONSTART = 399, ENUMERATIONEND = 400, NUMBEROF

```

FELATT = 401,
ITEMEMPTY = 402, ITEMSTART = 403, ITEMEND = 404, ITEMSTARTANDEND = 405,
BASE64START = 406, BASE64END = 407, INCRATT = 408, MULTATT = 409,
SIZEOFATT = 410, ELSTART = 411, ELEND = 412, MATRIXSTART = 413,
MATRIXEND = 414, BASEMATRIXEND = 415, BASEMATRIXSTART = 416, BLOCKSTART = 417,
BLOCKEND = 418, BLOCKSSTART = 419, BLOCKSEND = 420, EMPTYNAMEATT = 421,
NAMEATT = 422, EMPTYTYPEATT = 423, TYPEATT = 424, EMPTYSHAPEATT = 425,
SHAPEATT = 426, EMPTYSYMMETRYATT = 427, SYMMETRYATT = 428, EMPTYNEGATIVEPATTERNATT =
429,
NEGATIVEPATTERNATT = 430, CONSTANTATT = 431, NUMBEROFBLOCKSATT = 432, NUMBEROFCOLU-
MNSATT = 433,
NUMBEROFROWSATT = 434, NUMBEROFVALUESATT = 435, NUMBEROFVARIDXATT = 436, IDXATT =
437,
COEFATT = 438, BASEMATRIXIDXATT = 439, TARGETMATRIXFIRSTROWATT = 440, TARGETMATRIXFIR-
STCOLATT = 441,
BASEMATRIXSTARTROWATT = 442, BASEMATRIXSTARTCOLATT = 443, BASEMATRIXENDROWATT =
444, BASEMATRIXENDCOLATT = 445,
SCALARMULTIPLIERATT = 446, EMPTYBASETRANPOSEATT = 447, BASETRANPOSEATT = 448, ELEM-
ENTSSTART = 449,
ELEMENTSEND = 450, CONSTANTELEMENTSSTART = 451, CONSTANTELEMENTSEND = 452, STARTVE-
CTORSTART = 453,
STARTVECTOREND = 454, NONZEROSSTART = 455, NONZEROSEND = 456, INDEXESSTART = 457,
INDEXESEND = 458, VALUESSTART = 459, VALUESEND = 460, VARREFERENCEELEMENTSSTART = 461,
VARREFERENCEELEMENTSEND = 462, LINEARELEMENTSSTART = 463, LINEARELEMENTSEND = 464,
GENERALELEMENTSSTART = 465,
GENERALELEMENTSEND = 466, CONREFERENCEELEMENTSSTART = 467, CONREFERENCEELEMENT-
SEND = 468, VALUETYPEATT = 469,
OBJREFERENCEELEMENTSSTART = 470, OBJREFERENCEELEMENTSEND = 471, PATTERNELEMENTS-
START = 472, PATTERNELEMENTSEND = 473,
VARIDXSTART = 474, VARIDXEND = 475, TRANSFORMATIONSTART = 476, TRANSFORMATIONEND = 477,
COLOFFSETSSTART = 478, COLOFFSETSEND = 479, ROWOFFSETSSTART = 480, ROWOFFSETSEND =
481,
EMPTYROWMAJORATT = 482, ROWMAJORATT = 483, BLOCKROWIDXATT = 484, BLOCKCOLIDXATT =
485,
DUMMY = 486, NONLINEAREXPRESSIONSSTART = 487, NONLINEAREXPRESSIONSEND = 488, NUMBE-
ROFNONLINEAREXPRESSIONS = 489,
NLSTART = 490, NLEND = 491, MATRIXEXPRESSIONSSTART = 492, MATRIXEXPRESSIONSEND = 493,
NUMBEROFEXPR = 494, EXPRSTART = 495, EXPREND = 496, NUMBEROFMATRIXTERMSATT = 497,
MATRIXTERMSTART = 498, MATRIXTERMEND = 499, POWERSTART = 500, POWEREND = 501,
PLUSSTART = 502, PLUSEND = 503, MINUSSTART = 504, MINUSEND = 505,
DIVIDESTART = 506, DIVIDEEND = 507, LNSTART = 508, LNEEND = 509,
SQRTSTART = 510, SQRTEND = 511, SUMSTART = 512, SUMEND = 513,
PRODUCTSTART = 514, PRODUCTEND = 515, EXPSTART = 516, EXPEND = 517,
NEGATESTART = 518, NEGATEEND = 519, IFSTART = 520, IFEND = 521,
SQUARESTART = 522, SQUAREEND = 523, COSSTART = 524, COSEND = 525,
SINSTART = 526, SINEND = 527, VARIABLESTART = 528, VARIABLEEND = 529,
ABSSTART = 530, ABSEND = 531, ERFSTART = 532, ERFEND = 533,
MAXSTART = 534, MAXEND = 535, ALLDIFFSTART = 536, ALLDIFFEND = 537,
MINSTART = 538, MINEND = 539, ESTART = 540, EEND = 541,
PISTART = 542, PIEND = 543, TIMESSTART = 544, TIMESEND = 545,
NUMBERSTART = 546, NUMBEREND = 547, MATRIXDETERMINANTSTART = 548, MATRIXDETERMINANT-
END = 549,
MATRIXTRACESTART = 550, MATRIXTRACEEND = 551, MATRIXTOSCALARSTART = 552, MATRIXTOSCA-

```

LAREND = 553,
MATRIXDIAGONALSTART = 554, MATRIXDIAGONALEND = 555, MATRIXDOTTIMESSTART = 556, MATRIXDOTTIMESEND = 557,
MATRIXLOWERTRIANGLESTART = 558, MATRIXLOWERTRIANGLEEND = 559, MATRIXUPPERTRIANGLESTART = 560, MATRIXUPPERTRIANGLEEND = 561,
MATRIXMERGESTART = 562, MATRIXMERGEEND = 563, MATRIXMINUSSTART = 564, MATRIXMINUSEND = 565,
MATRIXNEGATESTART = 566, MATRIXNEGATEEND = 567, MATRIXPLUSSTART = 568, MATRIXPLUSEND = 569,
MATRIXTIMESSTART = 570, MATRIXTIMESEND = 571, MATRIXPRODUCTSTART = 572, MATRIXPRODUCTEND = 573,
MATRIXSCALARTIMESSTART = 574, MATRIXSCALARTIMESEND = 575, MATRIXSUBMATRIXATSTART = 576, MATRIXSUBMATRIXATEND = 577,
MATRIXTRANPOSESTART = 578, MATRIXTRANPOSEEND = 579, MATRIXREFERENCESTART = 580, MATRIXREFERENCEEND = 581,
IDENTITYMATRIXSTART = 582, IDENTITYMATRIXEND = 583, MATRIXINVERSESTART = 584, MATRIXINVERSEEND = 585,
EMPTYINCLUDEDIAGONALATT = 586, INCLUDEDIAGONALATT = 587, IDATT = 588, ATTRIBUTETEXT = 258,
ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
OSOLSTART = 267, OSOLSTARTEMPT = 268, OSOLATTRIBUTETEXT = 269, OSOLEND = 270,
NUMBEROFOTHEROPTIONSATT = 271, NUMBEROFENUMERATIONSATT = 272, NUMBEROFJOBIDSATT = 273, NUMBEROFPATHSATT = 274,
NUMBEROFPATHPAIRSATT = 275, FROMATT = 276, TOATT = 277, MAKECOPYATT = 278,
CATEGORYATT = 279, TYPEATT = 280, GROUPWEIGHTATT = 281, NUMBEROFPROCESSESATT = 282, NUMBEROF SOLVEROPTIONSATT = 283, NUMBEROF SOSATT = 284, NUMBEROFVARIABLESATT = 285, NUMBEROF OBJECTIVESATT = 286,
NUMBEROFCONSTRAINTSATT = 287, NUMBEROFOTHERVARIABLEOPTIONSATT = 288, NUMBEROFOTHEROBJECTIVEOPTIONSATT = 289, NUMBEROFOTHERCONSTRAINTOPTIONSATT = 290,
NUMBEROFITEMSATT = 291, NUMBEROFVARATT = 292, NUMBEROFOBJATT = 293, NUMBEROFCONATT = 294,
NUMBEROFELATT = 295, NAMEATT = 296, IDXATT = 297, SOSIDXATT = 298,
VALUEATT = 299, UNITATT = 300, DESCRIPTIONATT = 301, CONTYPEATT = 302,
EMPTYCONTYPEATT = 303, ENUMTYPEATT = 304, EMPTYENUMTYPEATT = 305, OBJTYPEATT = 306, EMPTYOBJTYPEATT = 307, VARTYPEATT = 308, EMPTYVARTYPEATT = 309, EMPTYTYPEATT = 310,
EMPTYNAMEATT = 311, EMPTYCATEGORYATT = 312, EMPTYDESCRIPTIONATT = 313, EMPTYUNITATT = 314,
EMPTYVALUEATT = 315, EMPTYLBVALUEATT = 316, EMPTYUBVALUEATT = 317, LBVALUEATT = 318, UBVALUEATT = 319, EMPTYLBDUALVALUEATT = 320, EMPTYUBDUALVALUEATT = 321, LBDUALVALUEATT = 322,
UBDUALVALUEATT = 323, SOLVERATT = 324, EMPTYSOLVERATT = 325, WEIGHTATT = 326, EMPTYWEIGHTATT = 327, TRANSPORTTYPEATT = 328, LOCATIONTYPEATT = 329, GENERALSTART = 330,
GENERALEND = 331, SYSTEMSTART = 332, SYSTEMEND = 333, SERVICESTART = 334, SERVICEEND = 335, JOBSTART = 336, JOBEND = 337, OPTIMIZATIONSTART = 338, OPTIMIZATIONEND = 339, SERVICEURISTART = 340, SERVICEURIEND = 341, SERVICENAMESTART = 342,
SERVICENAMEEND = 343, INSTANCENAMESTART = 344, INSTANCENAMEEND = 345, INSTANCELOCATI-

ONSTART = 346,
INSTANCELOCATIONEND = 347, JOBIDSTART = 348, JOBIDEND = 349, SOLVERTOINVOKESTART = 350,
SOLVERTOINVOKEEND = 351, LICENSESTART = 352, LICENSEEND = 353, USERNAMESTART = 354,
USERNAMEEND = 355, PASSWORDSTART = 356, PASSWORDEND = 357, CONTACTSTART = 358,
CONTACTEND = 359, OTHEROPTIONSSTART = 360, OTHEROPTIONSSEND = 361, OTHERSTART = 362,
OTHEREND = 363, MINDISKSPACESTART = 364, MINDISKSPACEEND = 365, MINMEMORYSTART = 366,
MINMEMORYEND = 367, MINCPUSPEEDSTART = 368, MINCPUSPEEDEND = 369, MINCPUNUMBERSTA-
RT = 370,
MINCPUNUMBEREND = 371, SERVICYPESTART = 372, SERVICYPEEND = 373, MAXTIMESTART =
374,
MAXTIMEEND = 375, REQUESTEDSTARTTIMESTART = 376, REQUESTEDSTARTTIMEEND = 377, DEPEND-
ENCIESSTART = 378,
DEPENDENCIESEND = 379, REQUIREDIRECTORIESSTART = 380, REQUIREDIRECTORIESEND = 381,
REQUIREDFILESSTART = 382,
REQUIREDFILESEND = 383, PATHSTART = 384, PATHEND = 385, PATHPAIRSTART = 386,
PATHPAIREND = 387, DIRECTORIESTOMAKESTART = 388, DIRECTORIESTOMAKEEND = 389, FILESTO-
MAKESTART = 390,
FILESTOMAKEEND = 391, DIRECTORIESTODELETESTART = 392, DIRECTORIESTODELETEEND = 393, F-
ILESTODELETESTART = 394,
FILESTODELETEEND = 395, INPUTDIRECTORIESTOMOVESTART = 396, INPUTDIRECTORIESTOMOVEE-
ND = 397, INPUTFILESTOMOVESTART = 398,
INPUTFILESTOMOVEEND = 399, OUTPUTDIRECTORIESTOMOVESTART = 400, OUTPUTDIRECTORIEST-
OMOVEEND = 401, OUTPUTFILESTOMOVESTART = 402,
OUTPUTFILESTOMOVEEND = 403, PROCESSESTOKILLSTART = 404, PROCESSESTOKILLEND = 405, P-
ROCESSSTART = 406,
PROCESSEND = 407, VARIABLESSTART = 408, VARIABLESEND = 409, INITIALVARIABLEVALUESSTART
= 410,
INITIALVARIABLEVALUESSEND = 411, VARSTART = 412, VAREND = 413, INITIALVARIABLEVALUESSTRIN-
GSTART = 414,
INITIALVARIABLEVALUESSTRINGEND = 415, INITIALBASISSTATUSSTART = 416, INITIALBASISSTATUSE-
ND = 417, BASICSTART = 418,
BASICEND = 419, ATUPPERSTART = 420, ATUPPEREND = 421, ATLOWERSTART = 422,
ATLOWEREND = 423, ATEQUALITYSTART = 424, ATEQUALITYEND = 425, SUPERBASICSTART = 426,
SUPERBASICEND = 427, ISFREESTART = 428, ISFREEEND = 429, UNKNOWNSTART = 430,
UNKNOWNEND = 431, INTEGERVARIABLEBRANCHINGWEIGHTSSTART = 432, INTEGERVARIABLEBRAN-
CHINGWEIGHTSEND = 433, SOSVARIABLEBRANCHINGWEIGHTSSTART = 434,
SOSVARIABLEBRANCHINGWEIGHTSEND = 435, SOSSTART = 436, SOSEND = 437, OBJECTIVESSTART =
438,
OBJECTIVESEND = 439, INITIALOBJECTIVEVALUESSTART = 440, INITIALOBJECTIVEVALUESEND = 441,
OBJSTART = 442,
OBJEND = 443, INITIALOBJECTIVEBOUNDSSTART = 444, INITIALOBJECTIVEBOUNDSEND = 445, CONST-
RAINTSSTART = 446,
CONSTRAINTSEND = 447, INITIALCONSTRAINTVALUESSTART = 448, INITIALCONSTRAINTVALUESEND
= 449, CONSTART = 450,
CONEND = 451, INITIALDUALVALUESSTART = 452, INITIALDUALVALUESEND = 453, SOLVEROPTIONSS-
TART = 454,
SOLVEROPTIONSEND = 455, SOLVEROPTIONSTART = 456, SOLVEROPTIONEND = 457, ENUMERATION-

START = 458,
ENUMERATIONEND = 459, ITEMEMPTY = 460, ITEMSTART = 461, ITEMEND = 462,
ITEMSTARTANDEND = 463, BASE64START = 464, BASE64END = 465, INCRATT = 466,
MULTATT = 467, SIZEOFATT = 468, ELSTART = 469, ELEND = 470,
MATRIXVARSTART = 471, MATRIXVAREND = 472, MATRIXOBJSTART = 473, MATRIXOBJEND = 474,
MATRIXCONSTART = 475, MATRIXCONEND = 476, HEADERSTART = 477, HEADEREND = 478,
FILENAMESTART = 479, FILENAMEEND = 480, FILENAMEEMPTY = 481, FILENAMESTARTANDEND = 482,
FILESOURCESTART = 483, FILESOURCEEND = 484, FILESOURCEEMPTY = 485, FILESOURCESTARTANDEND = 486,
FILEDESCRIPTIONSTART = 487, FILEDESCRIPTIONEND = 488, FILEDESCRIPTIONEMPTY = 489, FILEDESCRIPTIONSTARTANDEND = 490,
FILECREATORSTART = 491, FILECREATOREND = 492, FILECREATOREMPTY = 493, FILECREATORSTARTANDEND = 494,
FILELICENCESTART = 495, FILELICENCEEND = 496, FILELICENCEEMPTY = 497, FILELICENCESTARTANDEND = 498,
MATRIXSTART = 499, MATRIXEND = 500, BASEMATRIXEND = 501, BASEMATRIXSTART = 502,
BLOCKSTART = 503, BLOCKEND = 504, BLOCKSSTART = 505, BLOCKSEND = 506,
EMPTYSHAPEATT = 507, SHAPEATT = 508, EMPTYSYMMETRYATT = 509, SYMMETRYATT = 510,
EMPTYNEGATIVEPATTERNATT = 511, NEGATIVEPATTERNATT = 512, CONSTANTATT = 513, NUMBEROFBLOCKSATT = 514,
NUMBEROFCOLUMNSATT = 515, NUMBEROFROWSATT = 516, NUMBEROFVALUESATT = 517, NUMBEROFVARIDXATT = 518,
COEFATT = 519, BASEMATRIXIDXATT = 520, TARGETMATRIXFIRSTROWATT = 521, TARGETMATRIXFIRSTCOLATT = 522,
BASEMATRIXSTARTROWATT = 523, BASEMATRIXSTARTCOLATT = 524, BASEMATRIXENDROWATT = 525, BASEMATRIXENDCOLATT = 526,
SCALARMULTIPLIERATT = 527, EMPTYBASETRANPOSEATT = 528, BASETRANPOSEATT = 529, ELEMENTSSTART = 530,
ELEMENTSEND = 531, CONSTANTELEMENTSSTART = 532, CONSTANTELEMENTSEND = 533, STARTVECTORSTART = 534,
STARTVECTOREND = 535, NONZEROSSTART = 536, NONZEROSSEND = 537, INDEXESSTART = 538, INDEXESEND = 539, VALUESSTART = 540, VALUESEND = 541, VARREFERENCEELEMENTSSTART = 542, VARREFERENCEELEMENTSEND = 543, LINEARELEMENTSSTART = 544, LINEARELEMENTSEND = 545, GENERALELEMENTSSTART = 546, GENERALELEMENTSEND = 547, CONREFERENCEELEMENTSSTART = 548, CONREFERENCEELEMENTSEND = 549, VALUETYPEATT = 550,
OBJREFERENCEELEMENTSSTART = 551, OBJREFERENCEELEMENTSEND = 552, PATTERNELEMENTSSTART = 553, PATTERNELEMENTSEND = 554,
VARIDXSTART = 555, VARIDXEND = 556, TRANSFORMATIONSTART = 557, TRANSFORMATIONEND = 558, COLOFFSETSSTART = 559, COLOFFSETSEND = 560, ROWOFFSETSSTART = 561, ROWOFFSETSEND = 562,
EMPTYROWMAJORATT = 563, ROWMAJORATT = 564, BLOCKROWIDXATT = 565, BLOCKCOLIDXATT = 566,
DUMMY = 567, NONLINEAREXPRESSIONSSTART = 568, NONLINEAREXPRESSIONSEND = 569, NUMBE-

ROFNONLINEAREXPRESSIONS = 570,
NLSTART = 571, NLEND = 572, MATRIXEXPRESSIONSSTART = 573, MATRIXEXPRESSIONSEND = 574,
NUMBEROFEXPR = 575, EXPRSTART = 576, EXPREND = 577, NUMBEROFMATRIXTERMSATT = 578,
MATRIXTERMSTART = 579, MATRIXTERMEND = 580, POWERSTART = 581, POWEREND = 582,
PLUSSTART = 583, PLUSEND = 584, MINUSSTART = 585, MINUSEND = 586,
DIVIDESTART = 587, DIVIDEEND = 588, LNSTART = 589, LLEND = 590,
SQRTSTART = 591, SQRTEND = 592, SUMSTART = 593, SUMEND = 594,
PRODUCTSTART = 595, PRODUCTEND = 596, EXPSTART = 597, EXPEND = 598,
NEGATESTART = 599, NEGATEEND = 600, IFSTART = 601, IFEND = 602,
SQUARESTART = 603, SQUAREEND = 604, COSSTART = 605, COSEND = 606,
SINSTART = 607, SINEND = 608, VARIABLESTART = 609, VARIABLEEND = 610,
ABSSTART = 611, ABSEND = 612, ERFSTART = 613, ERFEND = 614,
MAXSTART = 615, MAXEND = 616, ALLDIFFSTART = 617, ALLDIFFEND = 618,
MINSTART = 619, MINEND = 620, ESTART = 621, EEND = 622,
PISTART = 623, PIEND = 624, TIMESSTART = 625, TIMESEND = 626,
NUMBERSTART = 627, NUMBEREND = 628, MATRIXDETERMINANTSTART = 629, MATRIXDETERMINANT-
END = 630,
MATRIXTRACESTART = 631, MATRIXTRACEEND = 632, MATRIXTOSCALARSTART = 633, MATRIXTOSCA-
LAREND = 634,
MATRIXDIAGONALSTART = 635, MATRIXDIAGONALEND = 636, MATRIXDOTTIMESSTART = 637, MATRIX-
DOTTIMESEND = 638,
MATRIXLOWERTRIANGLESTART = 639, MATRIXLOWERTRIANGLEEND = 640, MATRIXUPPERTRIANGLE-
START = 641, MATRIXUPPERTRIANGLEEND = 642,
MATRIXMERGESTART = 643, MATRIXMERGEEND = 644, MATRIXMINUSSTART = 645, MATRIXMINUSEND
= 646,
MATRIXNEGATESTART = 647, MATRIXNEGATEEND = 648, MATRIXPLUSSTART = 649, MATRIXPLUSEND
= 650,
MATRIXTIMESSTART = 651, MATRIXTIMESEND = 652, MATRIXPRODUCTSTART = 653, MATRIXPRODUC-
TEND = 654,
MATRIXSCALARTIMESSTART = 655, MATRIXSCALARTIMESEND = 656, MATRIXSUBMATRIXATSTART =
657, MATRIXSUBMATRIXATEND = 658,
MATRIXTRANPOSESTART = 659, MATRIXTRANPOSEEND = 660, MATRIXREFERENCESTART = 661, M-
ATRIXREFERENCEEND = 662,
IDENTITYMATRIXSTART = 663, IDENTITYMATRIXEND = 664, MATRIXINVERSESTART = 665, MATRIXINV-
ERSEEND = 666,
EMPTYINCLUDEDIAGONALATT = 667, INCLUDEDIAGONALATT = 668, IDATT = 669, ATTRIBUTETEXT =
258,
ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
OSRLSTART = 267, OSRLSTARTEMPT = 268, OSRLATTRIBUTETEXT = 269, OSRLLEND = 270,
NUMBEROFCONATT = 271, NUMBEROFCONSTRAINTSATT = 272, NUMBEROFELATT = 273, NUMBEROF-
ENUMERATIONSATT = 274,
NUMBEROFIDXATT = 275, NUMBEROFITEMSATT = 276, NUMBEROFOBJATT = 277, NUMBEROFOBJECT-
IVESATT = 278,
NUMBEROFOTHERCONSTRAINTRESULTSATT = 279, NUMBEROFOTHEROBJECTIVERESULTSATT = 280,
NUMBEROFOTHERRESULTSATT = 281, NUMBEROFOTHERSOLUTIONRESULTSATT = 282,
NUMBEROFOTHERVARIABLERESULTSATT = 283, NUMBEROFSOLUTIONSATT = 284, NUMBEROFSOLV-
EROUTPUTSATT = 285, NUMBEROFSUBSTATUSESATT = 286,
NUMBEROFTIMESATT = 287, NUMBEROFVARATT = 288, NUMBEROFVARIABLESATT = 289, NUMBEROF-

VARIDXATT = 290,
TARGETOBJECTIVEIDXATT = 291, IDXATT = 292, INCRATT = 293, MULTATT = 294,
SIZEOFATT = 295, CATEGORYATT = 296, EMPTYCATEGORYATT = 297, DESCRIPTIONATT = 298,
EMPTYDESCRIPTIONATT = 299, NAMEATT = 300, EMPTYNAMEATT = 301, TYPEATT = 302,
EMPTYTYPEATT = 303, CONTYPEATT = 304, EMPTYCONTYPEATT = 305, ENUMTYPEATT = 306,
EMPTYENUMTYPEATT = 307, OBJTYPEATT = 308, EMPTYOBJTYPEATT = 309, VARTYPEATT = 310,
EMPTYVARTYPEATT = 311, UNITATT = 312, EMPTYUNITATT = 313, VALUEATT = 314,
EMPTYVALUEATT = 315, WEIGHTEDOBJECTIVESATT = 316, EMPTYWEIGHTEDOBJECTIVESATT = 317,
TARGETOBJECTIVENAMEATT = 318,
EMPTYTARGETOBJECTIVENAMEATT = 319, HEADERSTART = 320, HEADEREND = 321, GENERALSTART
= 322,
GENERALEND = 323, SYSTEMSTART = 324, SYSTEMEND = 325, SERVICESTART = 326,
SERVICEEND = 327, JOBSTART = 328, JOBEND = 329, OPTIMIZATIONSTART = 330,
OPTIMIZATIONEND = 331, ITEMSTART = 332, ITEMEND = 333, ITEMSTARTANDEND = 334,
ITEMEMPTY = 335, ACTUALSTARTTIMESTART = 336, ACTUALSTARTTIMEEND = 337, ATEQUALITYSTART
= 338,
ATEQUALITYEND = 339, ATLOWERSTART = 340, ATLOWEREND = 341, ATUPPERSTART = 342,
ATUPPEREND = 343, AVAILABLECPUNUMBERSTART = 344, AVAILABLECPUNUMBEREND = 345, AVAILA-
BLECPUSPEEDSTART = 346,
AVAILABLECPUSPEEDEND = 347, AVAILABLEDISKSPACESTART = 348, AVAILABLEDISKSPACEEND =
349, AVAILABLEMEMORYSTART = 350,
AVAILABLEMEMORYEND = 351, BASE64START = 352, BASE64END = 353, BASICSTART = 354,
BASICEND = 355, BASISSTATUSSTART = 356, BASISSTATUSEND = 357, BASSTATUSSTART = 358,
BASSTATUSEND = 359, CONSTART = 360, CONEND = 361, CONSTRAINTSSTART = 362,
CONSTRAINTSEND = 363, CURRENTJOBCOUNTSTART = 364, CURRENTJOBCOUNTEND = 365, CURRE-
NTSTATESTART = 366,
CURRENTSTATEEND = 367, DUALVALUESSTART = 368, DUALVALUESEND = 369, ELSTART = 370,
EEND = 371, ENUMERATIONSTART = 372, ENUMERATIONEND = 373, ENDTIMESTART = 374,
ENDTIMEEND = 375, GENERALSTATUSSTART = 376, GENERALSTATUSEND = 377, GENERALSUBSTAT-
USSTART = 378,
GENERALSUBSTATUSEND = 379, IDXSTART = 380, IDXEND = 381, INSTANCENAMESTART = 382,
INSTANCENAMEEND = 383, ISFREESTART = 384, ISFREEEND = 385, JOBIDSTART = 386,
JOBIDEND = 387, MESSAGESTART = 388, MESSAGEEND = 389, OBJSTART = 390,
OBJEND = 391, OBJECTIVESSTART = 392, OBJECTIVESEND = 393, OPTIMIZATIONSOLUTIONSTATUSST-
ART = 394,
OPTIMIZATIONSOLUTIONSTATUSEND = 395, OPTIMIZATIONSOLUTIONSUBSTATUSSTART = 396, OPTI-
MIZATIONSOLUTIONSUBSTATUSEND = 397, OTHERSTART = 398,
OTHEREND = 399, OTHERRESULTSSTART = 400, OTHERRESULTSEND = 401, OTHERSOLUTIONRESUL-
TSTART = 402,
OTHERSOLUTIONRESULTEND = 403, OTHERSOLUTIONRESULTSSTART = 404, OTHERSOLUTIONRESUL-
TSEND = 405, OTHERSOLVEROUTPUTSTART = 406,
OTHERSOLVEROUTPUTEND = 407, SCHEDULEDSTARTTIMESTART = 408, SCHEDULEDSTARTTIMEEND
= 409, SERVICENAMESTART = 410,
SERVICENAMEEND = 411, SERVICEURISTART = 412, SERVICEURIEND = 413, SERVICEUTILIZATIONST-
ART = 414,
SERVICEUTILIZATIONEND = 415, SOLUTIONSTART = 416, SOLUTIONEND = 417, SOLVERINVOKEDSTA-
RT = 418,
SOLVERINVOKEDEND = 419, SOLVEROUTPUTSTART = 420, SOLVEROUTPUTEND = 421, STATUSSTART
= 422,
STATUSEND = 423, SUBMITTIMESTART = 424, SUBMITTIMEEND = 425, SUBSTATUSSTART = 426,
SUBSTATUSEND = 427, SUPERBASICSTART = 428, SUPERBASICEND = 429, SYSTEMINFORMATIONST-
ART = 430,
SYSTEMINFORMATIONEND = 431, TIMESTART = 432, TIMEEND = 433, TIMESERVICESTARTEDSTART =

434,
 TIMESERVICESTARTEDEND = 435, TIMESTAMPSTART = 436, TIMESTAMPEND = 437, TIMINGINFORMATIONSTART = 438,
 TIMINGINFORMATIONEND = 439, TOTALJOBSSOFARSTART = 440, TOTALJOBSSOFAREND = 441, UNKNOWNSTART = 442,
 UNKNOWNEND = 443, USEDCPUNUMBERSTART = 444, USEDCPUNUMBEREND = 445, USEDCPUSPEEDSTART = 446,
 USEDCPUSPEEDEND = 447, USEDDISKSPACESTART = 448, USEDDISKSPACEEND = 449, USEDMEMORYSTART = 450,
 USEDMEMORYEND = 451, VALUESSTRINGSTART = 452, VALUESSTRINGEND = 453, VARSTART = 454, VAREND = 455,
 VARIABLESSTART = 456, VARIABLESEND = 457, VARIDXSTART = 458, VARIDXEND = 459, MATRIXVARSTART = 460,
 MATRIXVAREND = 461, MATRIXOBJSTART = 462, MATRIXOBJEND = 463, MATRIXCONSTART = 464, MATRIXCONEND = 465,
 FILENAMESTART = 466, FILENAMEEND = 467, FILENAMEEMPTY = 468, FILENAMESTARTANDEND = 469, FILESOURCESTART = 470,
 FILESOURCEEND = 471, FILESOURCEEMPTY = 472, FILESOURCESTARTANDEND = 473, FILEDESCRIPTIONSTART = 474,
 FILEDESCRIPTIONEND = 475, FILEDESCRIPTIONEMPTY = 476, FILEDESCRIPTIONSTARTANDEND = 477, FILECREATORSTART = 478,
 FILECREATOREND = 479, FILECREATOREMPTY = 480, FILECREATORSTARTANDEND = 481, FILELICENCESTART = 482,
 FILELICENCEEND = 483, FILELICENCEEMPTY = 484, FILELICENCESTARTANDEND = 485, MATRIXSTART = 486,
 MATRIXEND = 487, BASEMATRIXEND = 488, BASEMATRIXSTART = 489, BLOCKSTART = 490, BLOCKEND = 491,
 BLOCKSSTART = 492, BLOCKSEND = 493, EMPTYSHAPEATT = 494, SHAPEATT = 495, EMPTYSYMMETRYATT = 496,
 SYMMETRYATT = 497, EMPTYNEGATIVEPATTERNATT = 498, NEGATIVEPATTERNATT = 499, CONSTANTATT = 500,
 NUMBEROFBLOCKSATT = 501, NUMBEROFCOLUMNSSATT = 502, NUMBEROFROWSATT = 503, NUMBEROFVALUESATT = 504,
 COEFATT = 505, BASEMATRIXIDXATT = 506, TARGETMATRIXFIRSTROWATT = 507, TARGETMATRIXFIRSTCOLATT = 508,
 BASEMATRIXSTARTROWATT = 509, BASEMATRIXSTARTCOLATT = 510, BASEMATRIXENDROWATT = 511, BASEMATRIXENDCOLATT = 512,
 SCALARMULTIPLIERATT = 513, EMPTYBASETRANPOSEATT = 514, BASETRANPOSEATT = 515, ELEMENTSSTART = 516,
 ELEMENTSEND = 517, CONSTATELEMENTSSTART = 518, CONSTATELEMENTSEND = 519, STARTVECTORSTART = 520,
 STARTVECTOREND = 521, NONZEROSTART = 522, NONZEROSSEND = 523, INDEXESSTART = 524, INDEXESEND = 525,
 VALUESSTART = 526, VALUESEND = 527, VARREFERENCEELEMENTSSTART = 528, VARREFERENCEELEMENTSEND = 529,
 LINEARELEMENTSSTART = 530, LINEARELEMENTSEND = 531, GENERALELEMENTSSTART = 532, GENERALELEMENTSEND = 533,
 CONREFERENCEELEMENTSSTART = 534, CONREFERENCEELEMENTSEND = 535, VALUETYPEATT = 536, OBJREFERENCEELEMENTSSTART = 537,
 OBJREFERENCEELEMENTSEND = 538, PATTERNELEMENTSSTART = 539, PATTERNELEMENTSEND = 540, TRANSFORMATIONSTART = 541,
 TRANSFORMATIONEND = 542, COLOFFSETSSTART = 543, COLOFFSETSEND = 544, ROWOFFSETSSTART = 545, ROWOFFSETSEND = 546,
 EMPTYROWMAJORATT = 547, ROWMAJORATT = 548, BLOCKROWIDXATT = 549, BLOCKCOLIDXATT = 550,
 DUMMY = 551, NONLINEAREXPRESSIONSSTART = 552, NONLINEAREXPRESSIONSEND = 553, NUMBE

```

ROFNONLINEAREXPRESSIONS = 554,
NLSTART = 555, NLEND = 556, MATRIXEXPRESSIONSSTART = 557, MATRIXEXPRESSIONSEND = 558,
NUMBEROFEXPR = 559, EXPRSTART = 560, EXPREND = 561, NUMBEROFMATRIXTERMSATT = 562,
MATRIXTERMSTART = 563, MATRIXTERMEND = 564, POWERSTART = 565, POWEREND = 566,
PLUSSTART = 567, PLUSEND = 568, MINUSSTART = 569, MINUSEND = 570,
DIVIDESTART = 571, DIVIDEEND = 572, LNSTART = 573, LNEND = 574,
SQRTSTART = 575, SQRTEND = 576, SUMSTART = 577, SUMEND = 578,
PRODUCTSTART = 579, PRODUCTEND = 580, EXPSTART = 581, EXPEND = 582,
NEGATESTART = 583, NEGATEEND = 584, IFSTART = 585, IFEND = 586,
SQUARESTART = 587, SQUAREEND = 588, COSSTART = 589, COSEND = 590,
SINSTART = 591, SINEND = 592, VARIABLESTART = 593, VARIABLEEND = 594,
ABSSTART = 595, ABSEND = 596, ERFSTART = 597, ERFEND = 598,
MAXSTART = 599, MAXEND = 600, ALLDIFFSTART = 601, ALLDIFFEND = 602,
MINSTART = 603, MINEND = 604, ESTART = 605, EEND = 606,
PISTART = 607, PIEND = 608, TIMESSTART = 609, TIMESEND = 610,
NUMBERSTART = 611, NUMBEREND = 612, MATRIXDETERMINANTSTART = 613, MATRIXDETERMINANT-
END = 614,
MATRIXTRACESTART = 615, MATRIXTRACEEND = 616, MATRIXTOSCALARSTART = 617, MATRIXTOSCA-
LAREND = 618,
MATRIXDIAGONALSTART = 619, MATRIXDIAGONALEND = 620, MATRIXDOTTIMESSTART = 621, MATRIX-
DOTTIMESEND = 622,
MATRIXLOWERTRIANGLESTART = 623, MATRIXLOWERTRIANGLEEND = 624, MATRIXUPPERTRIANGLE-
START = 625, MATRIXUPPERTRIANGLEEND = 626,
MATRIXMERGESTART = 627, MATRIXMERGEEND = 628, MATRIXMINUSSTART = 629, MATRIXMINUSEND
= 630,
MATRIXNEGATESTART = 631, MATRIXNEGATEEND = 632, MATRIXPLUSSTART = 633, MATRIXPLUSEND
= 634,
MATRIXTIMESSTART = 635, MATRIXTIMESEND = 636, MATRIXPRODUCTSTART = 637, MATRIXPRODUC-
TEND = 638,
MATRIXSCALARTIMESSTART = 639, MATRIXSCALARTIMESEND = 640, MATRIXSUBMATRIXATSTART =
641, MATRIXSUBMATRIXATEND = 642,
MATRIXTRANPOSESTART = 643, MATRIXTRANPOSEEND = 644, MATRIXREFERENCESTART = 645, M-
ATRIXREFERENCEEND = 646,
IDENTITYMATRIXSTART = 647, IDENTITYMATRIXEND = 648, MATRIXINVERSESTART = 649, MATRIXINV-
ERSEEND = 650,
EMPTYINCLUDEDIAGONALATT = 651, INCLUDEDIAGONALATT = 652, IDATT = 653 }

```

7.35.1 Macro Definition Documentation

7.35.1.1 #define QUOTE 258

Definition at line 376 of file OSParseosil.tab.hpp.

7.35.1.2 #define ATTRIBUTETEXT 259

Definition at line 377 of file OSParseosil.tab.hpp.

7.35.1.3 #define ELEMENTTEXT 260

Definition at line 378 of file OSParseosil.tab.hpp.

7.35.1.4 #define ITEMTEXT 261

Definition at line 379 of file OSParseosil.tab.hpp.

7.35.1.5 #define INTEGER 262

Definition at line 380 of file OSParseosil.tab.hpp.

7.35.1.6 #define DOUBLE 263

Definition at line 381 of file OSParseosil.tab.hpp.

7.35.1.7 #define TWOQUOTES 264

Definition at line 382 of file OSParseosil.tab.hpp.

7.35.1.8 #define ENDOFELEMENT 265

Definition at line 383 of file OSParseosil.tab.hpp.

7.35.1.9 #define GREATERTHAN 266

Definition at line 384 of file OSParseosil.tab.hpp.

7.35.1.10 #define OSILEND 267

Definition at line 385 of file OSParseosil.tab.hpp.

7.35.1.11 #define INSTANCEDATAEND 268

Definition at line 386 of file OSParseosil.tab.hpp.

7.35.1.12 #define INSTANCEDATASTARTEND 269

Definition at line 387 of file OSParseosil.tab.hpp.

7.35.1.13 #define EMPTYIDATT 270

Definition at line 388 of file OSParseosil.tab.hpp.

7.35.1.14 #define IDXONEATT 271

Definition at line 389 of file OSParseosil.tab.hpp.

7.35.1.15 #define IDXTWOATT 272

Definition at line 390 of file OSParseosil.tab.hpp.

7.35.1.16 #define VALUEATT 273

Definition at line 391 of file OSParseosil.tab.hpp.

7.35.1.17 #define QUADRATICCOEFFICIENTSSTART 274

Definition at line 392 of file OSParseosil.tab.hpp.

7.35.1.18 #define QUADRATICCOEFFICIENTSEND 275

Definition at line 393 of file OSParseosil.tab.hpp.

7.35.1.19 #define NUMBEROFQTERMSATT 276

Definition at line 394 of file OSParseosil.tab.hpp.

7.35.1.20 #define QTERMSTART 277

Definition at line 395 of file OSParseosil.tab.hpp.

7.35.1.21 #define QTERMEND 278

Definition at line 396 of file OSParseosil.tab.hpp.

7.35.1.22 #define MATRICESSTART 279

Definition at line 397 of file OSParseosil.tab.hpp.

7.35.1.23 #define MATRICESEND 280

Definition at line 398 of file OSParseosil.tab.hpp.

7.35.1.24 #define NUMBEROFMATRICESATT 281

Definition at line 399 of file OSParseosil.tab.hpp.

7.35.1.25 #define CONESSTART 282

Definition at line 400 of file OSParseosil.tab.hpp.

7.35.1.26 #define CONESEND 283

Definition at line 401 of file OSParseosil.tab.hpp.

7.35.1.27 #define NUMBEROFCONESATT 284

Definition at line 402 of file OSParseosil.tab.hpp.

7.35.1.28 #define NONNEGATIVECONESTART 285

Definition at line 403 of file OSParseosil.tab.hpp.

7.35.1.29 #define NONNEGATIVECONEEND 286

Definition at line 404 of file OSParseosil.tab.hpp.

7.35.1.30 #define NONPOSITIVECONESTART 287

Definition at line 405 of file OSParseosil.tab.hpp.

7.35.1.31 #define NONPOSITIVECONEEND 288

Definition at line 406 of file OSParseosil.tab.hpp.

7.35.1.32 #define ORTHANTCONESTART 289

Definition at line 407 of file OSParseosil.tab.hpp.

7.35.1.33 #define ORTHANTCONEEND 290

Definition at line 408 of file OSParseosil.tab.hpp.

7.35.1.34 #define POLYHEDRALCONESTART 291

Definition at line 409 of file OSParseosil.tab.hpp.

7.35.1.35 #define POLYHEDRALCONEEND 292

Definition at line 410 of file OSParseosil.tab.hpp.

7.35.1.36 #define QUADRATICCONESTART 293

Definition at line 411 of file OSParseosil.tab.hpp.

7.35.1.37 #define QUADRATICCONEEND 294

Definition at line 412 of file OSParseosil.tab.hpp.

7.35.1.38 #define ROTATEDQUADRATICCONESTART 295

Definition at line 413 of file OSParseosil.tab.hpp.

7.35.1.39 #define ROTATEDQUADRATICCONEEND 296

Definition at line 414 of file OSParseosil.tab.hpp.

7.35.1.40 #define SEMIDEFINITECONESTART 297

Definition at line 415 of file OSParseosil.tab.hpp.

7.35.1.41 #define SEMIDEFINITECONEEND 298

Definition at line 416 of file OSParseosil.tab.hpp.

7.35.1.42 #define PRODUCTCONESTART 299

Definition at line 417 of file OSParseosil.tab.hpp.

7.35.1.43 #define PRODUCTCONEEND 300

Definition at line 418 of file OSParseosil.tab.hpp.

7.35.1.44 #define INTERSECTIONCONESTART 301

Definition at line 419 of file OSParseosil.tab.hpp.

7.35.1.45 #define INTERSECTIONCONEEND 302

Definition at line 420 of file OSParseosil.tab.hpp.

7.35.1.46 #define DUALCONESTART 303

Definition at line 421 of file OSParseosil.tab.hpp.

7.35.1.47 #define DUALCONEEND 304

Definition at line 422 of file OSParseosil.tab.hpp.

7.35.1.48 #define POLARCONESTART 305

Definition at line 423 of file OSParseosil.tab.hpp.

7.35.1.49 #define POLARCONEEND 306

Definition at line 424 of file OSParseosil.tab.hpp.

7.35.1.50 #define DIRECTIONSTART 307

Definition at line 425 of file OSParseosil.tab.hpp.

7.35.1.51 #define DIRECTIONEND 308

Definition at line 426 of file OSParseosil.tab.hpp.

7.35.1.52 #define FACTORSSTART 309

Definition at line 427 of file OSParseosil.tab.hpp.

7.35.1.53 #define FACTORSEND 310

Definition at line 428 of file OSParseosil.tab.hpp.

7.35.1.54 #define COMPONENTSSTART 311

Definition at line 429 of file OSParseosil.tab.hpp.

7.35.1.55 #define COMPONENTSEND 312

Definition at line 430 of file OSParseosil.tab.hpp.

7.35.1.56 #define NORMSCALEFACTORATT 313

Definition at line 431 of file OSParseosil.tab.hpp.

7.35.1.57 #define DISTORTIONMATRIXIDXATT 314

Definition at line 432 of file OSParseosil.tab.hpp.

7.35.1.58 #define AXISDIRECTIONATT 315

Definition at line 433 of file OSParseosil.tab.hpp.

7.35.1.59 #define FIRSTAXISDIRECTIONATT 316

Definition at line 434 of file OSParseosil.tab.hpp.

7.35.1.60 #define SECONDAXISDIRECTIONATT 317

Definition at line 435 of file OSParseosil.tab.hpp.

7.35.1.61 **#define EMPTYSEMIDEFINITENESSATT 318**

Definition at line 436 of file OSParseosil.tab.hpp.

7.35.1.62 **#define SEMIDEFINITENESSATT 319**

Definition at line 437 of file OSParseosil.tab.hpp.

7.35.1.63 **#define REFERENCEMATRIXIDXATT 320**

Definition at line 438 of file OSParseosil.tab.hpp.

7.35.1.64 **#define MATRIXPROGRAMMINGSTART 321**

Definition at line 439 of file OSParseosil.tab.hpp.

7.35.1.65 **#define MATRIXPROGRAMMINGEND 322**

Definition at line 440 of file OSParseosil.tab.hpp.

7.35.1.66 **#define VARTYPEATT 323**

Definition at line 441 of file OSParseosil.tab.hpp.

7.35.1.67 **#define MATRIXVARIABLESSTART 324**

Definition at line 442 of file OSParseosil.tab.hpp.

7.35.1.68 **#define MATRIXVARIABLESEND 325**

Definition at line 443 of file OSParseosil.tab.hpp.

7.35.1.69 **#define NUMBEROFMATRIXVARATT 326**

Definition at line 444 of file OSParseosil.tab.hpp.

7.35.1.70 **#define MATRIXVARSTART 327**

Definition at line 445 of file OSParseosil.tab.hpp.

7.35.1.71 **#define MATRIXVAREND 328**

Definition at line 446 of file OSParseosil.tab.hpp.

7.35.1.72 **#define MATRIXOBJECTIVESSTART 329**

Definition at line 447 of file OSParseosil.tab.hpp.

7.35.1.73 **#define MATRIXOBJECTIVESEND 330**

Definition at line 448 of file OSParseosil.tab.hpp.

7.35.1.74 **#define NUMBEROFMATRIXOBJATT 331**

Definition at line 449 of file OSParseosil.tab.hpp.

7.35.1.75 #define MATRIXOBJSTART 332

Definition at line 450 of file OSParseosil.tab.hpp.

7.35.1.76 #define MATRIXOBJEND 333

Definition at line 451 of file OSParseosil.tab.hpp.

7.35.1.77 #define MATRIXCONSTRAINTSSTART 334

Definition at line 452 of file OSParseosil.tab.hpp.

7.35.1.78 #define MATRIXCONSTRAINTSEND 335

Definition at line 453 of file OSParseosil.tab.hpp.

7.35.1.79 #define NUMBEROFMATRIXCONATT 336

Definition at line 454 of file OSParseosil.tab.hpp.

7.35.1.80 #define MATRIXCONSTART 337

Definition at line 455 of file OSParseosil.tab.hpp.

7.35.1.81 #define MATRIXCONEND 338

Definition at line 456 of file OSParseosil.tab.hpp.

7.35.1.82 #define MATRIXIDXATT 339

Definition at line 457 of file OSParseosil.tab.hpp.

7.35.1.83 #define LBMATRIXIDXATT 340

Definition at line 458 of file OSParseosil.tab.hpp.

7.35.1.84 #define LBCONEIDXATT 341

Definition at line 459 of file OSParseosil.tab.hpp.

7.35.1.85 #define UBMATRIXIDXATT 342

Definition at line 460 of file OSParseosil.tab.hpp.

7.35.1.86 #define UBCONEIDXATT 343

Definition at line 461 of file OSParseosil.tab.hpp.

7.35.1.87 #define TEMPLATEMATRIXIDXATT 344

Definition at line 462 of file OSParseosil.tab.hpp.

7.35.1.88 #define VARREFERENCEMATRIXIDXATT 345

Definition at line 463 of file OSParseosil.tab.hpp.

7.35.1.89 **#define OBJREFERENCEMATRIXIDXATT 346**

Definition at line 464 of file OSParseosil.tab.hpp.

7.35.1.90 **#define CONREFERENCEMATRIXIDXATT 347**

Definition at line 465 of file OSParseosil.tab.hpp.

7.35.1.91 **#define ORDERCONEIDXATT 348**

Definition at line 466 of file OSParseosil.tab.hpp.

7.35.1.92 **#define CONSTANTMATRIXIDXATT 349**

Definition at line 467 of file OSParseosil.tab.hpp.

7.35.1.93 **#define TIMEDOMAINSTART 350**

Definition at line 468 of file OSParseosil.tab.hpp.

7.35.1.94 **#define TIMEDOMAINEND 351**

Definition at line 469 of file OSParseosil.tab.hpp.

7.35.1.95 **#define STAGESSTART 352**

Definition at line 470 of file OSParseosil.tab.hpp.

7.35.1.96 **#define STAGESEND 353**

Definition at line 471 of file OSParseosil.tab.hpp.

7.35.1.97 **#define STAGESTART 354**

Definition at line 472 of file OSParseosil.tab.hpp.

7.35.1.98 **#define STAGEEND 355**

Definition at line 473 of file OSParseosil.tab.hpp.

7.35.1.99 **#define NUMBEROFSTAGESATT 356**

Definition at line 474 of file OSParseosil.tab.hpp.

7.35.1.100 **#define HORIZONATT 357**

Definition at line 475 of file OSParseosil.tab.hpp.

7.35.1.101 **#define STARTATT 358**

Definition at line 476 of file OSParseosil.tab.hpp.

7.35.1.102 **#define VARIABLESSTART 359**

Definition at line 477 of file OSParseosil.tab.hpp.

7.35.1.103 #define CONSTRAINTSSTART 360

Definition at line 478 of file OSParseosil.tab.hpp.

7.35.1.104 #define OBJECTIVESSTART 361

Definition at line 479 of file OSParseosil.tab.hpp.

7.35.1.105 #define VARIABLESEND 362

Definition at line 480 of file OSParseosil.tab.hpp.

7.35.1.106 #define CONSTRAINTSEND 363

Definition at line 481 of file OSParseosil.tab.hpp.

7.35.1.107 #define OBJECTIVESEND 364

Definition at line 482 of file OSParseosil.tab.hpp.

7.35.1.108 #define NUMBEROFVARIABLESATT 365

Definition at line 483 of file OSParseosil.tab.hpp.

7.35.1.109 #define NUMBEROFCONSTRAINTSATT 366

Definition at line 484 of file OSParseosil.tab.hpp.

7.35.1.110 #define NUMBEROFOBJECTIVESATT 367

Definition at line 485 of file OSParseosil.tab.hpp.

7.35.1.111 #define STARTIDXATT 368

Definition at line 486 of file OSParseosil.tab.hpp.

7.35.1.112 #define VARSTART 369

Definition at line 487 of file OSParseosil.tab.hpp.

7.35.1.113 #define VAREND 370

Definition at line 488 of file OSParseosil.tab.hpp.

7.35.1.114 #define CONSTART 371

Definition at line 489 of file OSParseosil.tab.hpp.

7.35.1.115 #define CONEND 372

Definition at line 490 of file OSParseosil.tab.hpp.

7.35.1.116 #define OBJSTART 373

Definition at line 491 of file OSParseosil.tab.hpp.

7.35.1.117 #define OBJEND 374

Definition at line 492 of file OSParseosil.tab.hpp.

7.35.1.118 #define INTERVALSTART 375

Definition at line 493 of file OSParseosil.tab.hpp.

7.35.1.119 #define INTERVALEND 376

Definition at line 494 of file OSParseosil.tab.hpp.

7.35.1.120 #define HEADERSTART 377

Definition at line 495 of file OSParseosil.tab.hpp.

7.35.1.121 #define HEADEREND 378

Definition at line 496 of file OSParseosil.tab.hpp.

7.35.1.122 #define FILENAMESTART 379

Definition at line 497 of file OSParseosil.tab.hpp.

7.35.1.123 #define FILENAMEEND 380

Definition at line 498 of file OSParseosil.tab.hpp.

7.35.1.124 #define FILENAMEEMPTY 381

Definition at line 499 of file OSParseosil.tab.hpp.

7.35.1.125 #define FILENAMESTARTANDEND 382

Definition at line 500 of file OSParseosil.tab.hpp.

7.35.1.126 #define FILESOURCESTART 383

Definition at line 501 of file OSParseosil.tab.hpp.

7.35.1.127 #define FILESOURCEEND 384

Definition at line 502 of file OSParseosil.tab.hpp.

7.35.1.128 #define FILESOURCEEMPTY 385

Definition at line 503 of file OSParseosil.tab.hpp.

7.35.1.129 #define FILESOURCESTARTANDEND 386

Definition at line 504 of file OSParseosil.tab.hpp.

7.35.1.130 #define FILEDESCRIPTIONSTART 387

Definition at line 505 of file OSParseosil.tab.hpp.

7.35.1.131 **#define FILEDESCRIPTIONEND 388**

Definition at line 506 of file OSParseosil.tab.hpp.

7.35.1.132 **#define FILEDESCRIPTIONEMPTY 389**

Definition at line 507 of file OSParseosil.tab.hpp.

7.35.1.133 **#define FILEDESCRIPTIONSTARTANDEND 390**

Definition at line 508 of file OSParseosil.tab.hpp.

7.35.1.134 **#define FILECREATORSTART 391**

Definition at line 509 of file OSParseosil.tab.hpp.

7.35.1.135 **#define FILECREATOREND 392**

Definition at line 510 of file OSParseosil.tab.hpp.

7.35.1.136 **#define FILECREATOREMPTY 393**

Definition at line 511 of file OSParseosil.tab.hpp.

7.35.1.137 **#define FILECREATORSTARTANDEND 394**

Definition at line 512 of file OSParseosil.tab.hpp.

7.35.1.138 **#define FILELICENCESTART 395**

Definition at line 513 of file OSParseosil.tab.hpp.

7.35.1.139 **#define FILELICENCEEND 396**

Definition at line 514 of file OSParseosil.tab.hpp.

7.35.1.140 **#define FILELICENCEEMPTY 397**

Definition at line 515 of file OSParseosil.tab.hpp.

7.35.1.141 **#define FILELICENCESTARTANDEND 398**

Definition at line 516 of file OSParseosil.tab.hpp.

7.35.1.142 **#define ENUMERATIONSTART 399**

Definition at line 517 of file OSParseosil.tab.hpp.

7.35.1.143 **#define ENUMERATIONEND 400**

Definition at line 518 of file OSParseosil.tab.hpp.

7.35.1.144 **#define NUMBEROFELATT 401**

Definition at line 519 of file OSParseosil.tab.hpp.

7.35.1.145 **#define ITEMEMPTY 402**

Definition at line 520 of file OSParseosil.tab.hpp.

7.35.1.146 **#define ITEMSTART 403**

Definition at line 521 of file OSParseosil.tab.hpp.

7.35.1.147 **#define ITEMEND 404**

Definition at line 522 of file OSParseosil.tab.hpp.

7.35.1.148 **#define ITEMSTARTANDEND 405**

Definition at line 523 of file OSParseosil.tab.hpp.

7.35.1.149 **#define BASE64START 406**

Definition at line 524 of file OSParseosil.tab.hpp.

7.35.1.150 **#define BASE64END 407**

Definition at line 525 of file OSParseosil.tab.hpp.

7.35.1.151 **#define INCRATT 408**

Definition at line 526 of file OSParseosil.tab.hpp.

7.35.1.152 **#define MULTATT 409**

Definition at line 527 of file OSParseosil.tab.hpp.

7.35.1.153 **#define SIZEOFATT 410**

Definition at line 528 of file OSParseosil.tab.hpp.

7.35.1.154 **#define ELSTART 411**

Definition at line 529 of file OSParseosil.tab.hpp.

7.35.1.155 **#define ELEND 412**

Definition at line 530 of file OSParseosil.tab.hpp.

7.35.1.156 **#define MATRIXSTART 413**

Definition at line 531 of file OSParseosil.tab.hpp.

7.35.1.157 **#define MATRIXEND 414**

Definition at line 532 of file OSParseosil.tab.hpp.

7.35.1.158 **#define BASEMATRIXEND 415**

Definition at line 533 of file OSParseosil.tab.hpp.

7.35.1.159 **#define BASEMATRIXSTART 416**

Definition at line 534 of file OSParseosil.tab.hpp.

7.35.1.160 **#define BLOCKSTART 417**

Definition at line 535 of file OSParseosil.tab.hpp.

7.35.1.161 **#define BLOCKEND 418**

Definition at line 536 of file OSParseosil.tab.hpp.

7.35.1.162 **#define BLOCKSSTART 419**

Definition at line 537 of file OSParseosil.tab.hpp.

7.35.1.163 **#define BLOCKSEND 420**

Definition at line 538 of file OSParseosil.tab.hpp.

7.35.1.164 **#define EMPTYNAMEATT 421**

Definition at line 539 of file OSParseosil.tab.hpp.

7.35.1.165 **#define NAMEATT 422**

Definition at line 540 of file OSParseosil.tab.hpp.

7.35.1.166 **#define EMPTYTYPEATT 423**

Definition at line 541 of file OSParseosil.tab.hpp.

7.35.1.167 **#define TYPEATT 424**

Definition at line 542 of file OSParseosil.tab.hpp.

7.35.1.168 **#define EMPTYSHAPEATT 425**

Definition at line 543 of file OSParseosil.tab.hpp.

7.35.1.169 **#define SHAPEATT 426**

Definition at line 544 of file OSParseosil.tab.hpp.

7.35.1.170 **#define EMPTYSYMMETRYATT 427**

Definition at line 545 of file OSParseosil.tab.hpp.

7.35.1.171 **#define SYMMETRYATT 428**

Definition at line 546 of file OSParseosil.tab.hpp.

7.35.1.172 **#define EMPTYNEGATIVEPATTERNATT 429**

Definition at line 547 of file OSParseosil.tab.hpp.

7.35.1.173 **#define NEGATIVEPATTERNATT 430**

Definition at line 548 of file OSParseosil.tab.hpp.

7.35.1.174 **#define CONSTANTATT 431**

Definition at line 549 of file OSParseosil.tab.hpp.

7.35.1.175 **#define NUMBEROFBLOCKSATT 432**

Definition at line 550 of file OSParseosil.tab.hpp.

7.35.1.176 **#define NUMBEROFCOLUMNSATT 433**

Definition at line 551 of file OSParseosil.tab.hpp.

7.35.1.177 **#define NUMBEROFROWSATT 434**

Definition at line 552 of file OSParseosil.tab.hpp.

7.35.1.178 **#define NUMBEROFVALUESATT 435**

Definition at line 553 of file OSParseosil.tab.hpp.

7.35.1.179 **#define NUMBEROFVARIDXATT 436**

Definition at line 554 of file OSParseosil.tab.hpp.

7.35.1.180 **#define IDXATT 437**

Definition at line 555 of file OSParseosil.tab.hpp.

7.35.1.181 **#define COEFATT 438**

Definition at line 556 of file OSParseosil.tab.hpp.

7.35.1.182 **#define BASEMATRIXIDXATT 439**

Definition at line 557 of file OSParseosil.tab.hpp.

7.35.1.183 **#define TARGETMATRIXFIRSTROWATT 440**

Definition at line 558 of file OSParseosil.tab.hpp.

7.35.1.184 **#define TARGETMATRIXFIRSTCOLATT 441**

Definition at line 559 of file OSParseosil.tab.hpp.

7.35.1.185 **#define BASEMATRIXSTARTROWATT 442**

Definition at line 560 of file OSParseosil.tab.hpp.

7.35.1.186 **#define BASEMATRIXSTARTCOLATT 443**

Definition at line 561 of file OSParseosil.tab.hpp.

7.35.1.187 **#define BASEMATRIXENDROWATT 444**

Definition at line 562 of file OSParseosil.tab.hpp.

7.35.1.188 **#define BASEMATRIXENDCOLATT 445**

Definition at line 563 of file OSParseosil.tab.hpp.

7.35.1.189 **#define SCALARMULTIPLIERATT 446**

Definition at line 564 of file OSParseosil.tab.hpp.

7.35.1.190 **#define EMPTYBASETRANSPPOSEATT 447**

Definition at line 565 of file OSParseosil.tab.hpp.

7.35.1.191 **#define BASETRANSPPOSEATT 448**

Definition at line 566 of file OSParseosil.tab.hpp.

7.35.1.192 **#define ELEMENTSSTART 449**

Definition at line 567 of file OSParseosil.tab.hpp.

7.35.1.193 **#define ELEMENTSEND 450**

Definition at line 568 of file OSParseosil.tab.hpp.

7.35.1.194 **#define CONSTANTELEMENTSSTART 451**

Definition at line 569 of file OSParseosil.tab.hpp.

7.35.1.195 **#define CONSTANTELEMENTSEND 452**

Definition at line 570 of file OSParseosil.tab.hpp.

7.35.1.196 **#define STARTVECTORSTART 453**

Definition at line 571 of file OSParseosil.tab.hpp.

7.35.1.197 **#define STARTVECTOREND 454**

Definition at line 572 of file OSParseosil.tab.hpp.

7.35.1.198 **#define NONZEROSSTART 455**

Definition at line 573 of file OSParseosil.tab.hpp.

7.35.1.199 **#define NONZEROSEND 456**

Definition at line 574 of file OSParseosil.tab.hpp.

7.35.1.200 **#define INDEXESSTART 457**

Definition at line 575 of file OSParseosil.tab.hpp.

7.35.1.201 #define INDEXESEND 458

Definition at line 576 of file OSParseosil.tab.hpp.

7.35.1.202 #define VALUESSTART 459

Definition at line 577 of file OSParseosil.tab.hpp.

7.35.1.203 #define VALUESEND 460

Definition at line 578 of file OSParseosil.tab.hpp.

7.35.1.204 #define VARREFERENCEELEMENTSSTART 461

Definition at line 579 of file OSParseosil.tab.hpp.

7.35.1.205 #define VARREFERENCEELEMENTSEND 462

Definition at line 580 of file OSParseosil.tab.hpp.

7.35.1.206 #define LINEARELEMENTSSTART 463

Definition at line 581 of file OSParseosil.tab.hpp.

7.35.1.207 #define LINEARELEMENTSEND 464

Definition at line 582 of file OSParseosil.tab.hpp.

7.35.1.208 #define GENERALELEMENTSSTART 465

Definition at line 583 of file OSParseosil.tab.hpp.

7.35.1.209 #define GENERALELEMENTSEND 466

Definition at line 584 of file OSParseosil.tab.hpp.

7.35.1.210 #define CONREFERENCEELEMENTSSTART 467

Definition at line 585 of file OSParseosil.tab.hpp.

7.35.1.211 #define CONREFERENCEELEMENTSEND 468

Definition at line 586 of file OSParseosil.tab.hpp.

7.35.1.212 #define VALUETYPEATT 469

Definition at line 587 of file OSParseosil.tab.hpp.

7.35.1.213 #define OBJREFERENCEELEMENTSSTART 470

Definition at line 588 of file OSParseosil.tab.hpp.

7.35.1.214 #define OBJREFERENCEELEMENTSEND 471

Definition at line 589 of file OSParseosil.tab.hpp.

7.35.1.215 **#define PATTERNELEMENTSSTART 472**

Definition at line 590 of file OSParseosil.tab.hpp.

7.35.1.216 **#define PATTERNELEMENTSEND 473**

Definition at line 591 of file OSParseosil.tab.hpp.

7.35.1.217 **#define VARIDXSTART 474**

Definition at line 592 of file OSParseosil.tab.hpp.

7.35.1.218 **#define VARIDXEND 475**

Definition at line 593 of file OSParseosil.tab.hpp.

7.35.1.219 **#define TRANSFORMATIONSTART 476**

Definition at line 594 of file OSParseosil.tab.hpp.

7.35.1.220 **#define TRANSFORMATIONEND 477**

Definition at line 595 of file OSParseosil.tab.hpp.

7.35.1.221 **#define COLOFFSETSSTART 478**

Definition at line 596 of file OSParseosil.tab.hpp.

7.35.1.222 **#define COLOFFSETSEND 479**

Definition at line 597 of file OSParseosil.tab.hpp.

7.35.1.223 **#define ROWOFFSETSSTART 480**

Definition at line 598 of file OSParseosil.tab.hpp.

7.35.1.224 **#define ROWOFFSETSEND 481**

Definition at line 599 of file OSParseosil.tab.hpp.

7.35.1.225 **#define EMPTYROWMAJORATT 482**

Definition at line 600 of file OSParseosil.tab.hpp.

7.35.1.226 **#define ROWMAJORATT 483**

Definition at line 601 of file OSParseosil.tab.hpp.

7.35.1.227 **#define BLOCKROWIDXATT 484**

Definition at line 602 of file OSParseosil.tab.hpp.

7.35.1.228 **#define BLOCKCOLIDXATT 485**

Definition at line 603 of file OSParseosil.tab.hpp.

7.35.1.229 #define DUMMY 486

Definition at line 604 of file OSParseosil.tab.hpp.

7.35.1.230 #define NONLINEAREXPRESSIONSSTART 487

Definition at line 605 of file OSParseosil.tab.hpp.

7.35.1.231 #define NONLINEAREXPRESSIONSEND 488

Definition at line 606 of file OSParseosil.tab.hpp.

7.35.1.232 #define NUMBEROFNONLINEAREXPRESSIONS 489

Definition at line 607 of file OSParseosil.tab.hpp.

7.35.1.233 #define NLSTART 490

Definition at line 608 of file OSParseosil.tab.hpp.

7.35.1.234 #define NLEND 491

Definition at line 609 of file OSParseosil.tab.hpp.

7.35.1.235 #define MATRIXEXPRESSIONSSTART 492

Definition at line 610 of file OSParseosil.tab.hpp.

7.35.1.236 #define MATRIXEXPRESSIONSEND 493

Definition at line 611 of file OSParseosil.tab.hpp.

7.35.1.237 #define NUMBEROFEXPR 494

Definition at line 612 of file OSParseosil.tab.hpp.

7.35.1.238 #define EXPRSTART 495

Definition at line 613 of file OSParseosil.tab.hpp.

7.35.1.239 #define EXPREND 496

Definition at line 614 of file OSParseosil.tab.hpp.

7.35.1.240 #define NUMBEROFMATRIXTERMSATT 497

Definition at line 615 of file OSParseosil.tab.hpp.

7.35.1.241 #define MATRIXTERMSTART 498

Definition at line 616 of file OSParseosil.tab.hpp.

7.35.1.242 #define MATRIXTERMEND 499

Definition at line 617 of file OSParseosil.tab.hpp.

7.35.1.243 #define POWERSTART 500

Definition at line 618 of file OSParseosil.tab.hpp.

7.35.1.244 #define POWEREND 501

Definition at line 619 of file OSParseosil.tab.hpp.

7.35.1.245 #define PLUSSTART 502

Definition at line 620 of file OSParseosil.tab.hpp.

7.35.1.246 #define PLUSEND 503

Definition at line 621 of file OSParseosil.tab.hpp.

7.35.1.247 #define MINUSSTART 504

Definition at line 622 of file OSParseosil.tab.hpp.

7.35.1.248 #define MINUSEND 505

Definition at line 623 of file OSParseosil.tab.hpp.

7.35.1.249 #define DIVIDESTART 506

Definition at line 624 of file OSParseosil.tab.hpp.

7.35.1.250 #define DIVIDEEND 507

Definition at line 625 of file OSParseosil.tab.hpp.

7.35.1.251 #define LNSTART 508

Definition at line 626 of file OSParseosil.tab.hpp.

7.35.1.252 #define LNEND 509

Definition at line 627 of file OSParseosil.tab.hpp.

7.35.1.253 #define SQRTSTART 510

Definition at line 628 of file OSParseosil.tab.hpp.

7.35.1.254 #define SQRTEND 511

Definition at line 629 of file OSParseosil.tab.hpp.

7.35.1.255 #define SUMSTART 512

Definition at line 630 of file OSParseosil.tab.hpp.

7.35.1.256 #define SUMEND 513

Definition at line 631 of file OSParseosil.tab.hpp.

7.35.1.257 #define PRODUCTSTART 514

Definition at line 632 of file OSParseosil.tab.hpp.

7.35.1.258 #define PRODUCTEND 515

Definition at line 633 of file OSParseosil.tab.hpp.

7.35.1.259 #define EXPSTART 516

Definition at line 634 of file OSParseosil.tab.hpp.

7.35.1.260 #define EXPEND 517

Definition at line 635 of file OSParseosil.tab.hpp.

7.35.1.261 #define NEGATESTART 518

Definition at line 636 of file OSParseosil.tab.hpp.

7.35.1.262 #define NEGATEEND 519

Definition at line 637 of file OSParseosil.tab.hpp.

7.35.1.263 #define IFSTART 520

Definition at line 638 of file OSParseosil.tab.hpp.

7.35.1.264 #define IFEND 521

Definition at line 639 of file OSParseosil.tab.hpp.

7.35.1.265 #define SQUARESTART 522

Definition at line 640 of file OSParseosil.tab.hpp.

7.35.1.266 #define SQUAREEND 523

Definition at line 641 of file OSParseosil.tab.hpp.

7.35.1.267 #define COSSTART 524

Definition at line 642 of file OSParseosil.tab.hpp.

7.35.1.268 #define COSEND 525

Definition at line 643 of file OSParseosil.tab.hpp.

7.35.1.269 #define SINSTART 526

Definition at line 644 of file OSParseosil.tab.hpp.

7.35.1.270 #define SINEND 527

Definition at line 645 of file OSParseosil.tab.hpp.

7.35.1.271 #define VARIABLESTART 528

Definition at line 646 of file OSParseosil.tab.hpp.

7.35.1.272 #define VARIABLEEND 529

Definition at line 647 of file OSParseosil.tab.hpp.

7.35.1.273 #define ABSSTART 530

Definition at line 648 of file OSParseosil.tab.hpp.

7.35.1.274 #define ABSEND 531

Definition at line 649 of file OSParseosil.tab.hpp.

7.35.1.275 #define ERFSTART 532

Definition at line 650 of file OSParseosil.tab.hpp.

7.35.1.276 #define ERFEND 533

Definition at line 651 of file OSParseosil.tab.hpp.

7.35.1.277 #define MAXSTART 534

Definition at line 652 of file OSParseosil.tab.hpp.

7.35.1.278 #define MAXEND 535

Definition at line 653 of file OSParseosil.tab.hpp.

7.35.1.279 #define ALLDIFFSTART 536

Definition at line 654 of file OSParseosil.tab.hpp.

7.35.1.280 #define ALLDIFFEND 537

Definition at line 655 of file OSParseosil.tab.hpp.

7.35.1.281 #define MINSTART 538

Definition at line 656 of file OSParseosil.tab.hpp.

7.35.1.282 #define MINEND 539

Definition at line 657 of file OSParseosil.tab.hpp.

7.35.1.283 #define ESTART 540

Definition at line 658 of file OSParseosil.tab.hpp.

7.35.1.284 #define EEND 541

Definition at line 659 of file OSParseosil.tab.hpp.

7.35.1.285 #define PISTART 542

Definition at line 660 of file OSParseosil.tab.hpp.

7.35.1.286 #define PIEND 543

Definition at line 661 of file OSParseosil.tab.hpp.

7.35.1.287 #define TIMESSTART 544

Definition at line 662 of file OSParseosil.tab.hpp.

7.35.1.288 #define TIMESEND 545

Definition at line 663 of file OSParseosil.tab.hpp.

7.35.1.289 #define NUMBERSTART 546

Definition at line 664 of file OSParseosil.tab.hpp.

7.35.1.290 #define NUMBEREND 547

Definition at line 665 of file OSParseosil.tab.hpp.

7.35.1.291 #define MATRIXDETERMINANTSTART 548

Definition at line 666 of file OSParseosil.tab.hpp.

7.35.1.292 #define MATRIXDETERMINANTEND 549

Definition at line 667 of file OSParseosil.tab.hpp.

7.35.1.293 #define MATRIXTRACESTART 550

Definition at line 668 of file OSParseosil.tab.hpp.

7.35.1.294 #define MATRIXTRACEEND 551

Definition at line 669 of file OSParseosil.tab.hpp.

7.35.1.295 #define MATRIXTOSCALARSTART 552

Definition at line 670 of file OSParseosil.tab.hpp.

7.35.1.296 #define MATRIXTOSCALAREND 553

Definition at line 671 of file OSParseosil.tab.hpp.

7.35.1.297 #define MATRIXDIAGONALSTART 554

Definition at line 672 of file OSParseosil.tab.hpp.

7.35.1.298 #define MATRIXDIAGONALEND 555

Definition at line 673 of file OSParseosil.tab.hpp.

7.35.1.299 **#define** MATRIXDOTTIMESSTART 556

Definition at line 674 of file OSParseosil.tab.hpp.

7.35.1.300 **#define** MATRIXDOTTIMESEND 557

Definition at line 675 of file OSParseosil.tab.hpp.

7.35.1.301 **#define** MATRIXLOWERTRIANGLESTART 558

Definition at line 676 of file OSParseosil.tab.hpp.

7.35.1.302 **#define** MATRIXLOWERTRIANGLEEND 559

Definition at line 677 of file OSParseosil.tab.hpp.

7.35.1.303 **#define** MATRIXUPPERTRIANGLESTART 560

Definition at line 678 of file OSParseosil.tab.hpp.

7.35.1.304 **#define** MATRIXUPPERTRIANGLEEND 561

Definition at line 679 of file OSParseosil.tab.hpp.

7.35.1.305 **#define** MATRIXMERGESTART 562

Definition at line 680 of file OSParseosil.tab.hpp.

7.35.1.306 **#define** MATRIXMERGEEND 563

Definition at line 681 of file OSParseosil.tab.hpp.

7.35.1.307 **#define** MATRIXMINUSSTART 564

Definition at line 682 of file OSParseosil.tab.hpp.

7.35.1.308 **#define** MATRIXMINUSEND 565

Definition at line 683 of file OSParseosil.tab.hpp.

7.35.1.309 **#define** MATRIXNEGATESTART 566

Definition at line 684 of file OSParseosil.tab.hpp.

7.35.1.310 **#define** MATRIXNEGATEEND 567

Definition at line 685 of file OSParseosil.tab.hpp.

7.35.1.311 **#define** MATRIXPLUSSTART 568

Definition at line 686 of file OSParseosil.tab.hpp.

7.35.1.312 **#define** MATRIXPLUSEND 569

Definition at line 687 of file OSParseosil.tab.hpp.

7.35.1.313 #define MATRIXTIMESSTART 570

Definition at line 688 of file OSParseosil.tab.hpp.

7.35.1.314 #define MATRIXTIMESEND 571

Definition at line 689 of file OSParseosil.tab.hpp.

7.35.1.315 #define MATRIXPRODUCTSTART 572

Definition at line 690 of file OSParseosil.tab.hpp.

7.35.1.316 #define MATRIXPRODUCTEND 573

Definition at line 691 of file OSParseosil.tab.hpp.

7.35.1.317 #define MATRIXSCALARTIMESSTART 574

Definition at line 692 of file OSParseosil.tab.hpp.

7.35.1.318 #define MATRIXSCALARTIMESEND 575

Definition at line 693 of file OSParseosil.tab.hpp.

7.35.1.319 #define MATRIXSUBMATRIXATSTART 576

Definition at line 694 of file OSParseosil.tab.hpp.

7.35.1.320 #define MATRIXSUBMATRIXATEND 577

Definition at line 695 of file OSParseosil.tab.hpp.

7.35.1.321 #define MATRIXTRANSPOSESTART 578

Definition at line 696 of file OSParseosil.tab.hpp.

7.35.1.322 #define MATRIXTRANSPOSEEND 579

Definition at line 697 of file OSParseosil.tab.hpp.

7.35.1.323 #define MATRIXREFERENCESTART 580

Definition at line 698 of file OSParseosil.tab.hpp.

7.35.1.324 #define MATRIXREFERENCEEND 581

Definition at line 699 of file OSParseosil.tab.hpp.

7.35.1.325 #define IDENTITYMATRIXSTART 582

Definition at line 700 of file OSParseosil.tab.hpp.

7.35.1.326 #define IDENTITYMATRIXEND 583

Definition at line 701 of file OSParseosil.tab.hpp.

7.35.1.327 **#define MATRIXINVERSESTART 584**

Definition at line 702 of file OSParseosil.tab.hpp.

7.35.1.328 **#define MATRIXINVERSEEND 585**

Definition at line 703 of file OSParseosil.tab.hpp.

7.35.1.329 **#define EMPTYINCLUDEDIAGONALATT 586**

Definition at line 704 of file OSParseosil.tab.hpp.

7.35.1.330 **#define INCLUDEDIAGONALATT 587**

Definition at line 705 of file OSParseosil.tab.hpp.

7.35.1.331 **#define IDATT 588**

Definition at line 706 of file OSParseosil.tab.hpp.

7.35.1.332 **#define YYSTYPE_IS_TRIVIAL 1**

Definition at line 724 of file OSParseosil.tab.hpp.

7.35.1.333 **#define yystype YYSTYPE /* obsolescent; will be withdrawn */**

Definition at line 725 of file OSParseosil.tab.hpp.

7.35.1.334 **#define YYSTYPE_IS_DECLARED 1**

Definition at line 726 of file OSParseosil.tab.hpp.

7.35.1.335 **#define yyltype YYLTYPE /* obsolescent; will be withdrawn */**

Definition at line 739 of file OSParseosil.tab.hpp.

7.35.1.336 **#define YYLTYPE_IS_DECLARED 1**

Definition at line 740 of file OSParseosil.tab.hpp.

7.35.1.337 **#define YYLTYPE_IS_TRIVIAL 1**

Definition at line 741 of file OSParseosil.tab.hpp.

7.35.2 Typedef Documentation

7.35.2.1 **typedef union YYSTYPE YYSTYPE**

7.35.2.2 **typedef struct YYLTYPE YYLTYPE**

7.35.3 Enumeration Type Documentation

7.35.3.1 **enum yytokentype**

Enumerator:

QUOTE

ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
TWOQUOTES
ENDOFELEMENT
GREATERTHAN
OSILEND
INSTANCEDATAEND
INSTANCEDATASTARTEND
EMPTYIDATT
IDXONEATT
IDXTWOATT
VALUEATT
QUADRATICCOEFFICIENTSSTART
QUADRATICCOEFFICIENTSEND
NUMBEROFQTERMSATT
QTERMSTART
QTERMEND
MATRICESSTART
MATRICESEND
NUMBEROFMATRICESATT
CONESSTART
CONESEND
NUMBEROFCONESATT
NONNEGATIVECONESTART
NONNEGATIVECONEEND
NONPOSITIVECONESTART
NONPOSITIVECONEEND
ORTHANTCONESTART
ORTHANTCONEEND
POLYHEDRALCONESTART
POLYHEDRALCONEEND
QUADRATICCONESTART
QUADRATICCONEEND
ROTATEDQUADRATICCONESTART
ROTATEDQUADRATICCONEEND
SEMIDEFINITECONESTART
SEMIDEFINITECONEEND
PRODUCTCONESTART
PRODUCTCONEEND

INTERSECTIONCONESTART
INTERSECTIONCONEEND
DUALCONESTART
DUALCONEEND
POLARCONESTART
POLARCONEEND
DIRECTIONSTART
DIRECTIONEND
FACTORSSTART
FACTORSEND
COMPONENTSSTART
COMPONENTSEND
NORMSCALEFACTORATT
DISTORTIONMATRIXIDXATT
AXISDIRECTIONATT
FIRSTAXISDIRECTIONATT
SECONDAXISDIRECTIONATT
EMPTYSEMIDEFINITENESSATT
SEMIDEFINITENESSATT
REFERENCEMATRIXIDXATT
MATRIXPROGRAMMINGSTART
MATRIXPROGRAMMINGEND
VARTYPEATT
MATRIXVARIABLESSTART
MATRIXVARIABLESEND
NUMBEROFMATRIXVARATT
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJECTIVESSTART
MATRIXOBJECTIVESEND
NUMBEROFMATRIXOBJATT
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTRAINTSSTART
MATRIXCONSTRAINTSEND
NUMBEROFMATRIXCONATT
MATRIXCONSTART
MATRIXCONEND
MATRIXIDXATT
LBMATRIXIDXATT
LBCONEIDXATT
UBMATRIXIDXATT

UBCONEIDXATT
TEMPLATEMATRIXIDXATT
VARREFERENCEMATRIXIDXATT
OBJREFERENCEMATRIXIDXATT
CONREFERENCEMATRIXIDXATT
ORDERCONEIDXATT
CONSTANTMATRIXIDXATT
TIMEDOMAINSTART
TIMEDOMAINEND
STAGESSTART
STAGESEND
STAGESTART
STAGEEND
NUMBEROFSTAGESATT
HORIZONATT
STARTATT
VARIABLESSTART
CONSTRAINTSSTART
OBJECTIVESSTART
VARIABLESEND
CONSTRAINTSEND
OBJECTIVESEND
NUMBEROFVARIABLESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOBJECTIVESATT
STARTIDXATT
VARSTART
VAREND
CONSTART
CONEND
OBJSTART
OBJEND
INTERVALSTART
INTERVALEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND

FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
ENUMERATIONSTART
ENUMERATIONEND
NUMBEROFELATT
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYNAMEATT
NAMEATT
EMPTYTYPEATT
TYPEATT
EMPTYSHAPEATT
SHAPEATT

EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
NUMBEROFVARIDXATT
IDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPPOSEATT
BASETRANSPPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND

VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART

SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART

MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMSEEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMSEEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANPOSESTART
MATRIXTRANPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE

TWOQUOTES
GREATERTHAN
ENDOFELEMENT
OSOLSTART
OSOLSTARTEMPT
OSOLATTRIBUTETEXT
SOLEND
NUMBEROFOTHEROPTIONSATT
NUMBEROFENUMERATIONSATT
NUMBEROFJOBIDSATT
NUMBEROFPATHSATT
NUMBEROFPATHPAIRSATT
FROMATT
TOATT
MAKECOPYATT
CATEGORYATT
TYPEATT
GROUPWEIGHTATT
NUMBEROFPROCESSESATT
NUMBEROFSOLVEROPTIONSATT
NUMBEROFSOSATT
NUMBEROFVARIABLESATT
NUMBEROFOBJECTIVESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOTHERVARIABLEOPTIONSATT
NUMBEROFOTHEROBJECTIVEOPTIONSATT
NUMBEROFOTHERCONSTRAINTOPTIONSATT
NUMBEROFITEMSATT
NUMBEROFVARATT
NUMBEROFOBJATT
NUMBEROFCONATT
NUMBEROFELATT
NAMEATT
IDXATT
SOSIDXATT
VALUEATT
UNITATT
DESCRIPTIONATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT

OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
EMPTYTYPEATT
EMPTYNAMEATT
EMPTYCATEGORYATT
EMPTYDESCRIPTIONATT
EMPTYUNITATT
EMPTYVALUEATT
EMPTYLBVALUEATT
EMPTYUBVALUEATT
LBVALUEATT
UBVALUEATT
EMPTYLBDUALVALUEATT
EMPTYUBDUALVALUEATT
LBDUALVALUEATT
UBDUALVALUEATT
SOLVERATT
EMPTYSOLVERATT
WEIGHTATT
EMPTYWEIGHTATT
TRANSPORTTYPEATT
LOCATIONTYPEATT
GENERALSTART
GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND
JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
SERVICEURISTART
SERVICEURIEND
SERVICENAMESTART
SERVICENAMEEND
INSTANCENAMESTART
INSTANCENAMEEND
INSTANCELOCATIONSTART
INSTANCELOCATIONEND

JOBIDSTART
JOBIDEND
SOLVERTOINVOKESTART
SOLVERTOINVOKEEND
LICENSESTART
LICENSEEND
USERNAMESTART
USERNAMEEND
PASSWORDSTART
PASSWORDEND
CONTACTSTART
CONTACTEND
OTHEROPTIONSSTART
OTHEROPTIONSSEND
OTHERSTART
OTHEREND
MINDISKSPACESTART
MINDISKSPACEEND
MINMEMORYSTART
MINMEMORYEND
MINCPUSPEEDSTART
MINCPUSPEEDEND
MINCPUNUMBERSTART
MINCPUNUMBEREND
SERVICETYPESTART
SERVICETYPEEND
MAXTIMESTART
MAXTIMEEND
REQUESTEDSTARTTIMESTART
REQUESTEDSTARTTIMEEND
DEPENDENCIESSTART
DEPENDENCIESEND
REQUIREDDIRECTORIESSTART
REQUIREDDIRECTORIESEND
REQUIREDFILESSTART
REQUIREDFILESEND
PATHSTART
PATHEND
PATHPAIRSTART
PATHPAIREND
DIRECTORIESTOMAKESTART
DIRECTORIESTOMAKEEND

FILESTOMAKESTART
FILESTOMAKEEND
DIRECTORIESTODELETESTART
DIRECTORIESTODELETEEND
FILESTODELETESTART
FILESTODELETEEND
INPUTDIRECTORIESTOMOVESTART
INPUTDIRECTORIESTOMOVEEND
INPUTFILESTOMOVESTART
INPUTFILESTOMOVEEND
OUTPUTDIRECTORIESTOMOVESTART
OUTPUTDIRECTORIESTOMOVEEND
OUTPUTFILESTOMOVESTART
OUTPUTFILESTOMOVEEND
PROCESSESTOKILLSTART
PROCESSESTOKILLEND
PROCESSSTART
PROCESSEND
VARIABLESSTART
VARIABLESEND
INITIALVARIABLEVALUESSTART
INITIALVARIABLEVALUESEND
VARSTART
VAREND
INITIALVARIABLEVALUESSTRINGSTART
INITIALVARIABLEVALUESSTRINGEND
INITIALBASISSTATUSSTART
INITIALBASISSTATUSEND
BASICSTART
BASICEND
ATUPPERSTART
ATUPPEREND
ATLOWERSTART
ATLOWEREND
ATEQUALITYSTART
ATEQUALITYEND
SUPERBASICSTART
SUPERBASICEND
ISFREESTART
ISFREEEND
UNKNOWNSTART
UNKNOWNEND

INTEGervARIABLEBRANCHINGWEIGHTSSTART
INTEGervARIABLEBRANCHINGWEIGHTSEND
SOSVARIABLEBRANCHINGWEIGHTSSTART
SOSVARIABLEBRANCHINGWEIGHTSEND
SOSSTART
SOSEND
OBJECTIVESSTART
OBJECTIVESEND
INITIALOBJECTIVEVALUESSTART
INITIALOBJECTIVEVALUESEND
OBJSTART
OBJEND
INITIALOBJECTIVEBOUNDSSTART
INITIALOBJECTIVEBOUNDSEND
CONSTRAINTSSTART
CONSTRAINTSEND
INITIALCONSTRAINTVALUESSTART
INITIALCONSTRAINTVALUESEND
CONSTART
CONEND
INITIALDUALVALUESSTART
INITIALDUALVALUESEND
SOLVEROPTIONSSTART
SOLVEROPTIONSEND
SOLVEROPTIONSTART
SOLVEROPTIONEND
ENUMERATIONSTART
ENUMERATIONEND
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART

MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT

NUMBEROFROWSATT
NUMBEROFVALUESATT
NUMBEROFVARIDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPPOSEATT
BASETRANSPPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART

TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPEND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART

NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART

MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMSEEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMSEEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANSPOSESTART
MATRIXTRANSPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE
TWOQUOTES
GREATERTHAN
ENDOFELEMENT
OSRLSTART
OSRLSTARTEMPT
OSRLATTRIBUTETEXT
OSRLEND
NUMBEROFCONATT

NUMBEROFCONSTRAINTSATT
NUMBEROFELATT
NUMBEROFENUMERATIONSATT
NUMBEROFIDXATT
NUMBEROFITEMSATT
NUMBEROFOBJATT
NUMBEROFOBJECTIVESATT
NUMBEROFOTHERCONSTRAINTRESULTSATT
NUMBEROFOTHEROBJECTIVERESULTSATT
NUMBEROFOTHERRESULTSATT
NUMBEROFOTHERSOLUTIONRESULTSATT
NUMBEROFOTHERVARIABLERESULTSATT
NUMBEROFSOLUTIONSATT
NUMBEROFSOLVEROUTPUTSATT
NUMBEROFSUBSTATUSESATT
NUMBEROFTIMESATT
NUMBEROFVARATT
NUMBEROFVARIABLESATT
NUMBEROFVARIDXATT
TARGETOBJECTIVEIDXATT
IDXATT
INCRATT
MULTATT
SIZEOFATT
CATEGORYATT
EMPTYCATEGORYATT
DESCRIPTIONATT
EMPTYDESCRIPTIONATT
NAMEATT
EMPTYNAMEATT
TYPEATT
EMPTYTYPEATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT
OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
UNITATT
EMPTYUNITATT

VALUEATT
EMPTYVALUEATT
WEIGHTEDOBJECTIVESATT
EMPTYWEIGHTEDOBJECTIVESATT
TARGETOBJECTIVENAMEATT
EMPTYTARGETOBJECTIVENAMEATT
HEADERSTART
HEADEREND
GENERALSTART
GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND
JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
ITEMSTART
ITEMEND
ITEMSTARTANDEND
ITEMEMPTY
ACTUALSTARTTIMESTART
ACTUALSTARTTIMEEND
ATEQUALITYSTART
ATEQUALITYEND
ATLOWERSTART
ATLOWEREND
ATUPPERSTART
ATUPPEREND
AVAILABLECPUNUMBERSTART
AVAILABLECPUNUMBEREND
AVAILABLECPUSPEEDSTART
AVAILABLECPUSPEEDEND
AVAILABLEDISKSPACESTART
AVAILABLEDISKSPACEEND
AVAILABLEMEMORYSTART
AVAILABLEMEMORYEND
BASE64START
BASE64END
BASICSTART
BASICEND

BASISSTATUSSTART
BASISSTATUSEND
BASSTATUSSTART
BASSTATUSEND
CONSTART
CONEND
CONSTRAINTSSTART
CONSTRAINTSEND
CURRENTJOBCOUNTSTART
CURRENTJOBCOUNTEND
CURRENTSTATESTART
CURRENTSTATEEND
DUALVALUESSTART
DUALVALUESEND
ELSTART
ELEND
ENUMERATIONSTART
ENUMERATIONEND
ENDTIMESTART
ENDTIMEEND
GENERALSTATUSSTART
GENERALSTATUSEND
GENERALSUBSTATUSSTART
GENERALSUBSTATUSEND
IDXSTART
IDXEND
INSTANCENAMESTART
INSTANCENAMEEND
ISFREESTART
ISFREEEND
JOBIDSTART
JOBIDEND
MESSAGESTART
MESSAGEEND
OBJSTART
OBJEND
OBJECTIVESSTART
OBJECTIVESEND
OPTIMIZATIONSOLUTIONSTATUSSTART
OPTIMIZATIONSOLUTIONSTATUSEND
OPTIMIZATIONSOLUTIONSUBSTATUSSTART
OPTIMIZATIONSOLUTIONSUBSTATUSEND

OTHERSTART
OTHEREND
OTHERRESULTSSTART
OTHERRESULTSEND
OTHERSOLUTIONRESULTSTART
OTHERSOLUTIONRESULTEND
OTHERSOLUTIONRESULTSSTART
OTHERSOLUTIONRESULTSEND
OTHERSOLVEROUTPUTSTART
OTHERSOLVEROUTPUTEND
SCHEDULEDSTARTTimestart
SCHEDULEDSTARTTIMEEND
SERVICENAMESTART
SERVICENAMEEND
SERVICEURISTART
SERVICEURIEND
SERVICEUTILIZATIONSTART
SERVICEUTILIZATIONEND
SOLUTIONSTART
SOLUTIONEND
SOLVERINVOKEDSTART
SOLVERINVOKEDEND
SOLVEROUTPUTSTART
SOLVEROUTPUTEND
STATUSSTART
STATUSEND
SUBMITTimestart
SUBMITTIMEEND
SUBSTATUSSTART
SUBSTATUSEND
SUPERBASICSTART
SUPERBASICEND
SYSTEMINFORMATIONSTART
SYSTEMINFORMATIONEND
Timestart
TIMEEND
TIMESERVICESTARTEDSTART
TIMESERVICESTARTEDEND
TIMESTAMPSTART
TIMESTAMPEND
TIMINGINFORMATIONSTART
TIMINGINFORMATIONEND

TOTALJOBSSOFARSTART
TOTALJOBSSOFAREND
UNKNOWNSTART
UNKNOWNEND
USEDCPUNUMBERSTART
USEDCPUNUMBEREND
USEDCPUSPEEDSTART
USEDCPUSPEEDEND
USEDISKSPACESTART
USEDISKSPACEEND
USEDMEMORYSTART
USEDMEMORYEND
VALUESSTRINGSTART
VALUESSTRINGEND
VARSTART
VAREND
VARIABLESSTART
VARIABLESEND
VARIDXSTART
VARIDXEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND

FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPPOSEATT
BASETRANSPPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSEND

INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART

POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART

PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMESEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMESEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANSPOSESTART
MATRIXTRANSPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART

MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT

Definition at line 41 of file OSParseosol.tab.hpp.

7.36 /home/ted/COIN/trunk/OS/src/OSParsers/OSParseosol.tab.hpp File Reference

Classes

- union [YYSTYPE](#)
- struct [YYLTYPE](#)

Macros

- #define [ATTRIBUTETEXT](#) 258
- #define [ELEMENTTEXT](#) 259
- #define [ITEMTEXT](#) 260
- #define [INTEGER](#) 261
- #define [DOUBLE](#) 262
- #define [QUOTE](#) 263
- #define [TWOQUOTES](#) 264
- #define [GREATERTHAN](#) 265
- #define [ENDOFELEMENT](#) 266
- #define [OSOLSTART](#) 267
- #define [OSOLSTARTEMPT](#) 268
- #define [OSOLATTRIBUTETEXT](#) 269
- #define [SOLEND](#) 270
- #define [NUMBEROFOTHEROPTIONSATT](#) 271
- #define [NUMBEROFENUMERATIONSATT](#) 272
- #define [NUMBEROFJOBIDSATT](#) 273
- #define [NUMBEROFPATHSATT](#) 274
- #define [NUMBEROFPATHPAIRSATT](#) 275
- #define [FROMATT](#) 276
- #define [TOATT](#) 277
- #define [MAKECOPYATT](#) 278
- #define [CATEGORYATT](#) 279
- #define [TYPEATT](#) 280
- #define [GROUPWEIGHTATT](#) 281
- #define [NUMBEROFPROCESSESATT](#) 282
- #define [NUMBEROFSOLVEROPTIONSATT](#) 283
- #define [NUMBEROFSOSATT](#) 284
- #define [NUMBEROFVARIABLESATT](#) 285
- #define [NUMBEROFOBJECTIVESATT](#) 286
- #define [NUMBEROFCONSTRAINTSATT](#) 287
- #define [NUMBEROFOTHERVARIABLEOPTIONSATT](#) 288
- #define [NUMBEROFOTHEROBJECTIVEOPTIONSATT](#) 289
- #define [NUMBEROFOTHERCONSTRAINTOPTIONSATT](#) 290

- #define [NUMBEROFITEMSATT](#) 291
- #define [NUMBEROFVARATT](#) 292
- #define [NUMBEROFOBJATT](#) 293
- #define [NUMBEROFCONATT](#) 294
- #define [NUMBEROFELATT](#) 295
- #define [NAMEATT](#) 296
- #define [IDXATT](#) 297
- #define [SOSIDXATT](#) 298
- #define [VALUEATT](#) 299
- #define [UNITATT](#) 300
- #define [DESCRIPTIONATT](#) 301
- #define [CONTYPEATT](#) 302
- #define [EMPTYCONTYPEATT](#) 303
- #define [ENUMTYPEATT](#) 304
- #define [EMPTYENUMTYPEATT](#) 305
- #define [OBJTYPEATT](#) 306
- #define [EMPTYOBJTYPEATT](#) 307
- #define [VARTYPEATT](#) 308
- #define [EMPTYVARTYPEATT](#) 309
- #define [EMPTYTYPEATT](#) 310
- #define [EMPTYNAMEATT](#) 311
- #define [EMPTYCATEGORYATT](#) 312
- #define [EMPTYDESCRIPTIONATT](#) 313
- #define [EMPTYUNITATT](#) 314
- #define [EMPTYVALUEATT](#) 315
- #define [EMPTYLBVALUEATT](#) 316
- #define [EMPTYUBVALUEATT](#) 317
- #define [LBVALUEATT](#) 318
- #define [UBVALUEATT](#) 319
- #define [EMPTYLBDUALVALUEATT](#) 320
- #define [EMPTYUBDUALVALUEATT](#) 321
- #define [LBDUALVALUEATT](#) 322
- #define [UBDUALVALUEATT](#) 323
- #define [SOLVERATT](#) 324
- #define [EMPTYSOLVERATT](#) 325
- #define [WEIGHTATT](#) 326
- #define [EMPTYWEIGHTATT](#) 327
- #define [TRANSPORTTYPEATT](#) 328
- #define [LOCATIONTYPEATT](#) 329
- #define [GENERALSTART](#) 330
- #define [GENERALEND](#) 331
- #define [SYSTEMSTART](#) 332
- #define [SYSTEMEND](#) 333
- #define [SERVICESTART](#) 334
- #define [SERVICEEND](#) 335
- #define [JOBSTART](#) 336
- #define [JOBEND](#) 337
- #define [OPTIMIZATIONSTART](#) 338
- #define [OPTIMIZATIONEND](#) 339
- #define [SERVICEURISTART](#) 340
- #define [SERVICEURIEND](#) 341

- #define [SERVICENAMESTART](#) 342
- #define [SERVICENAMEEND](#) 343
- #define [INSTANCENAMESTART](#) 344
- #define [INSTANCENAMEEND](#) 345
- #define [INSTANCELOCATIONSTART](#) 346
- #define [INSTANCELOCATIONEND](#) 347
- #define [JOBIDSTART](#) 348
- #define [JOBIDEND](#) 349
- #define [SOLVERTOINVOKESTART](#) 350
- #define [SOLVERTOINVOKEEND](#) 351
- #define [LICENSESTART](#) 352
- #define [LICENSEEND](#) 353
- #define [USERNAMESTART](#) 354
- #define [USERNAMEEND](#) 355
- #define [PASSWORDSTART](#) 356
- #define [PASSWORDEND](#) 357
- #define [CONTACTSTART](#) 358
- #define [CONTACTEND](#) 359
- #define [OTHEROPTIONSSTART](#) 360
- #define [OTHEROPTIONSEND](#) 361
- #define [OTHERSTART](#) 362
- #define [OTHEREND](#) 363
- #define [MINDISKSPACESTART](#) 364
- #define [MINDISKSPACEEND](#) 365
- #define [MINMEMORYSTART](#) 366
- #define [MINMEMORYEND](#) 367
- #define [MINCPUSPEEDSTART](#) 368
- #define [MINCPUSPEEDEND](#) 369
- #define [MINCPUNUMBERSTART](#) 370
- #define [MINCPUNUMBEREND](#) 371
- #define [SERVICETYPESTART](#) 372
- #define [SERVICETYPEEND](#) 373
- #define [MAXTimestart](#) 374
- #define [MAXTIMEEND](#) 375
- #define [REQUESTEDSTARTTIMESTART](#) 376
- #define [REQUESTEDSTARTTIMEEND](#) 377
- #define [DEPENDENCIESSTART](#) 378
- #define [DEPENDENCIESEND](#) 379
- #define [REQUIREDDIRECTORIESSTART](#) 380
- #define [REQUIREDDIRECTORIESEND](#) 381
- #define [REQUIREDFILESSTART](#) 382
- #define [REQUIREDFILESEND](#) 383
- #define [PATHSTART](#) 384
- #define [PATHEND](#) 385
- #define [PATHPAIRSTART](#) 386
- #define [PATHPAIREND](#) 387
- #define [DIRECTORIESTOMAKESTART](#) 388
- #define [DIRECTORIESTOMAKEEND](#) 389
- #define [FILESTOMAKESTART](#) 390
- #define [FILESTOMAKEEND](#) 391
- #define [DIRECTORIESTODELETESTART](#) 392

- #define [DIRECTORIESTODELETEEND](#) 393
- #define [FILESTODELETESTART](#) 394
- #define [FILESTODELETEEND](#) 395
- #define [INPUTDIRECTORIESTOMOVESTART](#) 396
- #define [INPUTDIRECTORIESTOMOVEEND](#) 397
- #define [INPUTFILESTOMOVESTART](#) 398
- #define [INPUTFILESTOMOVEEND](#) 399
- #define [OUTPUTDIRECTORIESTOMOVESTART](#) 400
- #define [OUTPUTDIRECTORIESTOMOVEEND](#) 401
- #define [OUTPUTFILESTOMOVESTART](#) 402
- #define [OUTPUTFILESTOMOVEEND](#) 403
- #define [PROCESSESTOKILLSTART](#) 404
- #define [PROCESSESTOKILLEND](#) 405
- #define [PROCESSSTART](#) 406
- #define [PROCESSEND](#) 407
- #define [VARIABLESSTART](#) 408
- #define [VARIABLESEND](#) 409
- #define [INITIALVARIABLEVALUESSTART](#) 410
- #define [INITIALVARIABLEVALUESSEND](#) 411
- #define [VARSTART](#) 412
- #define [VAREND](#) 413
- #define [INITIALVARIABLEVALUESSTRINGSTART](#) 414
- #define [INITIALVARIABLEVALUESSTRINGEND](#) 415
- #define [INITIALBASISSTATUSSTART](#) 416
- #define [INITIALBASISSTATUSSEND](#) 417
- #define [BASICSTART](#) 418
- #define [BASICEND](#) 419
- #define [ATUPPERSTART](#) 420
- #define [ATUPPEREND](#) 421
- #define [ATLOWERSTART](#) 422
- #define [ATLOWEREND](#) 423
- #define [ATEQUALITYSTART](#) 424
- #define [ATEQUALITYEND](#) 425
- #define [SUPERBASICSTART](#) 426
- #define [SUPERBASICEND](#) 427
- #define [ISFREESTART](#) 428
- #define [ISFREEEND](#) 429
- #define [UNKNOWNSTART](#) 430
- #define [UNKNOWNEND](#) 431
- #define [INTEGERVARIABLEBRANCHINGWEIGHTSSTART](#) 432
- #define [INTEGERVARIABLEBRANCHINGWEIGHTSEND](#) 433
- #define [SOSVARIABLEBRANCHINGWEIGHTSSTART](#) 434
- #define [SOSVARIABLEBRANCHINGWEIGHTSEND](#) 435
- #define [SOSSTART](#) 436
- #define [SOSEND](#) 437
- #define [OBJECTIVESSTART](#) 438
- #define [OBJECTIVESEND](#) 439
- #define [INITIALOBJECTIVEVALUESSTART](#) 440
- #define [INITIALOBJECTIVEVALUESSEND](#) 441
- #define [OBJSTART](#) 442
- #define [OBJEND](#) 443

- #define INITIALOBJECTIVEBOUNDSSTART 444
- #define INITIALOBJECTIVEBOUNDSEND 445
- #define CONSTRAINTSSTART 446
- #define CONSTRAINTSEND 447
- #define INITIALCONSTRAINTVALUESSTART 448
- #define INITIALCONSTRAINTVALUESEND 449
- #define CONSTART 450
- #define CONEND 451
- #define INITIALDUALVALUESSTART 452
- #define INITIALDUALVALUESEND 453
- #define SOLVEROPTIONSSTART 454
- #define SOLVEROPTIONSEND 455
- #define SOLVEROPTIONSTART 456
- #define SOLVEROPTIONEND 457
- #define ENUMERATIONSTART 458
- #define ENUMERATIONEND 459
- #define ITEMEMPTY 460
- #define ITEMSTART 461
- #define ITEMEND 462
- #define ITEMSTARTANDEND 463
- #define BASE64START 464
- #define BASE64END 465
- #define INCRATT 466
- #define MULTATT 467
- #define SIZEOFATT 468
- #define ELSTART 469
- #define ELEND 470
- #define MATRIXVARSTART 471
- #define MATRIXVAREND 472
- #define MATRIXOBJSTART 473
- #define MATRIXOBJEND 474
- #define MATRIXCONSTART 475
- #define MATRIXCONEND 476
- #define HEADERSTART 477
- #define HEADEREND 478
- #define FILENAMESTART 479
- #define FILENAMEEND 480
- #define FILENAMEEMPTY 481
- #define FILENAMESTARTANDEND 482
- #define FILESOURCESTART 483
- #define FILESOURCEEND 484
- #define FILESOURCEEMPTY 485
- #define FILESOURCESTARTANDEND 486
- #define FILEDESCRIPTIONSTART 487
- #define FILEDESCRIPTIONEND 488
- #define FILEDESCRIPTIONEMPTY 489
- #define FILEDESCRIPTIONSTARTANDEND 490
- #define FILECREATORSTART 491
- #define FILECREATOREND 492
- #define FILECREATOREMPTY 493
- #define FILECREATORSTARTANDEND 494

- #define FILELICENCESTART 495
- #define FILELICENCEEND 496
- #define FILELICENCEEMPTY 497
- #define FILELICENCESTARTANDEND 498
- #define MATRIXSTART 499
- #define MATRIXEND 500
- #define BASEMATRIXEND 501
- #define BASEMATRIXSTART 502
- #define BLOCKSTART 503
- #define BLOCKEND 504
- #define BLOCKSSTART 505
- #define BLOCKSEND 506
- #define EMPTYSHAPEATT 507
- #define SHAPEATT 508
- #define EMPTYSYMMETRYATT 509
- #define SYMMETRYATT 510
- #define EMPTYNEGATIVEPATTERNATT 511
- #define NEGATIVEPATTERNATT 512
- #define CONSTANTATT 513
- #define NUMBEROFBLOCKSATT 514
- #define NUMBEROFCOLUMNSATT 515
- #define NUMBEROFROWSATT 516
- #define NUMBEROFVALUESATT 517
- #define NUMBEROFVARIDXATT 518
- #define COEFATT 519
- #define BASEMATRIXIDXATT 520
- #define TARGETMATRIXFIRSTROWATT 521
- #define TARGETMATRIXFIRSTCOLATT 522
- #define BASEMATRIXSTARTROWATT 523
- #define BASEMATRIXSTARTCOLATT 524
- #define BASEMATRIXENDROWATT 525
- #define BASEMATRIXENDCOLATT 526
- #define SCALARMULTIPLIERATT 527
- #define EMPTYBASETRANSPOSEATT 528
- #define BASETRANSPOSEATT 529
- #define ELEMENTSSTART 530
- #define ELEMENTSEND 531
- #define CONSTANTELEMENTSSTART 532
- #define CONSTANTELEMENTSEND 533
- #define STARTVECTORSTART 534
- #define STARTVECTOREND 535
- #define NONZEROSSTART 536
- #define NONZEROSEND 537
- #define INDEXESSTART 538
- #define INDEXESEND 539
- #define VALUESSTART 540
- #define VALUESEND 541
- #define VARREFERENCEELEMENTSSTART 542
- #define VARREFERENCEELEMENTSEND 543
- #define LINEARELEMENTSSTART 544
- #define LINEARELEMENTSEND 545

- #define [GENERALELEMENTSSTART](#) 546
- #define [GENERALELEMENTSEND](#) 547
- #define [CONREFERENCEELEMENTSSTART](#) 548
- #define [CONREFERENCEELEMENTSEND](#) 549
- #define [VALUETYPEATT](#) 550
- #define [OBJREFERENCEELEMENTSSTART](#) 551
- #define [OBJREFERENCEELEMENTSEND](#) 552
- #define [PATTERNELEMENTSSTART](#) 553
- #define [PATTERNELEMENTSEND](#) 554
- #define [VARIDXSTART](#) 555
- #define [VARIDXEND](#) 556
- #define [TRANSFORMATIONSTART](#) 557
- #define [TRANSFORMATIONEND](#) 558
- #define [COLOFFSETSSTART](#) 559
- #define [COLOFFSETSEND](#) 560
- #define [ROWOFFSETSSTART](#) 561
- #define [ROWOFFSETSEND](#) 562
- #define [EMPTYROWMAJORATT](#) 563
- #define [ROWMAJORATT](#) 564
- #define [BLOCKROWIDXATT](#) 565
- #define [BLOCKCOLIDXATT](#) 566
- #define [DUMMY](#) 567
- #define [NONLINEAREXPRESSIONSSTART](#) 568
- #define [NONLINEAREXPRESSIONSEND](#) 569
- #define [NUMBEROFNONLINEAREXPRESSIONS](#) 570
- #define [NLSTART](#) 571
- #define [NLEND](#) 572
- #define [MATRIXEXPRESSIONSSTART](#) 573
- #define [MATRIXEXPRESSIONSEND](#) 574
- #define [NUMBEROFEXPR](#) 575
- #define [EXPRSTART](#) 576
- #define [EXPREND](#) 577
- #define [NUMBEROFMATRIXTERMSATT](#) 578
- #define [MATRIXTERMSTART](#) 579
- #define [MATRIXTERMEND](#) 580
- #define [POWERSTART](#) 581
- #define [POWEREND](#) 582
- #define [PLUSSTART](#) 583
- #define [PLUSEND](#) 584
- #define [MINUSSTART](#) 585
- #define [MINUSEND](#) 586
- #define [DIVIDESTART](#) 587
- #define [DIVIDEEND](#) 588
- #define [LNSTART](#) 589
- #define [LNEND](#) 590
- #define [SQRTSTART](#) 591
- #define [SQRTEND](#) 592
- #define [SUMSTART](#) 593
- #define [SUMEND](#) 594
- #define [PRODUCTSTART](#) 595
- #define [PRODUCTEND](#) 596

- #define [EXPSTART](#) 597
- #define [EXPEND](#) 598
- #define [NEGATESTART](#) 599
- #define [NEGATEEND](#) 600
- #define [IFSTART](#) 601
- #define [IFEND](#) 602
- #define [SQUARESTART](#) 603
- #define [SQUAREEND](#) 604
- #define [COSSTART](#) 605
- #define [COSEND](#) 606
- #define [SINSTART](#) 607
- #define [SINEND](#) 608
- #define [VARIABLESTART](#) 609
- #define [VARIABLEEND](#) 610
- #define [ABSSTART](#) 611
- #define [ABSEND](#) 612
- #define [ERFSTART](#) 613
- #define [ERFEND](#) 614
- #define [MAXSTART](#) 615
- #define [MAXEND](#) 616
- #define [ALLDIFFSTART](#) 617
- #define [ALLDIFFEND](#) 618
- #define [MINSTART](#) 619
- #define [MINEND](#) 620
- #define [ESTART](#) 621
- #define [EEND](#) 622
- #define [PISTART](#) 623
- #define [PIEND](#) 624
- #define [TIMESSTART](#) 625
- #define [TIMESEND](#) 626
- #define [NUMBERSTART](#) 627
- #define [NUMBEREND](#) 628
- #define [MATRIXDETERMINANTSTART](#) 629
- #define [MATRIXDETERMINANTEND](#) 630
- #define [MATRIXTRACESTART](#) 631
- #define [MATRIXTRACEEND](#) 632
- #define [MATRIXTOSCALARSTART](#) 633
- #define [MATRIXTOSCALAREND](#) 634
- #define [MATRIXDIAGONALSTART](#) 635
- #define [MATRIXDIAGONALEND](#) 636
- #define [MATRIXDOTTIMESSTART](#) 637
- #define [MATRIXDOTTIMESEND](#) 638
- #define [MATRIXLOWERTRIANGLESTART](#) 639
- #define [MATRIXLOWERTRIANGLEEND](#) 640
- #define [MATRIXUPPERTRIANGLESTART](#) 641
- #define [MATRIXUPPERTRIANGLEEND](#) 642
- #define [MATRIXMERGESTART](#) 643
- #define [MATRIXMERGEEND](#) 644
- #define [MATRIXMINUSSTART](#) 645
- #define [MATRIXMINUSEND](#) 646
- #define [MATRIXNEGATESTART](#) 647

- #define [MATRIXNEGATEEND](#) 648
- #define [MATRIXPLUSSTART](#) 649
- #define [MATRIXPLUSEND](#) 650
- #define [MATRIXTIMESSTART](#) 651
- #define [MATRIXTIMSEEND](#) 652
- #define [MATRIXPRODUCTSTART](#) 653
- #define [MATRIXPRODUCTEND](#) 654
- #define [MATRIXSCALARTIMESSTART](#) 655
- #define [MATRIXSCALARTIMSEEND](#) 656
- #define [MATRIXSUBMATRIXATSTART](#) 657
- #define [MATRIXSUBMATRIXATEND](#) 658
- #define [MATRIXTRANSPOSESTART](#) 659
- #define [MATRIXTRANSPOSEEND](#) 660
- #define [MATRIXREFERENCESTART](#) 661
- #define [MATRIXREFERENCEEND](#) 662
- #define [IDENTITYMATRIXSTART](#) 663
- #define [IDENTITYMATRIXEND](#) 664
- #define [MATRIXINVERSESTART](#) 665
- #define [MATRIXINVERSEEND](#) 666
- #define [EMPTYINCLUDEDIAGONALATT](#) 667
- #define [INCLUDEDIAGONALATT](#) 668
- #define [IDATT](#) 669
- #define [YYSTYPE_IS_TRIVIAL](#) 1
- #define [yystype YYSYPE](#) /* obsolescent; will be withdrawn */
- #define [YYSTYPE_IS_DECLARED](#) 1
- #define [yyltype YYLTYPE](#) /* obsolescent; will be withdrawn */
- #define [YYLTYPE_IS_DECLARED](#) 1
- #define [YYLTYPE_IS_TRIVIAL](#) 1

Typedefs

- typedef union [YYSTYPE](#) YYSYPE
- typedef struct [YYLTYPE](#) YYLTYPE

Enumerations

- enum [yytokentype](#) {
[QUOTE](#) = 258, [ATTRIBUTETEXT](#) = 259, [ELEMENTTEXT](#) = 260, [ITEMTEXT](#) = 261,
[INTEGER](#) = 262, [DOUBLE](#) = 263, [TWOQUOTES](#) = 264, [ENDOFELEMENT](#) = 265,
[GREATERTHAN](#) = 266, [OSILEND](#) = 267, [INSTANCEDATAEND](#) = 268, [INSTANCEDATASTARTEND](#) = 269,
[EMPTYIDATT](#) = 270, [IDXONEATT](#) = 271, [IDXTWOATT](#) = 272, [VALUEATT](#) = 273,
[QUADRATICCOEFFICIENTSSTART](#) = 274, [QUADRATICCOEFFICIENTSEND](#) = 275, [NUMBEROFQTERMSA-](#)
[TT](#) = 276, [QTERMSTART](#) = 277,
[QTERMEND](#) = 278, [MATRICESSTART](#) = 279, [MATRICESEND](#) = 280, [NUMBEROFMATRICESATT](#) = 281,
[CONESSTART](#) = 282, [CONESEND](#) = 283, [NUMBEROFCONESATT](#) = 284, [NONNEGATIVECONESTART](#) =

285,
 NONNEGATIVECONEEND = 286, NONPOSITIVECONESTART = 287, NONPOSITIVECONEEND = 288, ORTHANTCONESTART = 289,
 ORTHANTCONEEND = 290, POLYHEDRALCONESTART = 291, POLYHEDRALCONEEND = 292, QUADRATICCONESTART = 293,
 QUADRATICCONEEND = 294, ROTATEDQUADRATICCONESTART = 295, ROTATEDQUADRATICCONEEND = 296, SEMIDEFINITECONESTART = 297,
 SEMIDEFINITECONEEND = 298, PRODUCTCONESTART = 299, PRODUCTCONEEND = 300, INTERSECTIONCONESTART = 301,
 INTERSECTIONCONEEND = 302, DUALCONESTART = 303, DUALCONEEND = 304, POLARCONESTART = 305,
 POLARCONEEND = 306, DIRECTIONSTART = 307, DIRECTIONEND = 308, FACTORSSTART = 309, FACTORSEND = 310, COMPONENTSSTART = 311, COMPONENTSEND = 312, NORMSCALEFACTORATT = 313,
 DISTORTIONMATRIXIDXATT = 314, AXISDIRECTIONATT = 315, FIRSTAXISDIRECTIONATT = 316, SECONDAXISDIRECTIONATT = 317,
 EMPTYSEMIDEFINITENESSATT = 318, SEMIDEFINITENESSATT = 319, REFERENCEMATRIXIDXATT = 320, MATRIXPROGRAMMINGSTART = 321,
 MATRIXPROGRAMMINGEND = 322, VARTYPEATT = 323, MATRIXVARIABLESSTART = 324, MATRIXVARIABLESEND = 325,
 NUMBEROFMATRIXVARATT = 326, MATRIXVARSTART = 327, MATRIXVAREND = 328, MATRIXOBJECTIVESSTART = 329,
 MATRIXOBJECTIVESEND = 330, NUMBEROFMATRIXOBJATT = 331, MATRIXOBJSTART = 332, MATRIXOBJEND = 333,
 MATRIXCONSTRAINTSSTART = 334, MATRIXCONSTRAINTSEND = 335, NUMBEROFMATRIXCONATT = 336, MATRIXCONSTART = 337,
 MATRIXCONEND = 338, MATRIXIDXATT = 339, LBMATRIXIDXATT = 340, LBCONEIDXATT = 341, UBMATRIXIDXATT = 342, UBCONEIDXATT = 343, TEMPLATEMATRIXIDXATT = 344, VARREFERENCEMATRIXIDXATT = 345,
 OBJREFERENCEMATRIXIDXATT = 346, CONREFERENCEMATRIXIDXATT = 347, ORDERCONEIDXATT = 348, CONSTANTMATRIXIDXATT = 349,
 TIMEDOMAINSTART = 350, TIMEDOMAINEND = 351, STAGESSTART = 352, STAGESEND = 353, STAGESTART = 354, STAGEEND = 355, NUMBEROFSTAGESATT = 356, HORIZONATT = 357,
 STARTATT = 358, VARIABLESSTART = 359, CONSTRAINTSSTART = 360, OBJECTIVESSTART = 361, VARIABLESEND = 362, CONSTRAINTSEND = 363, OBJECTIVESEND = 364, NUMBEROFVARIABLESATT = 365,
 NUMBEROFCONSTRAINTSATT = 366, NUMBEROFOBJECTIVESATT = 367, STARTIDXATT = 368, VARSTART = 369,
 VAREND = 370, CONSTART = 371, CONEND = 372, OBJSTART = 373, OBJEND = 374, INTERVALSTART = 375, INTERVALEND = 376, HEADERSTART = 377,
 HEADEREND = 378, FILENAMESTART = 379, FILENAMEEND = 380, FILENAMEEMPTY = 381, FILENAMESTARTANDEND = 382, FILESOURCESTART = 383, FILESOURCEEND = 384, FILESOURCEEMPTY = 385,
 FILESOURCESTARTANDEND = 386, FILEDESCRIPTIONSTART = 387, FILEDESCRIPTIONEND = 388, FILEDESCRIPTIONEMPTY = 389,
 FILEDESCRIPTIONSTARTANDEND = 390, FILECREATORSTART = 391, FILECREATOREND = 392, FILECREATOREMPTY = 393,
 FILECREATORSTARTANDEND = 394, FILELICENCESTART = 395, FILELICENCEEND = 396, FILELICENCEEMPTY = 397,
 FILELICENCESTARTANDEND = 398, ENUMERATIONSTART = 399, ENUMERATIONEND = 400, NUMBEROF

```

FELATT = 401,
ITEMEMPTY = 402, ITEMSTART = 403, ITEMEND = 404, ITEMSTARTANDEND = 405,
BASE64START = 406, BASE64END = 407, INCRATT = 408, MULTATT = 409,
SIZEOFATT = 410, ELSTART = 411, ELEND = 412, MATRIXSTART = 413,
MATRIXEND = 414, BASEMATRIXEND = 415, BASEMATRIXSTART = 416, BLOCKSTART = 417,
BLOCKEND = 418, BLOCKSSTART = 419, BLOCKSEND = 420, EMPTYNAMEATT = 421,
NAMEATT = 422, EMPTYTYPEATT = 423, TYPEATT = 424, EMPTYSHAPEATT = 425,
SHAPEATT = 426, EMPTYSYMMETRYATT = 427, SYMMETRYATT = 428, EMPTYNEGATIVEPATTERNATT =
429,
NEGATIVEPATTERNATT = 430, CONSTANTATT = 431, NUMBEROFBLOCKSATT = 432, NUMBEROFCOLU-
MNSATT = 433,
NUMBEROFROWSATT = 434, NUMBEROFVALUESATT = 435, NUMBEROFVARIDXATT = 436, IDXATT =
437,
COEFATT = 438, BASEMATRIXIDXATT = 439, TARGETMATRIXFIRSTROWATT = 440, TARGETMATRIXFIR-
STCOLATT = 441,
BASEMATRIXSTARTROWATT = 442, BASEMATRIXSTARTCOLATT = 443, BASEMATRIXENDROWATT =
444, BASEMATRIXENDCOLATT = 445,
SCALARMULTIPLIERATT = 446, EMPTYBASETRANPOSEATT = 447, BASETRANPOSEATT = 448, ELEM-
ENTSSTART = 449,
ELEMENTSEND = 450, CONSTANTELEMENTSSTART = 451, CONSTANTELEMENTSEND = 452, STARTVE-
CTORSTART = 453,
STARTVECTOREND = 454, NONZEROSSTART = 455, NONZEROSSEND = 456, INDEXESSTART = 457,
INDEXESEND = 458, VALUESSTART = 459, VALUESEND = 460, VARREFERENCEELEMENTSSTART = 461,
VARREFERENCEELEMENTSEND = 462, LINEARELEMENTSSTART = 463, LINEARELEMENTSEND = 464,
GENERALELEMENTSSTART = 465,
GENERALELEMENTSEND = 466, CONREFERENCEELEMENTSSTART = 467, CONREFERENCEELEMENT-
SEND = 468, VALUETYPEATT = 469,
OBJREFERENCEELEMENTSSTART = 470, OBJREFERENCEELEMENTSEND = 471, PATTERNELEMENTS-
START = 472, PATTERNELEMENTSEND = 473,
VARIDXSTART = 474, VARIDXEND = 475, TRANSFORMATIONSTART = 476, TRANSFORMATIONEND = 477,
COLOFFSETSSTART = 478, COLOFFSETSEND = 479, ROWOFFSETSSTART = 480, ROWOFFSETSEND =
481,
EMPTYROWMAJORATT = 482, ROWMAJORATT = 483, BLOCKROWIDXATT = 484, BLOCKCOLIDXATT =
485,
DUMMY = 486, NONLINEAREXPRESSIONSSTART = 487, NONLINEAREXPRESSIONSEND = 488, NUMBE-
ROFNONLINEAREXPRESSIONS = 489,
NLSTART = 490, NLEND = 491, MATRIXEXPRESSIONSSTART = 492, MATRIXEXPRESSIONSEND = 493,
NUMBEROFEXPR = 494, EXPRSTART = 495, EXPREND = 496, NUMBEROFMATRIXTERMSATT = 497,
MATRIXTERMSTART = 498, MATRIXTERMEND = 499, POWERSTART = 500, POWEREND = 501,
PLUSSTART = 502, PLUSEND = 503, MINUSSTART = 504, MINUSEND = 505,
DIVIDESTART = 506, DIVIDEEND = 507, LNSTART = 508, LNEEND = 509,
SQRTSTART = 510, SQRTEND = 511, SUMSTART = 512, SUMEND = 513,
PRODUCTSTART = 514, PRODUCTEND = 515, EXPSTART = 516, EXPEND = 517,
NEGATESTART = 518, NEGATEEND = 519, IFSTART = 520, IFEND = 521,
SQUARESTART = 522, SQUAREEND = 523, COSSTART = 524, COSEND = 525,
SINSTART = 526, SINEND = 527, VARIABLESTART = 528, VARIABLEEND = 529,
ABSSTART = 530, ABSEND = 531, ERFSTART = 532, ERFEND = 533,
MAXSTART = 534, MAXEND = 535, ALLDIFFSTART = 536, ALLDIFFEND = 537,
MINSTART = 538, MINEND = 539, ESTART = 540, EEND = 541,
PISTART = 542, PIEND = 543, TIMESSTART = 544, TIMESEND = 545,
NUMBERSTART = 546, NUMBEREND = 547, MATRIXDETERMINANTSTART = 548, MATRIXDETERMINANT-
END = 549,
MATRIXTRACESTART = 550, MATRIXTRACEEND = 551, MATRIXTOSCALARSTART = 552, MATRIXTOSCA-

```


LAREND = 553,
 MATRIXDIAGONALSTART = 554, MATRIXDIAGONALEND = 555, MATRIXDOTTIMESSTART = 556, MATRIXDOTTIMESEND = 557,
 MATRIXLOWERTRIANGLESTART = 558, MATRIXLOWERTRIANGLEEND = 559, MATRIXUPPERTRIANGLESTART = 560, MATRIXUPPERTRIANGLEEND = 561,
 MATRIXMERGESTART = 562, MATRIXMERGEEND = 563, MATRIXMINUSSTART = 564, MATRIXMINUSEND = 565,
 MATRIXNEGATESTART = 566, MATRIXNEGATEEND = 567, MATRIXPLUSSTART = 568, MATRIXPLUSEND = 569,
 MATRITIMESSTART = 570, MATRITIMESEND = 571, MATRIXPRODUCTSTART = 572, MATRIXPRODUCTEND = 573,
 MATRIXSCALARTIMESSTART = 574, MATRIXSCALARTIMESEND = 575, MATRIXSUBMATRIXATSTART = 576, MATRIXSUBMATRIXATEND = 577,
 MATRIXTRANPOSESTART = 578, MATRIXTRANPOSEEND = 579, MATRIXREFERENCESTART = 580, MATRIXREFERENCEEND = 581,
 IDENTITYMATRIXSTART = 582, IDENTITYMATRIXEND = 583, MATRIXINVERSESTART = 584, MATRIXINVERSEEND = 585,
 EMPTYINCLUDEDIAGONALATT = 586, INCLUDEDIAGONALATT = 587, IDATT = 588, ATTRIBUTETEXT = 258,
 ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
 QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
 OSOLSTART = 267, OSOLSTARTEMPT = 268, OSOLATTRIBUTETEXT = 269, OSOLEND = 270,
 NUMBEROFOTHEROPTIONSATT = 271, NUMBEROFENUMERATIONSATT = 272, NUMBEROFJOBIDSATT = 273, NUMBEROFPATHSATT = 274,
 NUMBEROFPATHPAIRSATT = 275, FROMATT = 276, TOATT = 277, MAKECOPYATT = 278,
 CATEGORYATT = 279, TYPEATT = 280, GROUPWEIGHTATT = 281, NUMBEROFPROCESSESATT = 282, NUMBEROF SOLVEROPTIONSATT = 283, NUMBEROF SOSATT = 284, NUMBEROFVARIABLESATT = 285, NUMBEROF OBJECTIVESATT = 286,
 NUMBEROFCONSTRAINTSATT = 287, NUMBEROFOTHERVARIABLEOPTIONSATT = 288, NUMBEROFOTHEROBJECTIVEOPTIONSATT = 289, NUMBEROFOTHERCONSTRAINTOPTIONSATT = 290,
 NUMBEROFITEMSATT = 291, NUMBEROFVARATT = 292, NUMBEROF OBJATT = 293, NUMBEROFCONATT = 294,
 NUMBEROFELATT = 295, NAMEATT = 296, IDXATT = 297, SOSIDXATT = 298,
 VALUEATT = 299, UNITATT = 300, DESCRIPTIONATT = 301, CONTYPEATT = 302,
 EMPTYCONTYPEATT = 303, ENUMTYPEATT = 304, EMPTYENUMTYPEATT = 305, OBJTYPEATT = 306, EMPTYOBJTYPEATT = 307, VARTYPEATT = 308, EMPTYVARTYPEATT = 309, EMPTYTYPEATT = 310,
 EMPTYNAMEATT = 311, EMPTYCATEGORYATT = 312, EMPTYDESCRIPTIONATT = 313, EMPTYUNITATT = 314,
 EMPTYVALUEATT = 315, EMPTYLBVALUEATT = 316, EMPTYUBVALUEATT = 317, LBVALUEATT = 318, UBVALUEATT = 319, EMPTYLBDUALVALUEATT = 320, EMPTYUBDUALVALUEATT = 321, LBDUALVALUEATT = 322,
 UBDUALVALUEATT = 323, SOLVERATT = 324, EMPTYSOLVERATT = 325, WEIGHTATT = 326, EMPTYWEIGHTATT = 327, TRANSPORTTYPEATT = 328, LOCATIONTYPEATT = 329, GENERALSTART = 330,
 GENERALEND = 331, SYSTEMSTART = 332, SYSTEMEND = 333, SERVICESTART = 334, SERVICEEND = 335, JOBSTART = 336, JOBEND = 337, OPTIMIZATIONSTART = 338, OPTIMIZATIONEND = 339, SERVICEURISTART = 340, SERVICEURIEND = 341, SERVICENAMESTART = 342,
 SERVICENAMEEND = 343, INSTANCENAMESTART = 344, INSTANCENAMEEND = 345, INSTANCELOCATI-

ONSTART = 346,
INSTANCELOCATIONEND = 347, JOBIDSTART = 348, JOBIDEND = 349, SOLVERTOINVOKESTART = 350,
SOLVERTOINVOKEEND = 351, LICENSESTART = 352, LICENSEEND = 353, USERNAMESTART = 354,
USERNAMEEND = 355, PASSWORDSTART = 356, PASSWORDEND = 357, CONTACTSTART = 358,
CONTACTEND = 359, OTHEROPTIONSSTART = 360, OTHEROPTIONSSEND = 361, OTHERSTART = 362,
OTHEREND = 363, MINDISKSPACESTART = 364, MINDISKSPACEEND = 365, MINMEMORYSTART = 366,
MINMEMORYEND = 367, MINCPUSPEEDSTART = 368, MINCPUSPEEDEND = 369, MINCPUNUMBERSTA-
RT = 370,
MINCPUNUMBEREND = 371, SERVICYPESTART = 372, SERVICYPEEND = 373, MAXTIMESTART =
374,
MAXTIMEEND = 375, REQUESTEDSTARTTIMESTART = 376, REQUESTEDSTARTTIMEEND = 377, DEPEND-
ENCIESSTART = 378,
DEPENDENCIESEND = 379, REQUIREDIRECTORIESSTART = 380, REQUIREDIRECTORIESEND = 381,
REQUIREDFILESSTART = 382,
REQUIREDFILESEND = 383, PATHSTART = 384, PATHEND = 385, PATHPAIRSTART = 386,
PATHPAIREND = 387, DIRECTORIESTOMAKESTART = 388, DIRECTORIESTOMAKEEND = 389, FILESTO-
MAKESTART = 390,
FILESTOMAKEEND = 391, DIRECTORIESTODELETESTART = 392, DIRECTORIESTODELETEEND = 393, F-
ILESTODELETESTART = 394,
FILESTODELETEEND = 395, INPUTDIRECTORIESTOMOVESTART = 396, INPUTDIRECTORIESTOMOVEE-
ND = 397, INPUTFILESTOMOVESTART = 398,
INPUTFILESTOMOVEEND = 399, OUTPUTDIRECTORIESTOMOVESTART = 400, OUTPUTDIRECTORIEST-
OMOVEEND = 401, OUTPUTFILESTOMOVESTART = 402,
OUTPUTFILESTOMOVEEND = 403, PROCESSESTOKILLSTART = 404, PROCESSESTOKILLEND = 405, P-
ROCESSSTART = 406,
PROCESSEND = 407, VARIABLESSTART = 408, VARIABLESEND = 409, INITIALVARIABLEVALUESSTART
= 410,
INITIALVARIABLEVALUESSEND = 411, VARSTART = 412, VAREND = 413, INITIALVARIABLEVALUESSTRIN-
GSTART = 414,
INITIALVARIABLEVALUESSTRINGEND = 415, INITIALBASISSTATUSSTART = 416, INITIALBASISSTATUSE-
ND = 417, BASICSTART = 418,
BASICEND = 419, ATUPPERSTART = 420, ATUPPEREND = 421, ATLOWERSTART = 422,
ATLOWEREND = 423, ATEQUALITYSTART = 424, ATEQUALITYEND = 425, SUPERBASICSTART = 426,
SUPERBASICEND = 427, ISFREESTART = 428, ISFREEEND = 429, UNKNOWNSTART = 430,
UNKNOWNEND = 431, INTEGERVARIABLEBRANCHINGWEIGHTSSTART = 432, INTEGERVARIABLEBRAN-
CHINGWEIGHTSEND = 433, SOSVARIABLEBRANCHINGWEIGHTSSTART = 434,
SOSVARIABLEBRANCHINGWEIGHTSEND = 435, SOSSTART = 436, SOSEND = 437, OBJECTIVESSTART =
438,
OBJECTIVESEND = 439, INITIALOBJECTIVEVALUESSTART = 440, INITIALOBJECTIVEVALUESEND = 441,
OBJSTART = 442,
OBJEND = 443, INITIALOBJECTIVEBOUNDSSTART = 444, INITIALOBJECTIVEBOUNDSEND = 445, CONST-
RAINTSSTART = 446,
CONSTRAINTSEND = 447, INITIALCONSTRAINTVALUESSTART = 448, INITIALCONSTRAINTVALUESEND
= 449, CONSTART = 450,
CONEND = 451, INITIALDUALVALUESSTART = 452, INITIALDUALVALUESEND = 453, SOLVEROPTIONSS-
TART = 454,
SOLVEROPTIONSEND = 455, SOLVEROPTIONSTART = 456, SOLVEROPTIONEND = 457, ENUMERATION-

START = 458,
ENUMERATIONEND = 459, ITEMEMPTY = 460, ITEMSTART = 461, ITEMEND = 462,
ITEMSTARTANDEND = 463, BASE64START = 464, BASE64END = 465, INCRATT = 466,
MULTATT = 467, SIZEOFATT = 468, ELSTART = 469, ELEND = 470,
MATRIXVARSTART = 471, MATRIXVAREND = 472, MATRIXOBJSTART = 473, MATRIXOBJEND = 474,
MATRIXCONSTART = 475, MATRIXCONEND = 476, HEADERSTART = 477, HEADEREND = 478,
FILENAMESTART = 479, FILENAMEEND = 480, FILENAMEEMPTY = 481, FILENAMESTARTANDEND = 482,
FILESOURCESTART = 483, FILESOURCEEND = 484, FILESOURCEEMPTY = 485, FILESOURCESTARTANDEND = 486,
FILEDESCRIPTIONSTART = 487, FILEDESCRIPTIONEND = 488, FILEDESCRIPTIONEMPTY = 489, FILEDESCRIPTIONSTARTANDEND = 490,
FILECREATORSTART = 491, FILECREATOREND = 492, FILECREATOREMPTY = 493, FILECREATORSTARTANDEND = 494,
FILELICENCESTART = 495, FILELICENCEEND = 496, FILELICENCEEMPTY = 497, FILELICENCESTARTANDEND = 498,
MATRIXSTART = 499, MATRIXEND = 500, BASEMATRIXEND = 501, BASEMATRIXSTART = 502,
BLOCKSTART = 503, BLOCKEND = 504, BLOCKSSTART = 505, BLOCKSEND = 506,
EMPTYSHAPEATT = 507, SHAPEATT = 508, EMPTYSYMMETRYATT = 509, SYMMETRYATT = 510,
EMPTYNEGATIVEPATTERNATT = 511, NEGATIVEPATTERNATT = 512, CONSTANTATT = 513, NUMBEROFBLOCKSATT = 514,
NUMBEROFCOLUMNSATT = 515, NUMBEROFROWSATT = 516, NUMBEROFVALUESATT = 517, NUMBEROFVARIDXATT = 518,
COEFATT = 519, BASEMATRIXIDXATT = 520, TARGETMATRIXFIRSTROWATT = 521, TARGETMATRIXFIRSTCOLATT = 522,
BASEMATRIXSTARTROWATT = 523, BASEMATRIXSTARTCOLATT = 524, BASEMATRIXENDROWATT = 525, BASEMATRIXENDCOLATT = 526,
SCALARMULTIPLIERATT = 527, EMPTYBASETRANSPOSEATT = 528, BASETRANSPOSEATT = 529, ELEMENTSSTART = 530,
ELEMENTSEND = 531, CONSTANTELEMENTSSTART = 532, CONSTANTELEMENTSEND = 533, STARTVECTORSTART = 534,
STARTVECTOREND = 535, NONZEROSSTART = 536, NONZEROSSEND = 537, INDEXESSTART = 538, INDEXESEND = 539, VALUESSTART = 540, VALUESEND = 541, VARREFERENCEELEMENTSSTART = 542, VARREFERENCEELEMENTSEND = 543, LINEARELEMENTSSTART = 544, LINEARELEMENTSEND = 545, GENERALELEMENTSSTART = 546, GENERALELEMENTSEND = 547, CONREFERENCEELEMENTSSTART = 548, CONREFERENCEELEMENTSEND = 549, VALUETYPEATT = 550,
OBJREFERENCEELEMENTSSTART = 551, OBJREFERENCEELEMENTSEND = 552, PATTERNELEMENTSSTART = 553, PATTERNELEMENTSEND = 554,
VARIDXSTART = 555, VARIDXEND = 556, TRANSFORMATIONSTART = 557, TRANSFORMATIONEND = 558, COLOFFSETSSTART = 559, COLOFFSETSEND = 560, ROWOFFSETSSTART = 561, ROWOFFSETSEND = 562,
EMPTYROWMAJORATT = 563, ROWMAJORATT = 564, BLOCKROWIDXATT = 565, BLOCKCOLIDXATT = 566,
DUMMY = 567, NONLINEAREXPRESSIONSSTART = 568, NONLINEAREXPRESSIONSEND = 569, NUMBE-

ROFNONLINEAREXPRESSIONS = 570,
NLSTART = 571, NLEND = 572, MATRIXEXPRESSIONSSTART = 573, MATRIXEXPRESSIONSEND = 574,
NUMBEROFEXPR = 575, EXPRSTART = 576, EXPREND = 577, NUMBEROFMATRIXTERMSATT = 578,
MATRIXTERMSTART = 579, MATRIXTERMEND = 580, POWERSTART = 581, POWEREND = 582,
PLUSSTART = 583, PLUSEND = 584, MINUSSTART = 585, MINUSEND = 586,
DIVIDESTART = 587, DIVIDEEND = 588, LNSTART = 589, LLEND = 590,
SQRTSTART = 591, SQRTEND = 592, SUMSTART = 593, SUMEND = 594,
PRODUCTSTART = 595, PRODUCTEND = 596, EXPSTART = 597, EXPEND = 598,
NEGATESTART = 599, NEGATEEND = 600, IFSTART = 601, IFEND = 602,
SQUARESTART = 603, SQUAREEND = 604, COSSTART = 605, COSEND = 606,
SINSTART = 607, SINEND = 608, VARIABLESTART = 609, VARIABLEEND = 610,
ABSSTART = 611, ABSEND = 612, ERFSTART = 613, ERFEND = 614,
MAXSTART = 615, MAXEND = 616, ALLDIFFSTART = 617, ALLDIFFEND = 618,
MINSTART = 619, MINEND = 620, ESTART = 621, EEND = 622,
PISTART = 623, PIEND = 624, TIMESSTART = 625, TIMESEND = 626,
NUMBERSTART = 627, NUMBEREND = 628, MATRIXDETERMINANTSTART = 629, MATRIXDETERMINANT-
END = 630,
MATRIXTRACESTART = 631, MATRIXTRACEEND = 632, MATRIXTOSCALARSTART = 633, MATRIXTOSCA-
LAREND = 634,
MATRIXDIAGONALSTART = 635, MATRIXDIAGONALEND = 636, MATRIXDOTTIMESSTART = 637, MATRIX-
DOTTIMSEND = 638,
MATRIXLOWERTRIANGLESTART = 639, MATRIXLOWERTRIANGLEEND = 640, MATRIXUPPERTRIANGLE-
START = 641, MATRIXUPPERTRIANGLEEND = 642,
MATRIXMERGESTART = 643, MATRIXMERGEEND = 644, MATRIXMINUSSTART = 645, MATRIXMINUSEND
= 646,
MATRIXNEGATESTART = 647, MATRIXNEGATEEND = 648, MATRIXPLUSSTART = 649, MATRIXPLUSEND
= 650,
MATRIXTIMESSTART = 651, MATRIXTIMSEND = 652, MATRIXPRODUCTSTART = 653, MATRIXPRODUC-
TEND = 654,
MATRIXSCALARTIMESSTART = 655, MATRIXSCALARTIMSEND = 656, MATRIXSUBMATRIXATSTART =
657, MATRIXSUBMATRIXATEND = 658,
MATRIXTRANPOSESTART = 659, MATRIXTRANPOSEEND = 660, MATRIXREFERENCESTART = 661, M-
ATRIXREFERENCEEND = 662,
IDENTITYMATRIXSTART = 663, IDENTITYMATRIXEND = 664, MATRIXINVERSESTART = 665, MATRIXINV-
ERSEEND = 666,
EMPTYINCLUDEDIAGONALATT = 667, INCLUDEDIAGONALATT = 668, IDATT = 669, ATTRIBUTETEXT =
258,
ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
OSRLSTART = 267, OSRLSTARTEMPT = 268, OSRLATTRIBUTETEXT = 269, OSRLLEND = 270,
NUMBEROFCONATT = 271, NUMBEROFCONSTRAINTSATT = 272, NUMBEROFELATT = 273, NUMBEROF-
ENUMERATIONSATT = 274,
NUMBEROFIDXATT = 275, NUMBEROFITEMSATT = 276, NUMBEROFOBJATT = 277, NUMBEROFOBJECT-
IVESATT = 278,
NUMBEROFOTHERCONSTRAINTRESULTSATT = 279, NUMBEROFOTHEROBJECTIVERESULTSATT = 280,
NUMBEROFOTHERRESULTSATT = 281, NUMBEROFOTHERSOLUTIONRESULTSATT = 282,
NUMBEROFOTHERVARIABLERESULTSATT = 283, NUMBEROFSOLUTIONSATT = 284, NUMBEROFSOLV-
EROUTPUTSATT = 285, NUMBEROFSUBSTATUSESATT = 286,
NUMBEROFTIMESATT = 287, NUMBEROFVARATT = 288, NUMBEROFVARIABLESATT = 289, NUMBEROF-

VARIDXATT = 290,
 TARGETOBJECTIVEIDXATT = 291, IDXATT = 292, INCRATT = 293, MULTATT = 294,
 SIZEOFATT = 295, CATEGORYATT = 296, EMPTYCATEGORYATT = 297, DESCRIPTIONATT = 298,
 EMPTYDESCRIPTIONATT = 299, NAMEATT = 300, EMPTYNAMEATT = 301, TYPEATT = 302,
 EMPTYTYPEATT = 303, CONTYPEATT = 304, EMPTYCONTYPEATT = 305, ENUMTYPEATT = 306,
 EMPTYENUMTYPEATT = 307, OBJTYPEATT = 308, EMPTYOBJTYPEATT = 309, VARTYPEATT = 310,
 EMPTYVARTYPEATT = 311, UNITATT = 312, EMPTYUNITATT = 313, VALUEATT = 314,
 EMPTYVALUEATT = 315, WEIGHTEDOBJECTIVESATT = 316, EMPTYWEIGHTEDOBJECTIVESATT = 317,
 TARGETOBJECTIVENAMEATT = 318,
 EMPTYTARGETOBJECTIVENAMEATT = 319, HEADERSTART = 320, HEADEREND = 321, GENERALSTART
 = 322,
 GENERALEND = 323, SYSTEMSTART = 324, SYSTEMEND = 325, SERVICESTART = 326,
 SERVICEEND = 327, JOBSTART = 328, JOBEND = 329, OPTIMIZATIONSTART = 330,
 OPTIMIZATIONEND = 331, ITEMSTART = 332, ITEMEND = 333, ITEMSTARTANDEND = 334,
 ITEMEMPTY = 335, ACTUALSTARTTIMESTART = 336, ACTUALSTARTTIMEEND = 337, ATEQUALITYSTART
 = 338,
 ATEQUALITYEND = 339, ATLOWERSTART = 340, ATLOWEREND = 341, ATUPPERSTART = 342,
 ATUPPEREND = 343, AVAILABLECPUNUMBERSTART = 344, AVAILABLECPUNUMBEREND = 345, AVAILA-
 BLECPUSPEEDSTART = 346,
 AVAILABLECPUSPEEDEND = 347, AVAILABLEDISKSPACESTART = 348, AVAILABLEDISKSPACEEND =
 349, AVAILABLEMEMORYSTART = 350,
 AVAILABLEMEMORYEND = 351, BASE64START = 352, BASE64END = 353, BASICSTART = 354,
 BASICEND = 355, BASISSTATUSSTART = 356, BASISSTATUSEND = 357, BASSTATUSSTART = 358,
 BASSTATUSEND = 359, CONSTART = 360, CONEND = 361, CONSTRAINTSSTART = 362,
 CONSTRAINTSEND = 363, CURRENTJOBCOUNTSTART = 364, CURRENTJOBCOUNTEND = 365, CURRE-
 NTSTATESTART = 366,
 CURRENTSTATEEND = 367, DUALVALUESSTART = 368, DUALVALUESEND = 369, ELSTART = 370,
 EEND = 371, ENUMERATIONSTART = 372, ENUMERATIONEND = 373, ENDTIMESTART = 374,
 ENDTIMEEND = 375, GENERALSTATUSSTART = 376, GENERALSTATUSEND = 377, GENERALSUBSTAT-
 USSTART = 378,
 GENERALSUBSTATUSEND = 379, IDXSTART = 380, IDXEND = 381, INSTANCENAMESTART = 382,
 INSTANCENAMEEND = 383, ISFREESTART = 384, ISFREEEND = 385, JOBIDSTART = 386,
 JOBIDEND = 387, MESSAGESTART = 388, MESSAGEEND = 389, OBJSTART = 390,
 OBJEND = 391, OBJECTIVESSTART = 392, OBJECTIVESEND = 393, OPTIMIZATIONSOLUTIONSTATUSST-
 ART = 394,
 OPTIMIZATIONSOLUTIONSTATUSEND = 395, OPTIMIZATIONSOLUTIONSUBSTATUSSTART = 396, OPTI-
 MIZATIONSOLUTIONSUBSTATUSEND = 397, OTHERSTART = 398,
 OTHEREND = 399, OTHERRESULTSSTART = 400, OTHERRESULTSEND = 401, OTHERSOLUTIONRESUL-
 TSTART = 402,
 OTHERSOLUTIONRESULTEND = 403, OTHERSOLUTIONRESULTSSTART = 404, OTHERSOLUTIONRESUL-
 TSEND = 405, OTHERSOLVEROUTPUTSTART = 406,
 OTHERSOLVEROUTPUTEND = 407, SCHEDULEDSTARTTIMESTART = 408, SCHEDULEDSTARTTIMEEND
 = 409, SERVICENAMESTART = 410,
 SERVICENAMEEND = 411, SERVICEURISTART = 412, SERVICEURIEND = 413, SERVICEUTILIZATIONST-
 ART = 414,
 SERVICEUTILIZATIONEND = 415, SOLUTIONSTART = 416, SOLUTIONEND = 417, SOLVERINVOKEDSTA-
 RT = 418,
 SOLVERINVOKEDEND = 419, SOLVEROUTPUTSTART = 420, SOLVEROUTPUTEND = 421, STATUSSTART
 = 422,
 STATUSEND = 423, SUBMITTIMESTART = 424, SUBMITTIMEEND = 425, SUBSTATUSSTART = 426,
 SUBSTATUSEND = 427, SUPERBASICSTART = 428, SUPERBASICEND = 429, SYSTEMINFORMATIONST-
 ART = 430,
 SYSTEMINFORMATIONEND = 431, TIMESTART = 432, TIMEEND = 433, TIMESERVICESTARTEDSTART =

434,
TIMESERVICESTARTEDEND = 435, TIMESTAMPSTART = 436, TIMESTAMPEND = 437, TIMINGINFORMATIONSTART = 438,
TIMINGINFORMATIONEND = 439, TOTALJOBSSOFARSTART = 440, TOTALJOBSSOFAREND = 441, UNKNOWNSTART = 442,
UNKNOWNEND = 443, USEDCPUNUMBERSTART = 444, USEDCPUNUMBEREND = 445, USEDCPUSPEEDSTART = 446,
USEDCPUSPEEDEND = 447, USEDDISKSPACESTART = 448, USEDDISKSPACEEND = 449, USEDMEMORYSTART = 450,
USEDMEMORYEND = 451, VALUESSTRINGSTART = 452, VALUESSTRINGEND = 453, VARSTART = 454, VAREND = 455, VARIABLESSTART = 456, VARIABLESEND = 457, VARIDXSTART = 458,
VARIDXEND = 459, MATRIXVARSTART = 460, MATRIXVAREND = 461, MATRIXOBJSTART = 462, MATRIXOBJEND = 463, MATRIXCONSTART = 464, MATRIXCONEND = 465, FILENAMESTART = 466,
FILENAMEEND = 467, FILENAMEEMPTY = 468, FILENAMESTARTANDEND = 469, FILESOURCESTART = 470,
FILESOURCEEND = 471, FILESOURCEEMPTY = 472, FILESOURCESTARTANDEND = 473, FILEDESCRIPTIONSTART = 474,
FILEDESCRIPTIONEND = 475, FILEDESCRIPTIONEMPTY = 476, FILEDESCRIPTIONSTARTANDEND = 477, FILECREATORSTART = 478,
FILECREATOREND = 479, FILECREATOREMPTY = 480, FILECREATORSTARTANDEND = 481, FILELICENCESTART = 482,
FILELICENCEEND = 483, FILELICENCEEMPTY = 484, FILELICENCESTARTANDEND = 485, MATRIXSTART = 486,
MATRIXEND = 487, BASEMATRIXEND = 488, BASEMATRIXSTART = 489, BLOCKSTART = 490, BLOCKEND = 491, BLOCKSSTART = 492, BLOCKSEND = 493, EMPTYSHAPEATT = 494,
SHAPEATT = 495, EMPTYSYMMETRYATT = 496, SYMMETRYATT = 497, EMPTYNEGATIVEPATTERNATT = 498,
NEGATIVEPATTERNATT = 499, CONSTANTATT = 500, NUMBEROFBLOCKSATT = 501, NUMBEROFCOLUMNSATT = 502,
NUMBEROFFROWSATT = 503, NUMBEROFVALUESATT = 504, COEFATT = 505, BASEMATRIXIDXATT = 506, TARGETMATRIXFIRSTROWATT = 507, TARGETMATRIXFIRSTCOLATT = 508, BASEMATRIXSTARTROWATT = 509, BASEMATRIXSTARTCOLATT = 510,
BASEMATRIXENDROWATT = 511, BASEMATRIXENDCOLATT = 512, SCALARMULTIPLIERATT = 513, EMPTYBASETRANPOSEATT = 514,
BASETRANPOSEATT = 515, ELEMENTSSTART = 516, ELEMENTSEND = 517, CONSTATELEMENTSSTART = 518,
CONSTATELEMENTSEND = 519, STARTVECTORSTART = 520, STARTVECTOREND = 521, NONZEROSTART = 522,
NONZEROSSEND = 523, INDEXESSTART = 524, INDEXESEND = 525, VALUESSTART = 526, VALUESEND = 527, VARREFERENCEELEMENTSSTART = 528, VARREFERENCEELEMENTSEND = 529, LINEARELEMENTSSTART = 530,
LINEARELEMENTSEND = 531, GENERALELEMENTSSTART = 532, GENERALELEMENTSEND = 533, CONREFERENCEELEMENTSSTART = 534,
CONREFERENCEELEMENTSEND = 535, VALUETYPEATT = 536, OBJREFERENCEELEMENTSSTART = 537, OBJREFERENCEELEMENTSEND = 538,
PATTERNELEMENTSSTART = 539, PATTERNELEMENTSEND = 540, TRANSFORMATIONSTART = 541, TRANSFORMATIONEND = 542,
COLOFFSETSSTART = 543, COLOFFSETSEND = 544, ROWOFFSETSSTART = 545, ROWOFFSETSEND = 546,
EMPTYROWMAJORATT = 547, ROWMAJORATT = 548, BLOCKROWIDXATT = 549, BLOCKCOLIDXATT = 550,
DUMMY = 551, NONLINEAREXPRESSIONSSTART = 552, NONLINEAREXPRESSIONSEND = 553, NUMBE-

```

ROFNONLINEAREXPRESSIONS = 554,
NLSTART = 555, NLEND = 556, MATRIXEXPRESSIONSSTART = 557, MATRIXEXPRESSIONSEND = 558,
NUMBEROFEXPR = 559, EXPRSTART = 560, EXPREND = 561, NUMBEROFMATRIXTERMSATT = 562,
MATRIXTERMSTART = 563, MATRIXTERMEND = 564, POWERSTART = 565, POWEREND = 566,
PLUSSTART = 567, PLUSEND = 568, MINUSSTART = 569, MINUSEND = 570,
DIVIDESTART = 571, DIVIDEEND = 572, LNSTART = 573, LNEND = 574,
SQRTSTART = 575, SQRTEND = 576, SUMSTART = 577, SUMEND = 578,
PRODUCTSTART = 579, PRODUCTEND = 580, EXPSTART = 581, EXPEND = 582,
NEGATESTART = 583, NEGATEEND = 584, IFSTART = 585, IFEND = 586,
SQUARESTART = 587, SQUAREEND = 588, COSSTART = 589, COSEND = 590,
SINSTART = 591, SINEND = 592, VARIABLESTART = 593, VARIABLEEND = 594,
ABSSTART = 595, ABSEND = 596, ERFSTART = 597, ERFEND = 598,
MAXSTART = 599, MAXEND = 600, ALLDIFFSTART = 601, ALLDIFFEND = 602,
MINSTART = 603, MINEND = 604, ESTART = 605, EEND = 606,
PISTART = 607, PIEND = 608, TIMESSTART = 609, TIMESEND = 610,
NUMBERSTART = 611, NUMBEREND = 612, MATRIXDETERMINANTSTART = 613, MATRIXDETERMINANT-
END = 614,
MATRIXTRACESTART = 615, MATRIXTRACEEND = 616, MATRIXTOSCALARSTART = 617, MATRIXTOSCA-
LAREND = 618,
MATRIXDIAGONALSTART = 619, MATRIXDIAGONALEND = 620, MATRIXDOTTIMESSTART = 621, MATRIX-
DOTTIMESEND = 622,
MATRIXLOWERTRIANGLESTART = 623, MATRIXLOWERTRIANGLEEND = 624, MATRIXUPPERTRIANGLE-
START = 625, MATRIXUPPERTRIANGLEEND = 626,
MATRIXMERGESTART = 627, MATRIXMERGEEND = 628, MATRIXMINUSSTART = 629, MATRIXMINUSEND
= 630,
MATRIXNEGATESTART = 631, MATRIXNEGATEEND = 632, MATRIXPLUSSTART = 633, MATRIXPLUSEND
= 634,
MATRIXTIMESSTART = 635, MATRIXTIMESEND = 636, MATRIXPRODUCTSTART = 637, MATRIXPRODUC-
TEND = 638,
MATRIXSCALARTIMESSTART = 639, MATRIXSCALARTIMESEND = 640, MATRIXSUBMATRIXATSTART =
641, MATRIXSUBMATRIXATEND = 642,
MATRIXTRANPOSESTART = 643, MATRIXTRANPOSEEND = 644, MATRIXREFERENCESTART = 645, M-
ATRIXREFERENCEEND = 646,
IDENTITYMATRIXSTART = 647, IDENTITYMATRIXEND = 648, MATRIXINVERSESTART = 649, MATRIXINV-
ERSEEND = 650,
EMPTYINCLUDEDIAGONALATT = 651, INCLUDEDIAGONALATT = 652, IDATT = 653 }

```

7.36.1 Macro Definition Documentation

7.36.1.1 #define ATTRIBUTETEXT 258

Definition at line 457 of file OSParseosol.tab.hpp.

7.36.1.2 #define ELEMENTTEXT 259

Definition at line 458 of file OSParseosol.tab.hpp.

7.36.1.3 #define ITEMTEXT 260

Definition at line 459 of file OSParseosol.tab.hpp.

7.36.1.4 #define INTEGER 261

Definition at line 460 of file OSParseosol.tab.hpp.

7.36.1.5 #define DOUBLE 262

Definition at line 461 of file OSParseosol.tab.hpp.

7.36.1.6 #define QUOTE 263

Definition at line 462 of file OSParseosol.tab.hpp.

7.36.1.7 #define TWOQUOTES 264

Definition at line 463 of file OSParseosol.tab.hpp.

7.36.1.8 #define GREATERTHAN 265

Definition at line 464 of file OSParseosol.tab.hpp.

7.36.1.9 #define ENDOFELEMENT 266

Definition at line 465 of file OSParseosol.tab.hpp.

7.36.1.10 #define OSOLSTART 267

Definition at line 466 of file OSParseosol.tab.hpp.

7.36.1.11 #define OSOLSTARTEMPT 268

Definition at line 467 of file OSParseosol.tab.hpp.

7.36.1.12 #define OSOLATTRIBUTETEXT 269

Definition at line 468 of file OSParseosol.tab.hpp.

7.36.1.13 #define OSOLEND 270

Definition at line 469 of file OSParseosol.tab.hpp.

7.36.1.14 #define NUMBEROFOTHEROPTIONSATT 271

Definition at line 470 of file OSParseosol.tab.hpp.

7.36.1.15 #define NUMBEROFENUMERATIONSATT 272

Definition at line 471 of file OSParseosol.tab.hpp.

7.36.1.16 #define NUMBEROFJOBIDSATT 273

Definition at line 472 of file OSParseosol.tab.hpp.

7.36.1.17 #define NUMBEROFPATHSATT 274

Definition at line 473 of file OSParseosol.tab.hpp.

7.36.1.18 #define NUMBEROFPATHPAIRSATT 275

Definition at line 474 of file OSParseosol.tab.hpp.

7.36.1.19 #define FROMATT 276

Definition at line 475 of file OSParseosol.tab.hpp.

7.36.1.20 #define TOATT 277

Definition at line 476 of file OSParseosol.tab.hpp.

7.36.1.21 #define MAKECOPYATT 278

Definition at line 477 of file OSParseosol.tab.hpp.

7.36.1.22 #define CATEGORYATT 279

Definition at line 478 of file OSParseosol.tab.hpp.

7.36.1.23 #define TYPEATT 280

Definition at line 479 of file OSParseosol.tab.hpp.

7.36.1.24 #define GROUPWEIGHTATT 281

Definition at line 480 of file OSParseosol.tab.hpp.

7.36.1.25 #define NUMBEROFPROCESSESATT 282

Definition at line 481 of file OSParseosol.tab.hpp.

7.36.1.26 #define NUMBEROF SOLVEROPTIONSATT 283

Definition at line 482 of file OSParseosol.tab.hpp.

7.36.1.27 #define NUMBEROF SOSATT 284

Definition at line 483 of file OSParseosol.tab.hpp.

7.36.1.28 #define NUMBEROF VARIABLESATT 285

Definition at line 484 of file OSParseosol.tab.hpp.

7.36.1.29 #define NUMBEROF OBJECTIVESATT 286

Definition at line 485 of file OSParseosol.tab.hpp.

7.36.1.30 #define NUMBEROF CONSTRAINTSATT 287

Definition at line 486 of file OSParseosol.tab.hpp.

7.36.1.31 #define NUMBEROF OTHER VARIABLE OPTIONSATT 288

Definition at line 487 of file OSParseosol.tab.hpp.

7.36.1.32 #define NUMBEROF OTHER OBJECTIVE OPTIONSATT 289

Definition at line 488 of file OSParseosol.tab.hpp.

7.36.1.33 #define NUMBEROFOTHERCONSTRAINTOPTIONSATT 290

Definition at line 489 of file OSParseosol.tab.hpp.

7.36.1.34 #define NUMBEROFITEMSATT 291

Definition at line 490 of file OSParseosol.tab.hpp.

7.36.1.35 #define NUMBEROFVARATT 292

Definition at line 491 of file OSParseosol.tab.hpp.

7.36.1.36 #define NUMBEROFOBJATT 293

Definition at line 492 of file OSParseosol.tab.hpp.

7.36.1.37 #define NUMBEROFCONATT 294

Definition at line 493 of file OSParseosol.tab.hpp.

7.36.1.38 #define NUMBEROFELATT 295

Definition at line 494 of file OSParseosol.tab.hpp.

7.36.1.39 #define NAMEATT 296

Definition at line 495 of file OSParseosol.tab.hpp.

7.36.1.40 #define IDXATT 297

Definition at line 496 of file OSParseosol.tab.hpp.

7.36.1.41 #define SOSIDXATT 298

Definition at line 497 of file OSParseosol.tab.hpp.

7.36.1.42 #define VALUEATT 299

Definition at line 498 of file OSParseosol.tab.hpp.

7.36.1.43 #define UNITATT 300

Definition at line 499 of file OSParseosol.tab.hpp.

7.36.1.44 #define DESCRIPTIONATT 301

Definition at line 500 of file OSParseosol.tab.hpp.

7.36.1.45 #define CONTYPEATT 302

Definition at line 501 of file OSParseosol.tab.hpp.

7.36.1.46 #define EMPTYCONTYPEATT 303

Definition at line 502 of file OSParseosol.tab.hpp.

7.36.1.47 #define ENUMTYPEATT 304

Definition at line 503 of file OSParseosol.tab.hpp.

7.36.1.48 #define EMPTYENUMTYPEATT 305

Definition at line 504 of file OSParseosol.tab.hpp.

7.36.1.49 #define OBJTYPEATT 306

Definition at line 505 of file OSParseosol.tab.hpp.

7.36.1.50 #define EMPTYOBJTYPEATT 307

Definition at line 506 of file OSParseosol.tab.hpp.

7.36.1.51 #define VARTYPEATT 308

Definition at line 507 of file OSParseosol.tab.hpp.

7.36.1.52 #define EMPTYVARTYPEATT 309

Definition at line 508 of file OSParseosol.tab.hpp.

7.36.1.53 #define EMPTYTYPEATT 310

Definition at line 509 of file OSParseosol.tab.hpp.

7.36.1.54 #define EMPTYNAMEATT 311

Definition at line 510 of file OSParseosol.tab.hpp.

7.36.1.55 #define EMPTYCATEGORYATT 312

Definition at line 511 of file OSParseosol.tab.hpp.

7.36.1.56 #define EMPTYDESCRIPTIONATT 313

Definition at line 512 of file OSParseosol.tab.hpp.

7.36.1.57 #define EMPTYUNITATT 314

Definition at line 513 of file OSParseosol.tab.hpp.

7.36.1.58 #define EMPTYVALUEATT 315

Definition at line 514 of file OSParseosol.tab.hpp.

7.36.1.59 #define EMPTYLBVALUEATT 316

Definition at line 515 of file OSParseosol.tab.hpp.

7.36.1.60 #define EMPTYUBVALUEATT 317

Definition at line 516 of file OSParseosol.tab.hpp.

7.36.1.61 #define LBVALUEATT 318

Definition at line 517 of file OSParseosol.tab.hpp.

7.36.1.62 #define UBVALUEATT 319

Definition at line 518 of file OSParseosol.tab.hpp.

7.36.1.63 #define EMPTYLBDUALVALUEATT 320

Definition at line 519 of file OSParseosol.tab.hpp.

7.36.1.64 #define EMPTYUBDUALVALUEATT 321

Definition at line 520 of file OSParseosol.tab.hpp.

7.36.1.65 #define LBDUALVALUEATT 322

Definition at line 521 of file OSParseosol.tab.hpp.

7.36.1.66 #define UBDUALVALUEATT 323

Definition at line 522 of file OSParseosol.tab.hpp.

7.36.1.67 #define SOLVERATT 324

Definition at line 523 of file OSParseosol.tab.hpp.

7.36.1.68 #define EMPTYSOLVERATT 325

Definition at line 524 of file OSParseosol.tab.hpp.

7.36.1.69 #define WEIGHTATT 326

Definition at line 525 of file OSParseosol.tab.hpp.

7.36.1.70 #define EMPTYWEIGHTATT 327

Definition at line 526 of file OSParseosol.tab.hpp.

7.36.1.71 #define TRANSPORTTYPEATT 328

Definition at line 527 of file OSParseosol.tab.hpp.

7.36.1.72 #define LOCATIONTYPEATT 329

Definition at line 528 of file OSParseosol.tab.hpp.

7.36.1.73 #define GENERALSTART 330

Definition at line 529 of file OSParseosol.tab.hpp.

7.36.1.74 #define GENERALEND 331

Definition at line 530 of file OSParseosol.tab.hpp.

7.36.1.75 #define SYSTEMSTART 332

Definition at line 531 of file OSParseosol.tab.hpp.

7.36.1.76 #define SYSTEMEND 333

Definition at line 532 of file OSParseosol.tab.hpp.

7.36.1.77 #define SERVICESTART 334

Definition at line 533 of file OSParseosol.tab.hpp.

7.36.1.78 #define SERVICEEND 335

Definition at line 534 of file OSParseosol.tab.hpp.

7.36.1.79 #define JOBSTART 336

Definition at line 535 of file OSParseosol.tab.hpp.

7.36.1.80 #define JOBEND 337

Definition at line 536 of file OSParseosol.tab.hpp.

7.36.1.81 #define OPTIMIZATIONSTART 338

Definition at line 537 of file OSParseosol.tab.hpp.

7.36.1.82 #define OPTIMIZATIONEND 339

Definition at line 538 of file OSParseosol.tab.hpp.

7.36.1.83 #define SERVICEURISTART 340

Definition at line 539 of file OSParseosol.tab.hpp.

7.36.1.84 #define SERVICEURIEND 341

Definition at line 540 of file OSParseosol.tab.hpp.

7.36.1.85 #define SERVICENAMESTART 342

Definition at line 541 of file OSParseosol.tab.hpp.

7.36.1.86 #define SERVICENAMEEND 343

Definition at line 542 of file OSParseosol.tab.hpp.

7.36.1.87 #define INSTANCENAMESTART 344

Definition at line 543 of file OSParseosol.tab.hpp.

7.36.1.88 #define INSTANCENAMEEND 345

Definition at line 544 of file OSParseosol.tab.hpp.

7.36.1.89 #define INSTANCELOCATIONSTART 346

Definition at line 545 of file OSParseosol.tab.hpp.

7.36.1.90 #define INSTANCELOCATIONEND 347

Definition at line 546 of file OSParseosol.tab.hpp.

7.36.1.91 #define JOBIDSTART 348

Definition at line 547 of file OSParseosol.tab.hpp.

7.36.1.92 #define JOBIDEND 349

Definition at line 548 of file OSParseosol.tab.hpp.

7.36.1.93 #define SOLVERTOINVOKESTART 350

Definition at line 549 of file OSParseosol.tab.hpp.

7.36.1.94 #define SOLVERTOINVOKEEND 351

Definition at line 550 of file OSParseosol.tab.hpp.

7.36.1.95 #define LICENSESTART 352

Definition at line 551 of file OSParseosol.tab.hpp.

7.36.1.96 #define LICENSEEND 353

Definition at line 552 of file OSParseosol.tab.hpp.

7.36.1.97 #define USERNAMESTART 354

Definition at line 553 of file OSParseosol.tab.hpp.

7.36.1.98 #define USERNAMEEND 355

Definition at line 554 of file OSParseosol.tab.hpp.

7.36.1.99 #define PASSWORDSTART 356

Definition at line 555 of file OSParseosol.tab.hpp.

7.36.1.100 #define PASSWORDEND 357

Definition at line 556 of file OSParseosol.tab.hpp.

7.36.1.101 #define CONTACTSTART 358

Definition at line 557 of file OSParseosol.tab.hpp.

7.36.1.102 #define CONTACTEND 359

Definition at line 558 of file OSParseosol.tab.hpp.

7.36.1.103 **#define OTHEROPTIONSSTART 360**

Definition at line 559 of file OSParseosol.tab.hpp.

7.36.1.104 **#define OTHEROPTIONSEND 361**

Definition at line 560 of file OSParseosol.tab.hpp.

7.36.1.105 **#define OTHERSTART 362**

Definition at line 561 of file OSParseosol.tab.hpp.

7.36.1.106 **#define OTHEREND 363**

Definition at line 562 of file OSParseosol.tab.hpp.

7.36.1.107 **#define MINDISKSPACESTART 364**

Definition at line 563 of file OSParseosol.tab.hpp.

7.36.1.108 **#define MINDISKSPACEEND 365**

Definition at line 564 of file OSParseosol.tab.hpp.

7.36.1.109 **#define MINMEMORYSTART 366**

Definition at line 565 of file OSParseosol.tab.hpp.

7.36.1.110 **#define MINMEMORYEND 367**

Definition at line 566 of file OSParseosol.tab.hpp.

7.36.1.111 **#define MINCPUSPEEDSTART 368**

Definition at line 567 of file OSParseosol.tab.hpp.

7.36.1.112 **#define MINCPUSPEEDEND 369**

Definition at line 568 of file OSParseosol.tab.hpp.

7.36.1.113 **#define MINCPUNUMBERSTART 370**

Definition at line 569 of file OSParseosol.tab.hpp.

7.36.1.114 **#define MINCPUNUMBEREND 371**

Definition at line 570 of file OSParseosol.tab.hpp.

7.36.1.115 **#define SERVICETYPESTART 372**

Definition at line 571 of file OSParseosol.tab.hpp.

7.36.1.116 **#define SERVICETYPEEND 373**

Definition at line 572 of file OSParseosol.tab.hpp.

7.36.1.117 **#define MAXTimestart 374**

Definition at line 573 of file OSParseosol.tab.hpp.

7.36.1.118 **#define MAXTIMEEND 375**

Definition at line 574 of file OSParseosol.tab.hpp.

7.36.1.119 **#define REQUESTEDSTARTTimestart 376**

Definition at line 575 of file OSParseosol.tab.hpp.

7.36.1.120 **#define REQUESTEDSTARTTIMEEND 377**

Definition at line 576 of file OSParseosol.tab.hpp.

7.36.1.121 **#define DEPENDENCIESSTART 378**

Definition at line 577 of file OSParseosol.tab.hpp.

7.36.1.122 **#define DEPENDENCIESEND 379**

Definition at line 578 of file OSParseosol.tab.hpp.

7.36.1.123 **#define REQUIREDDIRECTORIESSTART 380**

Definition at line 579 of file OSParseosol.tab.hpp.

7.36.1.124 **#define REQUIREDDIRECTORIESEND 381**

Definition at line 580 of file OSParseosol.tab.hpp.

7.36.1.125 **#define REQUIREDFILESSTART 382**

Definition at line 581 of file OSParseosol.tab.hpp.

7.36.1.126 **#define REQUIREDFILESEND 383**

Definition at line 582 of file OSParseosol.tab.hpp.

7.36.1.127 **#define PATHSTART 384**

Definition at line 583 of file OSParseosol.tab.hpp.

7.36.1.128 **#define PATHEND 385**

Definition at line 584 of file OSParseosol.tab.hpp.

7.36.1.129 **#define PATHPAIRSTART 386**

Definition at line 585 of file OSParseosol.tab.hpp.

7.36.1.130 **#define PATHPAIREND 387**

Definition at line 586 of file OSParseosol.tab.hpp.

7.36.1.131 **#define DIRECTORIESTOMAKESTART 388**

Definition at line 587 of file OSParseosol.tab.hpp.

7.36.1.132 **#define DIRECTORIESTOMAKEEND 389**

Definition at line 588 of file OSParseosol.tab.hpp.

7.36.1.133 **#define FILESTOMAKESTART 390**

Definition at line 589 of file OSParseosol.tab.hpp.

7.36.1.134 **#define FILESTOMAKEEND 391**

Definition at line 590 of file OSParseosol.tab.hpp.

7.36.1.135 **#define DIRECTORIESTODELETESTART 392**

Definition at line 591 of file OSParseosol.tab.hpp.

7.36.1.136 **#define DIRECTORIESTODELETEEND 393**

Definition at line 592 of file OSParseosol.tab.hpp.

7.36.1.137 **#define FILESTODELETESTART 394**

Definition at line 593 of file OSParseosol.tab.hpp.

7.36.1.138 **#define FILESTODELETEEND 395**

Definition at line 594 of file OSParseosol.tab.hpp.

7.36.1.139 **#define INPUTDIRECTORIESTOMOVESTART 396**

Definition at line 595 of file OSParseosol.tab.hpp.

7.36.1.140 **#define INPUTDIRECTORIESTOMOVEEND 397**

Definition at line 596 of file OSParseosol.tab.hpp.

7.36.1.141 **#define INPUTFILESTOMOVESTART 398**

Definition at line 597 of file OSParseosol.tab.hpp.

7.36.1.142 **#define INPUTFILESTOMOVEEND 399**

Definition at line 598 of file OSParseosol.tab.hpp.

7.36.1.143 **#define OUTPUTDIRECTORIESTOMOVESTART 400**

Definition at line 599 of file OSParseosol.tab.hpp.

7.36.1.144 **#define OUTPUTDIRECTORIESTOMOVEEND 401**

Definition at line 600 of file OSParseosol.tab.hpp.

7.36.1.145 **#define OUTPUTFILESTOMOVESTART 402**

Definition at line 601 of file OSParseosol.tab.hpp.

7.36.1.146 **#define OUTPUTFILESTOMOVEEND 403**

Definition at line 602 of file OSParseosol.tab.hpp.

7.36.1.147 **#define PROCESSESTOKILLSTART 404**

Definition at line 603 of file OSParseosol.tab.hpp.

7.36.1.148 **#define PROCESSESTOKILLEND 405**

Definition at line 604 of file OSParseosol.tab.hpp.

7.36.1.149 **#define PROCESSSTART 406**

Definition at line 605 of file OSParseosol.tab.hpp.

7.36.1.150 **#define PROCESSEND 407**

Definition at line 606 of file OSParseosol.tab.hpp.

7.36.1.151 **#define VARIABLESSTART 408**

Definition at line 607 of file OSParseosol.tab.hpp.

7.36.1.152 **#define VARIABLESEND 409**

Definition at line 608 of file OSParseosol.tab.hpp.

7.36.1.153 **#define INITIALVARIABLEVALUESSTART 410**

Definition at line 609 of file OSParseosol.tab.hpp.

7.36.1.154 **#define INITIALVARIABLEVALUESEND 411**

Definition at line 610 of file OSParseosol.tab.hpp.

7.36.1.155 **#define VARSTART 412**

Definition at line 611 of file OSParseosol.tab.hpp.

7.36.1.156 **#define VAREND 413**

Definition at line 612 of file OSParseosol.tab.hpp.

7.36.1.157 **#define INITIALVARIABLEVALUESSTRINGSTART 414**

Definition at line 613 of file OSParseosol.tab.hpp.

7.36.1.158 **#define INITIALVARIABLEVALUESSTRINGEND 415**

Definition at line 614 of file OSParseosol.tab.hpp.

7.36.1.159 **#define INITIALBASISSTATUSSTART 416**

Definition at line 615 of file OSParseosol.tab.hpp.

7.36.1.160 **#define INITIALBASISSTATUSEND 417**

Definition at line 616 of file OSParseosol.tab.hpp.

7.36.1.161 **#define BASICSTART 418**

Definition at line 617 of file OSParseosol.tab.hpp.

7.36.1.162 **#define BASICEND 419**

Definition at line 618 of file OSParseosol.tab.hpp.

7.36.1.163 **#define ATUPPERSTART 420**

Definition at line 619 of file OSParseosol.tab.hpp.

7.36.1.164 **#define ATUPPEREND 421**

Definition at line 620 of file OSParseosol.tab.hpp.

7.36.1.165 **#define ATLOWERSTART 422**

Definition at line 621 of file OSParseosol.tab.hpp.

7.36.1.166 **#define ATLOWEREND 423**

Definition at line 622 of file OSParseosol.tab.hpp.

7.36.1.167 **#define ATEQUALITYSTART 424**

Definition at line 623 of file OSParseosol.tab.hpp.

7.36.1.168 **#define ATEQUALITYEND 425**

Definition at line 624 of file OSParseosol.tab.hpp.

7.36.1.169 **#define SUPERBASICSTART 426**

Definition at line 625 of file OSParseosol.tab.hpp.

7.36.1.170 **#define SUPERBASICEND 427**

Definition at line 626 of file OSParseosol.tab.hpp.

7.36.1.171 **#define ISFREESTART 428**

Definition at line 627 of file OSParseosol.tab.hpp.

7.36.1.172 **#define ISFREEEND 429**

Definition at line 628 of file OSParseosol.tab.hpp.

7.36.1.173 **#define UNKNOWNSTART 430**

Definition at line 629 of file OSParseosol.tab.hpp.

7.36.1.174 **#define UNKNOWNEND 431**

Definition at line 630 of file OSParseosol.tab.hpp.

7.36.1.175 **#define INTEGERVARIABLEBRANCHINGWEIGHTSSTART 432**

Definition at line 631 of file OSParseosol.tab.hpp.

7.36.1.176 **#define INTEGERVARIABLEBRANCHINGWEIGHTSEND 433**

Definition at line 632 of file OSParseosol.tab.hpp.

7.36.1.177 **#define SOSVARIABLEBRANCHINGWEIGHTSSTART 434**

Definition at line 633 of file OSParseosol.tab.hpp.

7.36.1.178 **#define SOSVARIABLEBRANCHINGWEIGHTSEND 435**

Definition at line 634 of file OSParseosol.tab.hpp.

7.36.1.179 **#define SOSSTART 436**

Definition at line 635 of file OSParseosol.tab.hpp.

7.36.1.180 **#define SOSEND 437**

Definition at line 636 of file OSParseosol.tab.hpp.

7.36.1.181 **#define OBJECTIVESSTART 438**

Definition at line 637 of file OSParseosol.tab.hpp.

7.36.1.182 **#define OBJECTIVESEND 439**

Definition at line 638 of file OSParseosol.tab.hpp.

7.36.1.183 **#define INITIALOBJECTIVEVALUESSTART 440**

Definition at line 639 of file OSParseosol.tab.hpp.

7.36.1.184 **#define INITIALOBJECTIVEVALUESEND 441**

Definition at line 640 of file OSParseosol.tab.hpp.

7.36.1.185 **#define OBJSTART 442**

Definition at line 641 of file OSParseosol.tab.hpp.

7.36.1.186 **#define OBJEND 443**

Definition at line 642 of file OSParseosol.tab.hpp.

7.36.1.187 **#define INITIALOBJECTIVEBOUNDSSTART 444**

Definition at line 643 of file OSParseosol.tab.hpp.

7.36.1.188 **#define INITIALOBJECTIVEBOUNSEND 445**

Definition at line 644 of file OSParseosol.tab.hpp.

7.36.1.189 **#define CONSTRAINTSSTART 446**

Definition at line 645 of file OSParseosol.tab.hpp.

7.36.1.190 **#define CONSTRAINTSEND 447**

Definition at line 646 of file OSParseosol.tab.hpp.

7.36.1.191 **#define INITIALCONSTRAINTVALUESSTART 448**

Definition at line 647 of file OSParseosol.tab.hpp.

7.36.1.192 **#define INITIALCONSTRAINTVALUESEND 449**

Definition at line 648 of file OSParseosol.tab.hpp.

7.36.1.193 **#define CONSTART 450**

Definition at line 649 of file OSParseosol.tab.hpp.

7.36.1.194 **#define CONEND 451**

Definition at line 650 of file OSParseosol.tab.hpp.

7.36.1.195 **#define INITIALDUALVALUESSTART 452**

Definition at line 651 of file OSParseosol.tab.hpp.

7.36.1.196 **#define INITIALDUALVALUESEND 453**

Definition at line 652 of file OSParseosol.tab.hpp.

7.36.1.197 **#define SOLVEROPTIONSSTART 454**

Definition at line 653 of file OSParseosol.tab.hpp.

7.36.1.198 **#define SOLVEROPTIONSEND 455**

Definition at line 654 of file OSParseosol.tab.hpp.

7.36.1.199 **#define SOLVEROPTIONSTART 456**

Definition at line 655 of file OSParseosol.tab.hpp.

7.36.1.200 **#define SOLVEROPTIONEND 457**

Definition at line 656 of file OSParseosol.tab.hpp.

7.36.1.201 #define ENUMERATIONSTART 458

Definition at line 657 of file OSParseosol.tab.hpp.

7.36.1.202 #define ENUMERATIONEND 459

Definition at line 658 of file OSParseosol.tab.hpp.

7.36.1.203 #define ITEMEMPTY 460

Definition at line 659 of file OSParseosol.tab.hpp.

7.36.1.204 #define ITEMSTART 461

Definition at line 660 of file OSParseosol.tab.hpp.

7.36.1.205 #define ITEMEND 462

Definition at line 661 of file OSParseosol.tab.hpp.

7.36.1.206 #define ITEMSTARTANDEND 463

Definition at line 662 of file OSParseosol.tab.hpp.

7.36.1.207 #define BASE64START 464

Definition at line 663 of file OSParseosol.tab.hpp.

7.36.1.208 #define BASE64END 465

Definition at line 664 of file OSParseosol.tab.hpp.

7.36.1.209 #define INCRATT 466

Definition at line 665 of file OSParseosol.tab.hpp.

7.36.1.210 #define MULTATT 467

Definition at line 666 of file OSParseosol.tab.hpp.

7.36.1.211 #define SIZEOFATT 468

Definition at line 667 of file OSParseosol.tab.hpp.

7.36.1.212 #define ELSTART 469

Definition at line 668 of file OSParseosol.tab.hpp.

7.36.1.213 #define ELEND 470

Definition at line 669 of file OSParseosol.tab.hpp.

7.36.1.214 #define MATRIXVARSTART 471

Definition at line 670 of file OSParseosol.tab.hpp.

7.36.1.215 #define MATRIXVAREND 472

Definition at line 671 of file OSParseosol.tab.hpp.

7.36.1.216 #define MATRIXOBJSTART 473

Definition at line 672 of file OSParseosol.tab.hpp.

7.36.1.217 #define MATRIXOBJEND 474

Definition at line 673 of file OSParseosol.tab.hpp.

7.36.1.218 #define MATRIXCONSTART 475

Definition at line 674 of file OSParseosol.tab.hpp.

7.36.1.219 #define MATRIXCONEND 476

Definition at line 675 of file OSParseosol.tab.hpp.

7.36.1.220 #define HEADERSTART 477

Definition at line 676 of file OSParseosol.tab.hpp.

7.36.1.221 #define HEADEREND 478

Definition at line 677 of file OSParseosol.tab.hpp.

7.36.1.222 #define FILENAMESTART 479

Definition at line 678 of file OSParseosol.tab.hpp.

7.36.1.223 #define FILENAMEEND 480

Definition at line 679 of file OSParseosol.tab.hpp.

7.36.1.224 #define FILENAMEEMPTY 481

Definition at line 680 of file OSParseosol.tab.hpp.

7.36.1.225 #define FILENAMESTARTANDEND 482

Definition at line 681 of file OSParseosol.tab.hpp.

7.36.1.226 #define FILESOURCSTART 483

Definition at line 682 of file OSParseosol.tab.hpp.

7.36.1.227 #define FILESOURCEEND 484

Definition at line 683 of file OSParseosol.tab.hpp.

7.36.1.228 #define FILESOURCEEMPTY 485

Definition at line 684 of file OSParseosol.tab.hpp.

7.36.1.229 **#define** FILESOURCESTARTANDEND 486

Definition at line 685 of file OSParseosol.tab.hpp.

7.36.1.230 **#define** FILEDESCRIPTIONSTART 487

Definition at line 686 of file OSParseosol.tab.hpp.

7.36.1.231 **#define** FILEDESCRIPTIONEND 488

Definition at line 687 of file OSParseosol.tab.hpp.

7.36.1.232 **#define** FILEDESCRIPTIONEMPTY 489

Definition at line 688 of file OSParseosol.tab.hpp.

7.36.1.233 **#define** FILEDESCRIPTIONSTARTANDEND 490

Definition at line 689 of file OSParseosol.tab.hpp.

7.36.1.234 **#define** FILECREATORSTART 491

Definition at line 690 of file OSParseosol.tab.hpp.

7.36.1.235 **#define** FILECREATOREND 492

Definition at line 691 of file OSParseosol.tab.hpp.

7.36.1.236 **#define** FILECREATOREMPTY 493

Definition at line 692 of file OSParseosol.tab.hpp.

7.36.1.237 **#define** FILECREATORSTARTANDEND 494

Definition at line 693 of file OSParseosol.tab.hpp.

7.36.1.238 **#define** FILELICENCESTART 495

Definition at line 694 of file OSParseosol.tab.hpp.

7.36.1.239 **#define** FILELICENCEEND 496

Definition at line 695 of file OSParseosol.tab.hpp.

7.36.1.240 **#define** FILELICENCEEMPTY 497

Definition at line 696 of file OSParseosol.tab.hpp.

7.36.1.241 **#define** FILELICENCESTARTANDEND 498

Definition at line 697 of file OSParseosol.tab.hpp.

7.36.1.242 **#define** MATRIXSTART 499

Definition at line 698 of file OSParseosol.tab.hpp.

7.36.1.243 #define MATRIXEND 500

Definition at line 699 of file OSParseosol.tab.hpp.

7.36.1.244 #define BASEMATRIXEND 501

Definition at line 700 of file OSParseosol.tab.hpp.

7.36.1.245 #define BASEMATRIXSTART 502

Definition at line 701 of file OSParseosol.tab.hpp.

7.36.1.246 #define BLOCKSTART 503

Definition at line 702 of file OSParseosol.tab.hpp.

7.36.1.247 #define BLOCKEND 504

Definition at line 703 of file OSParseosol.tab.hpp.

7.36.1.248 #define BLOCKSSTART 505

Definition at line 704 of file OSParseosol.tab.hpp.

7.36.1.249 #define BLOCKSEND 506

Definition at line 705 of file OSParseosol.tab.hpp.

7.36.1.250 #define EMPTYSHAPEATT 507

Definition at line 706 of file OSParseosol.tab.hpp.

7.36.1.251 #define SHAPEATT 508

Definition at line 707 of file OSParseosol.tab.hpp.

7.36.1.252 #define EMPTYSYMMETRYATT 509

Definition at line 708 of file OSParseosol.tab.hpp.

7.36.1.253 #define SYMMETRYATT 510

Definition at line 709 of file OSParseosol.tab.hpp.

7.36.1.254 #define EMPTYNEGATIVEPATTERNATT 511

Definition at line 710 of file OSParseosol.tab.hpp.

7.36.1.255 #define NEGATIVEPATTERNATT 512

Definition at line 711 of file OSParseosol.tab.hpp.

7.36.1.256 #define CONSTANTATT 513

Definition at line 712 of file OSParseosol.tab.hpp.

7.36.1.257 **#define** NUMBEROFBLOCKSATT 514

Definition at line 713 of file OSParseosol.tab.hpp.

7.36.1.258 **#define** NUMBEROFCOLUMNSATT 515

Definition at line 714 of file OSParseosol.tab.hpp.

7.36.1.259 **#define** NUMBEROFROWSATT 516

Definition at line 715 of file OSParseosol.tab.hpp.

7.36.1.260 **#define** NUMBEROFVALUESATT 517

Definition at line 716 of file OSParseosol.tab.hpp.

7.36.1.261 **#define** NUMBEROFVARIDXATT 518

Definition at line 717 of file OSParseosol.tab.hpp.

7.36.1.262 **#define** COEFATT 519

Definition at line 718 of file OSParseosol.tab.hpp.

7.36.1.263 **#define** BASEMATRIXIDXATT 520

Definition at line 719 of file OSParseosol.tab.hpp.

7.36.1.264 **#define** TARGETMATRIXFIRSTROWATT 521

Definition at line 720 of file OSParseosol.tab.hpp.

7.36.1.265 **#define** TARGETMATRIXFIRSTCOLATT 522

Definition at line 721 of file OSParseosol.tab.hpp.

7.36.1.266 **#define** BASEMATRIXSTARTROWATT 523

Definition at line 722 of file OSParseosol.tab.hpp.

7.36.1.267 **#define** BASEMATRIXSTARTCOLATT 524

Definition at line 723 of file OSParseosol.tab.hpp.

7.36.1.268 **#define** BASEMATRIXENDROWATT 525

Definition at line 724 of file OSParseosol.tab.hpp.

7.36.1.269 **#define** BASEMATRIXENDCOLATT 526

Definition at line 725 of file OSParseosol.tab.hpp.

7.36.1.270 **#define** SCALARMULTIPLIERATT 527

Definition at line 726 of file OSParseosol.tab.hpp.

7.36.1.271 **#define EMPTYBASETRANSPOSEATT 528**

Definition at line 727 of file OSParseosol.tab.hpp.

7.36.1.272 **#define BASETRANSPOSEATT 529**

Definition at line 728 of file OSParseosol.tab.hpp.

7.36.1.273 **#define ELEMENTSSTART 530**

Definition at line 729 of file OSParseosol.tab.hpp.

7.36.1.274 **#define ELEMENTSEND 531**

Definition at line 730 of file OSParseosol.tab.hpp.

7.36.1.275 **#define CONSTANTELEMENTSSTART 532**

Definition at line 731 of file OSParseosol.tab.hpp.

7.36.1.276 **#define CONSTANTELEMENTSEND 533**

Definition at line 732 of file OSParseosol.tab.hpp.

7.36.1.277 **#define STARTVECTORSTART 534**

Definition at line 733 of file OSParseosol.tab.hpp.

7.36.1.278 **#define STARTVECTOREND 535**

Definition at line 734 of file OSParseosol.tab.hpp.

7.36.1.279 **#define NONZEROSSTART 536**

Definition at line 735 of file OSParseosol.tab.hpp.

7.36.1.280 **#define NONZEROSEND 537**

Definition at line 736 of file OSParseosol.tab.hpp.

7.36.1.281 **#define INDEXESSTART 538**

Definition at line 737 of file OSParseosol.tab.hpp.

7.36.1.282 **#define INDEXESEND 539**

Definition at line 738 of file OSParseosol.tab.hpp.

7.36.1.283 **#define VALUESSTART 540**

Definition at line 739 of file OSParseosol.tab.hpp.

7.36.1.284 **#define VALUESEND 541**

Definition at line 740 of file OSParseosol.tab.hpp.

7.36.1.285 **#define VARREFERENCEELEMENTSSTART 542**

Definition at line 741 of file OSParseosol.tab.hpp.

7.36.1.286 **#define VARREFERENCEELEMENTSEND 543**

Definition at line 742 of file OSParseosol.tab.hpp.

7.36.1.287 **#define LINEARELEMENTSSTART 544**

Definition at line 743 of file OSParseosol.tab.hpp.

7.36.1.288 **#define LINEARELEMENTSEND 545**

Definition at line 744 of file OSParseosol.tab.hpp.

7.36.1.289 **#define GENERALELEMENTSSTART 546**

Definition at line 745 of file OSParseosol.tab.hpp.

7.36.1.290 **#define GENERALELEMENTSEND 547**

Definition at line 746 of file OSParseosol.tab.hpp.

7.36.1.291 **#define CONREFERENCEELEMENTSSTART 548**

Definition at line 747 of file OSParseosol.tab.hpp.

7.36.1.292 **#define CONREFERENCEELEMENTSEND 549**

Definition at line 748 of file OSParseosol.tab.hpp.

7.36.1.293 **#define VALUETYPEATT 550**

Definition at line 749 of file OSParseosol.tab.hpp.

7.36.1.294 **#define OBJREFERENCEELEMENTSSTART 551**

Definition at line 750 of file OSParseosol.tab.hpp.

7.36.1.295 **#define OBJREFERENCEELEMENTSEND 552**

Definition at line 751 of file OSParseosol.tab.hpp.

7.36.1.296 **#define PATTERNELEMENTSSTART 553**

Definition at line 752 of file OSParseosol.tab.hpp.

7.36.1.297 **#define PATTERNELEMENTSEND 554**

Definition at line 753 of file OSParseosol.tab.hpp.

7.36.1.298 **#define VARIDXSTART 555**

Definition at line 754 of file OSParseosol.tab.hpp.

7.36.1.299 **#define VARIDXEND 556**

Definition at line 755 of file OSParseosol.tab.hpp.

7.36.1.300 **#define TRANSFORMATIONSTART 557**

Definition at line 756 of file OSParseosol.tab.hpp.

7.36.1.301 **#define TRANSFORMATIONEND 558**

Definition at line 757 of file OSParseosol.tab.hpp.

7.36.1.302 **#define COLOFFSETSSTART 559**

Definition at line 758 of file OSParseosol.tab.hpp.

7.36.1.303 **#define COLOFFSETSEND 560**

Definition at line 759 of file OSParseosol.tab.hpp.

7.36.1.304 **#define ROWOFFSETSSTART 561**

Definition at line 760 of file OSParseosol.tab.hpp.

7.36.1.305 **#define ROWOFFSETSEND 562**

Definition at line 761 of file OSParseosol.tab.hpp.

7.36.1.306 **#define EMPTYROWMAJORATT 563**

Definition at line 762 of file OSParseosol.tab.hpp.

7.36.1.307 **#define ROWMAJORATT 564**

Definition at line 763 of file OSParseosol.tab.hpp.

7.36.1.308 **#define BLOCKROWIDXATT 565**

Definition at line 764 of file OSParseosol.tab.hpp.

7.36.1.309 **#define BLOCKCOLIDXATT 566**

Definition at line 765 of file OSParseosol.tab.hpp.

7.36.1.310 **#define DUMMY 567**

Definition at line 766 of file OSParseosol.tab.hpp.

7.36.1.311 **#define NONLINEAREXPRESSIONSSTART 568**

Definition at line 767 of file OSParseosol.tab.hpp.

7.36.1.312 **#define NONLINEAREXPRESSIONSEND 569**

Definition at line 768 of file OSParseosol.tab.hpp.

7.36.1.313 **#define** NUMBEROFNONLINEAREXPRESSIONS 570

Definition at line 769 of file OSParseosol.tab.hpp.

7.36.1.314 **#define** NLSTART 571

Definition at line 770 of file OSParseosol.tab.hpp.

7.36.1.315 **#define** NLEND 572

Definition at line 771 of file OSParseosol.tab.hpp.

7.36.1.316 **#define** MATRIXEXPRESSIONSSTART 573

Definition at line 772 of file OSParseosol.tab.hpp.

7.36.1.317 **#define** MATRIXEXPRESSIONSEND 574

Definition at line 773 of file OSParseosol.tab.hpp.

7.36.1.318 **#define** NUMBEROFEXPR 575

Definition at line 774 of file OSParseosol.tab.hpp.

7.36.1.319 **#define** EXPRSTART 576

Definition at line 775 of file OSParseosol.tab.hpp.

7.36.1.320 **#define** EXPREND 577

Definition at line 776 of file OSParseosol.tab.hpp.

7.36.1.321 **#define** NUMBEROFMATRIXTERMSATT 578

Definition at line 777 of file OSParseosol.tab.hpp.

7.36.1.322 **#define** MATRIXTERMSTART 579

Definition at line 778 of file OSParseosol.tab.hpp.

7.36.1.323 **#define** MATRIXTERMEND 580

Definition at line 779 of file OSParseosol.tab.hpp.

7.36.1.324 **#define** POWERSTART 581

Definition at line 780 of file OSParseosol.tab.hpp.

7.36.1.325 **#define** POWEREND 582

Definition at line 781 of file OSParseosol.tab.hpp.

7.36.1.326 **#define** PLUSSTART 583

Definition at line 782 of file OSParseosol.tab.hpp.

7.36.1.327 **#define PLUSEND 584**

Definition at line 783 of file OSParseosol.tab.hpp.

7.36.1.328 **#define MINUSSTART 585**

Definition at line 784 of file OSParseosol.tab.hpp.

7.36.1.329 **#define MINUSEND 586**

Definition at line 785 of file OSParseosol.tab.hpp.

7.36.1.330 **#define DIVIDESTART 587**

Definition at line 786 of file OSParseosol.tab.hpp.

7.36.1.331 **#define DIVIDEEND 588**

Definition at line 787 of file OSParseosol.tab.hpp.

7.36.1.332 **#define LNSTART 589**

Definition at line 788 of file OSParseosol.tab.hpp.

7.36.1.333 **#define LNEND 590**

Definition at line 789 of file OSParseosol.tab.hpp.

7.36.1.334 **#define SQRTSTART 591**

Definition at line 790 of file OSParseosol.tab.hpp.

7.36.1.335 **#define SQRTEND 592**

Definition at line 791 of file OSParseosol.tab.hpp.

7.36.1.336 **#define SUMSTART 593**

Definition at line 792 of file OSParseosol.tab.hpp.

7.36.1.337 **#define SUMEND 594**

Definition at line 793 of file OSParseosol.tab.hpp.

7.36.1.338 **#define PRODUCTSTART 595**

Definition at line 794 of file OSParseosol.tab.hpp.

7.36.1.339 **#define PRODUCTEND 596**

Definition at line 795 of file OSParseosol.tab.hpp.

7.36.1.340 **#define EXPSTART 597**

Definition at line 796 of file OSParseosol.tab.hpp.

7.36.1.341 `#define EXPEND 598`

Definition at line 797 of file `OSParseosol.tab.hpp`.

7.36.1.342 `#define NEGATESTART 599`

Definition at line 798 of file `OSParseosol.tab.hpp`.

7.36.1.343 `#define NEGATEEND 600`

Definition at line 799 of file `OSParseosol.tab.hpp`.

7.36.1.344 `#define IFSTART 601`

Definition at line 800 of file `OSParseosol.tab.hpp`.

7.36.1.345 `#define IFEND 602`

Definition at line 801 of file `OSParseosol.tab.hpp`.

7.36.1.346 `#define SQUARESTART 603`

Definition at line 802 of file `OSParseosol.tab.hpp`.

7.36.1.347 `#define SQUAREEND 604`

Definition at line 803 of file `OSParseosol.tab.hpp`.

7.36.1.348 `#define COSSTART 605`

Definition at line 804 of file `OSParseosol.tab.hpp`.

7.36.1.349 `#define COSEND 606`

Definition at line 805 of file `OSParseosol.tab.hpp`.

7.36.1.350 `#define SINSTART 607`

Definition at line 806 of file `OSParseosol.tab.hpp`.

7.36.1.351 `#define SINEND 608`

Definition at line 807 of file `OSParseosol.tab.hpp`.

7.36.1.352 `#define VARIABLESTART 609`

Definition at line 808 of file `OSParseosol.tab.hpp`.

7.36.1.353 `#define VARIABLEEND 610`

Definition at line 809 of file `OSParseosol.tab.hpp`.

7.36.1.354 `#define ABSSTART 611`

Definition at line 810 of file `OSParseosol.tab.hpp`.

7.36.1.355 **#define ABSEND 612**

Definition at line 811 of file OSParseosol.tab.hpp.

7.36.1.356 **#define ERFSTART 613**

Definition at line 812 of file OSParseosol.tab.hpp.

7.36.1.357 **#define ERFEND 614**

Definition at line 813 of file OSParseosol.tab.hpp.

7.36.1.358 **#define MAXSTART 615**

Definition at line 814 of file OSParseosol.tab.hpp.

7.36.1.359 **#define MAXEND 616**

Definition at line 815 of file OSParseosol.tab.hpp.

7.36.1.360 **#define ALLDIFFSTART 617**

Definition at line 816 of file OSParseosol.tab.hpp.

7.36.1.361 **#define ALLDIFFEND 618**

Definition at line 817 of file OSParseosol.tab.hpp.

7.36.1.362 **#define MINSTART 619**

Definition at line 818 of file OSParseosol.tab.hpp.

7.36.1.363 **#define MINEND 620**

Definition at line 819 of file OSParseosol.tab.hpp.

7.36.1.364 **#define ESTART 621**

Definition at line 820 of file OSParseosol.tab.hpp.

7.36.1.365 **#define EEND 622**

Definition at line 821 of file OSParseosol.tab.hpp.

7.36.1.366 **#define PISTART 623**

Definition at line 822 of file OSParseosol.tab.hpp.

7.36.1.367 **#define PIEND 624**

Definition at line 823 of file OSParseosol.tab.hpp.

7.36.1.368 **#define TIMESSTART 625**

Definition at line 824 of file OSParseosol.tab.hpp.

7.36.1.369 **#define TIMESEND 626**

Definition at line 825 of file OSParseosol.tab.hpp.

7.36.1.370 **#define NUMBERSTART 627**

Definition at line 826 of file OSParseosol.tab.hpp.

7.36.1.371 **#define NUMBEREND 628**

Definition at line 827 of file OSParseosol.tab.hpp.

7.36.1.372 **#define MATRIXDETERMINANTSTART 629**

Definition at line 828 of file OSParseosol.tab.hpp.

7.36.1.373 **#define MATRIXDETERMINANTEND 630**

Definition at line 829 of file OSParseosol.tab.hpp.

7.36.1.374 **#define MATRIXTRACESTART 631**

Definition at line 830 of file OSParseosol.tab.hpp.

7.36.1.375 **#define MATRIXTRACEEND 632**

Definition at line 831 of file OSParseosol.tab.hpp.

7.36.1.376 **#define MATRXTOSCALARSTART 633**

Definition at line 832 of file OSParseosol.tab.hpp.

7.36.1.377 **#define MATRXTOSCALAREND 634**

Definition at line 833 of file OSParseosol.tab.hpp.

7.36.1.378 **#define MATRIXDIAGONALSTART 635**

Definition at line 834 of file OSParseosol.tab.hpp.

7.36.1.379 **#define MATRIXDIAGONALEND 636**

Definition at line 835 of file OSParseosol.tab.hpp.

7.36.1.380 **#define MATRIXDOTTIMESSTART 637**

Definition at line 836 of file OSParseosol.tab.hpp.

7.36.1.381 **#define MATRIXDOTTIMESEND 638**

Definition at line 837 of file OSParseosol.tab.hpp.

7.36.1.382 **#define MATRIXLOWERTRIANGLESTART 639**

Definition at line 838 of file OSParseosol.tab.hpp.

7.36.1.383 `#define MATRIXLOWERTRIANGLEEND 640`

Definition at line 839 of file OSParseosol.tab.hpp.

7.36.1.384 `#define MATRIXUPPERTRIANGLESTART 641`

Definition at line 840 of file OSParseosol.tab.hpp.

7.36.1.385 `#define MATRIXUPPERTRIANGLEEND 642`

Definition at line 841 of file OSParseosol.tab.hpp.

7.36.1.386 `#define MATRIXMERGESTART 643`

Definition at line 842 of file OSParseosol.tab.hpp.

7.36.1.387 `#define MATRIXMERGEEND 644`

Definition at line 843 of file OSParseosol.tab.hpp.

7.36.1.388 `#define MATRIXMINUSSTART 645`

Definition at line 844 of file OSParseosol.tab.hpp.

7.36.1.389 `#define MATRIXMINUSEND 646`

Definition at line 845 of file OSParseosol.tab.hpp.

7.36.1.390 `#define MATRIXNEGATESTART 647`

Definition at line 846 of file OSParseosol.tab.hpp.

7.36.1.391 `#define MATRIXNEGATEEND 648`

Definition at line 847 of file OSParseosol.tab.hpp.

7.36.1.392 `#define MATRIXPLUSSTART 649`

Definition at line 848 of file OSParseosol.tab.hpp.

7.36.1.393 `#define MATRIXPLUSEND 650`

Definition at line 849 of file OSParseosol.tab.hpp.

7.36.1.394 `#define MATRIXTIMESSTART 651`

Definition at line 850 of file OSParseosol.tab.hpp.

7.36.1.395 `#define MATRIXTIMSEEND 652`

Definition at line 851 of file OSParseosol.tab.hpp.

7.36.1.396 `#define MATRIXPRODUCTSTART 653`

Definition at line 852 of file OSParseosol.tab.hpp.

7.36.1.397 **#define MATRIXPRODUCTEND 654**

Definition at line 853 of file OSParseosol.tab.hpp.

7.36.1.398 **#define MATRIXSCALARTIMESSTART 655**

Definition at line 854 of file OSParseosol.tab.hpp.

7.36.1.399 **#define MATRIXSCALARTIMESEND 656**

Definition at line 855 of file OSParseosol.tab.hpp.

7.36.1.400 **#define MATRIXSUBMATRIXATSTART 657**

Definition at line 856 of file OSParseosol.tab.hpp.

7.36.1.401 **#define MATRIXSUBMATRIXATEND 658**

Definition at line 857 of file OSParseosol.tab.hpp.

7.36.1.402 **#define MATRIXTRANSPOSESTART 659**

Definition at line 858 of file OSParseosol.tab.hpp.

7.36.1.403 **#define MATRIXTRANSPOSEEND 660**

Definition at line 859 of file OSParseosol.tab.hpp.

7.36.1.404 **#define MATRIXREFERENCESTART 661**

Definition at line 860 of file OSParseosol.tab.hpp.

7.36.1.405 **#define MATRIXREFERENCEEND 662**

Definition at line 861 of file OSParseosol.tab.hpp.

7.36.1.406 **#define IDENTITYMATRIXSTART 663**

Definition at line 862 of file OSParseosol.tab.hpp.

7.36.1.407 **#define IDENTITYMATRIXEND 664**

Definition at line 863 of file OSParseosol.tab.hpp.

7.36.1.408 **#define MATRIXINVERSESTART 665**

Definition at line 864 of file OSParseosol.tab.hpp.

7.36.1.409 **#define MATRIXINVERSEEND 666**

Definition at line 865 of file OSParseosol.tab.hpp.

7.36.1.410 **#define EMPTYINCLUDEDIAGONALATT 667**

Definition at line 866 of file OSParseosol.tab.hpp.

7.36.1.411 **#define INCLUDEDIAGONALATT 668**

Definition at line 867 of file OSParseosol.tab.hpp.

7.36.1.412 **#define IDATT 669**

Definition at line 868 of file OSParseosol.tab.hpp.

7.36.1.413 **#define YYSTYPE_IS_TRIVIAL 1**

Definition at line 885 of file OSParseosol.tab.hpp.

7.36.1.414 **#define yystype YYSTYPE /* obsolescent; will be withdrawn */**

Definition at line 886 of file OSParseosol.tab.hpp.

7.36.1.415 **#define YYSTYPE_IS_DECLARED 1**

Definition at line 887 of file OSParseosol.tab.hpp.

7.36.1.416 **#define yyltype YYLTYPE /* obsolescent; will be withdrawn */**

Definition at line 900 of file OSParseosol.tab.hpp.

7.36.1.417 **#define YYLTYPE_IS_DECLARED 1**

Definition at line 901 of file OSParseosol.tab.hpp.

7.36.1.418 **#define YYLTYPE_IS_TRIVIAL 1**

Definition at line 902 of file OSParseosol.tab.hpp.

7.36.2 Typedef Documentation

7.36.2.1 **typedef union YYSTYPE YYSTYPE**

7.36.2.2 **typedef struct YYLTYPE YYLTYPE**

7.36.3 Enumeration Type Documentation

7.36.3.1 **enum yytokentype**

Enumerator:

QUOTE

ATTRIBUTETEXT

ELEMENTTEXT

ITEMTEXT

INTEGER

DOUBLE

TWOQUOTES

ENDOFELEMENT

GREATERTHAN

OSILEND

INSTANCEDATAEND
INSTANCEDATASTARTEND
EMPTYIDATT
IDXONEATT
IDXTWOATT
VALUEATT
QUADRATICCOEFFICIENTSSTART
QUADRATICCOEFFICIENTSEND
NUMBEROFQTERMSATT
QTERMSTART
QTERMEND
MATRICESSTART
MATRICESEND
NUMBEROFMATRICESATT
CONESSTART
CONESEND
NUMBEROFCONESATT
NONNEGATIVECONESTART
NONNEGATIVECONEEND
NONPOSITIVECONESTART
NONPOSITIVECONEEND
ORTHANTCONESTART
ORTHANTCONEEND
POLYHEDRALCONESTART
POLYHEDRALCONEEND
QUADRATICCONESTART
QUADRATICCONEEND
ROTATEDQUADRATICCONESTART
ROTATEDQUADRATICCONEEND
SEMIDEFINITECONESTART
SEMIDEFINITECONEEND
PRODUCTCONESTART
PRODUCTCONEEND
INTERSECTIONCONESTART
INTERSECTIONCONEEND
DUALCONESTART
DUALCONEEND
POLARCONESTART
POLARCONEEND
DIRECTIONSTART
DIRECTIONEND
FACTORSSTART

FACTORSEND
COMPONENTSSTART
COMPONENTSEND
NORMSCALEFACTORATT
DISTORTIONMATRIXIDXATT
AXISDIRECTIONATT
FIRSTAXISDIRECTIONATT
SECONDAXISDIRECTIONATT
EMPTYSEMIDEFINITENESSATT
SEMIDEFINITENESSATT
REFERENCEMATRIXIDXATT
MATRIXPROGRAMMINGSTART
MATRIXPROGRAMMINGEND
VARTYPEATT
MATRIXVARIABLESSTART
MATRIXVARIABLESEND
NUMBEROFMATRIXVARATT
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJECTIVESSTART
MATRIXOBJECTIVESEND
NUMBEROFMATRIXOBJATT
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTRAINTSSTART
MATRIXCONSTRAINTSEND
NUMBEROFMATRIXCONATT
MATRIXCONSTART
MATRIXCONEND
MATRIXIDXATT
LBMATRIXIDXATT
LBCONEIDXATT
UBMATRIXIDXATT
UBCONEIDXATT
TEMPLATEMATRIXIDXATT
VARREFERENCEMATRIXIDXATT
OBJREFERENCEMATRIXIDXATT
CONREFERENCEMATRIXIDXATT
ORDERCONEIDXATT
CONSTANTMATRIXIDXATT
TIMEDOMAINSTART
TIMEDOMAINEND

STAGESSTART
STAGESEND
STAGESTART
STAGEEND
NUMBEROFSTAGESATT
HORIZONATT
STARTATT
VARIABLESSTART
CONSTRAINTSSTART
OBJECTIVESSTART
VARIABLESEND
CONSTRAINTSEND
OBJECTIVESEND
NUMBEROFVARIABLESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOBJECTIVESATT
STARTIDXATT
VARSTART
VAREND
CONSTART
CONEND
OBJSTART
OBJEND
INTERVALSTART
INTERVALEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY

FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
ENUMERATIONSTART
ENUMERATIONEND
NUMBEROFELATT
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYNAMEATT
NAMEATT
EMPTYTYPEATT
TYPEATT
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT

NUMBEROFVARIDXATT
IDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPOSEATT
BASETRANSPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART
TRANSFORMATIONEND

COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND

IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND

MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMSEEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMSEEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANSPOSESTART
MATRIXTRANSPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE
TWOQUOTES
GREATERTHAN
ENDOFELEMENT
OSOLSTART
OSOLSTARTEMPT
OSOLATTRIBUTETEXT
SOLEND
NUMBEROFOTHEROPTIONSATT
NUMBEROFENUMERATIONSATT

NUMBEROFJOBIDSATT
NUMBEROFPATHSATT
NUMBEROFPATHPAIRSATT
FROMATT
TOATT
MAKECOPYATT
CATEGORYATT
TYPEATT
GROUPWEIGHTATT
NUMBEROFPROCESSESATT
NUMBEROFSOLVEROPTIONSATT
NUMBEROFSOSATT
NUMBEROFVARIABLESATT
NUMBEROFOBJECTIVESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOTHERVARIABLEOPTIONSATT
NUMBEROFOTHEROBJECTIVEOPTIONSATT
NUMBEROFOTHERCONSTRAINTOPTIONSATT
NUMBEROFITEMSATT
NUMBEROFVARATT
NUMBEROFOBJATT
NUMBEROFCONATT
NUMBEROFELATT
NAMEATT
IDXATT
SOSIDXATT
VALUEATT
UNITATT
DESCRIPTIONATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT
OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
EMPTYTYPEATT
EMPTYNAMEATT
EMPTYCATEGORYATT
EMPTYDESCRIPTIONATT
EMPTYUNITATT

EMPTYVALUEATT
EMPTYLBVALUEATT
EMPTYUBVALUEATT
LBVALUEATT
UBVALUEATT
EMPTYLBDUALVALUEATT
EMPTYUBDUALVALUEATT
LBDUALVALUEATT
UBDUALVALUEATT
SOLVERATT
EMPTYSOLVERATT
WEIGHTATT
EMPTYWEIGHTATT
TRANSPORTTYPEATT
LOCATIONTYPEATT
GENERALSTART
GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND
JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
SERVICEURISTART
SERVICEURIEND
SERVICENAMESTART
SERVICENAMEEND
INSTANCENAMESTART
INSTANCENAMEEND
INSTANCELOCATIONSTART
INSTANCELOCATIONEND
JOBIDSTART
JOBIDEND
SOLVERTOINVOKESTART
SOLVERTOINVOKEEND
LICENSESTART
LICENSEEND
USERNAMESTART
USERNAMEEND
PASSWORDSTART

PASSWORDEND
CONTACTSTART
CONTACTEND
OTHEROPTIONSSTART
OTHEROPTIONSSEND
OTHERSTART
OTHEREND
MINDISKSPACESTART
MINDISKSPACEEND
MINMEMORYSTART
MINMEMORYEND
MINCPUSPEEDSTART
MINCPUSPEEDEND
MINCPUNUMBERSTART
MINCPUNUMBEREND
SERVICETYPESTART
SERVICETYPEEND
MAXTimestart
MAXTIMEEND
REQUESTEDSTARTTIMESTART
REQUESTEDSTARTTIMEEND
DEPENDENCIESSTART
DEPENDENCIESEND
REQUIREDDIRECTORIESSTART
REQUIREDDIRECTORIESEND
REQUIREDFILESSTART
REQUIREDFILESEND
PATHSTART
PATHEND
PATHPAIRSTART
PATHPAIREND
DIRECTORIESTOMAKESTART
DIRECTORIESTOMAKEEND
FILESTOMAKESTART
FILESTOMAKEEND
DIRECTORIESTODELETESTART
DIRECTORIESTODELETEEND
FILESTODELETESTART
FILESTODELETEEND
INPUTDIRECTORIESTOMOVESTART
INPUTDIRECTORIESTOMOVEEND
INPUTFILESTOMOVESTART

INPUTFILESTOMOVEEND
OUTPUTDIRECTORIESTOMOVESTART
OUTPUTDIRECTORIESTOMOVEEND
OUTPUTFILESTOMOVESTART
OUTPUTFILESTOMOVEEND
PROCESSESTOKILLSTART
PROCESSESTOKILLEND
PROCESSSTART
PROCESSEND
VARIABLESSTART
VARIABLESEND
INITIALVARIABLEVALUESSTART
INITIALVARIABLEVALUESEND
VARSTART
VAREND
INITIALVARIABLEVALUESSTRINGSTART
INITIALVARIABLEVALUESSTRINGEND
INITIALBASISSTATUSSTART
INITIALBASISSTATUSEND
BASICSTART
BASICEND
ATUPPERSTART
ATUPPEREND
ATLOWERSTART
ATLOWEREND
ATEQUALITYSTART
ATEQUALITYEND
SUPERBASICSTART
SUPERBASICEND
ISFREESTART
ISFREEEND
UNKNOWNSTART
UNKNOWNEND
INTEGERVARIABLEBRANCHINGWEIGHTSSTART
INTEGERVARIABLEBRANCHINGWEIGHTSEND
SOSVARIABLEBRANCHINGWEIGHTSSTART
SOSVARIABLEBRANCHINGWEIGHTSEND
SOSSTART
SOSEND
OBJECTIVESSTART
OBJECTIVESEND
INITIALOBJECTIVEVALUESSTART

INITIALOBJECTIVEVALUESEND
OBJSTART
OBJEND
INITIALOBJECTIVEBOUNDSSTART
INITIALOBJECTIVEBOUNDSEND
CONSTRAINTSSTART
CONSTRAINTSEND
INITIALCONSTRAINTVALUESSTART
INITIALCONSTRAINTVALUESEND
CONSTART
CONEND
INITIALDUALVALUESSTART
INITIALDUALVALUESEND
SOLVEROPTIONSSTART
SOLVEROPTIONSEND
SOLVEROPTIONSTART
SOLVEROPTIONEND
ENUMERATIONSTART
ENUMERATIONEND
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND

FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
NUMBEROFVARIDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT

BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPOSEATT
BASETRANSPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT

DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND

VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND

MATRIXTIMESSTART
MATRIXTIMESEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMESEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANSPOSESTART
MATRIXTRANSPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE
TWOQUOTES
GREATERTHAN
ENDOFELEMENT
OSRLSTART
OSRLSTARTEMPT
OSRLATTRIBUTETEXT
OSRLEND
NUMBEROFCONATT
NUMBEROFCONSTRAINTSATT
NUMBEROFELATT
NUMBEROFENUMERATIONSATT
NUMBEROFIDXATT
NUMBEROFITEMSATT
NUMBEROFOBJATT
NUMBEROFOBJECTIVESATT
NUMBEROFOTHERCONSTRAINTRESULTSATT
NUMBEROFOTHEROBJECTIVERESULTSATT

NUMBEROFOTHERRESULTSATT
NUMBEROFOTHERSOLUTIONRESULTSATT
NUMBEROFOTHERVARIABLERESULTSATT
NUMBEROFSOLUTIONSATT
NUMBEROFSOLVEROUTPUTSATT
NUMBEROFSUBSTATUSESATT
NUMBEROFTIMESATT
NUMBEROFVARATT
NUMBEROFVARIABLESATT
NUMBEROFVARIDXATT
TARGETOBJECTIVEIDXATT
IDXATT
INCRATT
MULTATT
SIZEOFATT
CATEGORYATT
EMPTYCATEGORYATT
DESCRIPTIONATT
EMPTYDESCRIPTIONATT
NAMEATT
EMPTYNAMEATT
TYPEATT
EMPTYTYPEATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT
OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
UNITATT
EMPTYUNITATT
VALUEATT
EMPTYVALUEATT
WEIGHTEDOBJECTIVESATT
EMPTYWEIGHTEDOBJECTIVESATT
TARGETOBJECTIVENAMEATT
EMPTYTARGETOBJECTIVENAMEATT
HEADERSTART
HEADEREND
GENERALSTART

GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND
JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
ITEMSTART
ITEMEND
ITEMSTARTANDEND
ITEMEMPTY
ACTUALSTARTTIMESTART
ACTUALSTARTTIMEEND
ATEQUALITYSTART
ATEQUALITYEND
ATLOWERSTART
ATLOWEREND
ATUPPERSTART
ATUPPEREND
AVAILABLECPUNUMBERSTART
AVAILABLECPUNUMBEREND
AVAILABLECPUSPEEDSTART
AVAILABLECPUSPEEDEND
AVAILABLEDISKSPACESTART
AVAILABLEDISKSPACEEND
AVAILABLEMEMORYSTART
AVAILABLEMEMORYEND
BASE64START
BASE64END
BASICSTART
BASICEND
BASISSTATUSSTART
BASISSTATUSEND
BASSTATUSSTART
BASSTATUSEND
CONSTART
CONEND
CONSTRAINTSSTART
CONSTRAINTSEND
CURRENTJOBCOUNTSTART

CURRENTJOBCOUNTEND
CURRENTSTATESTART
CURRENTSTATEEND
DUALVALUESSTART
DUALVALUESEND
ELSTART
ELEND
ENUMERATIONSTART
ENUMERATIONEND
ENDTIMESTART
ENDTIMEEND
GENERALSTATUSSTART
GENERALSTATUSEND
GENERALSUBSTATUSSTART
GENERALSUBSTATUSEND
IDXSTART
IDXEND
INSTANCENAMESTART
INSTANCENAMEEND
ISFREESTART
ISFREEEND
JOBIDSTART
JOBIDEND
MESSAGESTART
MESSAGEEND
OBJSTART
OBJEND
OBJECTIVESSTART
OBJECTIVESEND
OPTIMIZATIONSOLUTIONSTATUSSTART
OPTIMIZATIONSOLUTIONSTATUSEND
OPTIMIZATIONSOLUTIONSUBSTATUSSTART
OPTIMIZATIONSOLUTIONSUBSTATUSEND
OTHERSTART
OTHEREND
OTHERRESULTSSTART
OTHERRESULTSEND
OTHERSOLUTIONRESULTSTART
OTHERSOLUTIONRESULTEND
OTHERSOLUTIONRESULTSSTART
OTHERSOLUTIONRESULTSEND
OTHERSOLVEROUTPUTSTART

OTHERSOLVEROUTPUTEND
SCHEDULEDSTARTTimestart
SCHEDULEDSTARTTIMEEND
SERVICENAMESTART
SERVICENAMEEND
SERVICEURISTART
SERVICEURIEND
SERVICEUTILIZATIONSTART
SERVICEUTILIZATIONEND
SOLUTIONSTART
SOLUTIONEND
SOLVERINVOKEDSTART
SOLVERINVOKEDEND
SOLVEROUTPUTSTART
SOLVEROUTPUTEND
STATUSSTART
STATUSEND
SUBMITTimestart
SUBMITTIMEEND
SUBSTATUSSTART
SUBSTATUSEND
SUPERBASICSTART
SUPERBASICEND
SYSTEMINFORMATIONSTART
SYSTEMINFORMATIONEND
Timestart
TIMEEND
TIMESERVICESTARTEDSTART
TIMESERVICESTARTEDEND
TIMESTAMPSTART
TIMESTAMPEND
TIMINGINFORMATIONSTART
TIMINGINFORMATIONEND
TOTALJOBSSOFARSTART
TOTALJOBSSOFAREND
UNKNOWNSTART
UNKNOWNEND
USEDCPUNUMBERSTART
USEDCPUNUMBEREND
USEDCPUSPEEDSTART
USEDCPUSPEEDEND
USEDDISKSPACESTART

USEDISKSPACEEND
USEDMEMORYSTART
USEDMEMORYEND
VALUESSTRINGSTART
VALUESSTRINGEND
VARSTART
VAREND
VARIABLESSTART
VARIABLESEND
VARIDXSTART
VARIDXEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART

BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPPOSEATT
BASETRANSPPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART

GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND

SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND

MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMESEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMESEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANPOSESTART
MATRIXTRANPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT

Definition at line 41 of file OSParseosol.tab.hpp.

7.37 /home/ted/COIN/trunk/OS/src/OSParsers/OSParseosrl.tab.hpp File Reference

Classes

- union [YYSTYPE](#)
- struct [YYLTYPE](#)

Macros

- #define [ATTRIBUTETEXT](#) 258
- #define [ELEMENTTEXT](#) 259
- #define [ITEMTEXT](#) 260
- #define [INTEGER](#) 261
- #define [DOUBLE](#) 262
- #define [QUOTE](#) 263
- #define [TWOQUOTES](#) 264
- #define [GREATERTHAN](#) 265
- #define [ENDOFELEMENT](#) 266
- #define [OSRLSTART](#) 267
- #define [OSRLSTARTEEMPTY](#) 268
- #define [OSRLATTRIBUTETEXT](#) 269
- #define [OSRLEND](#) 270
- #define [NUMBEROFCONATT](#) 271
- #define [NUMBEROFCONSTRAINTSATT](#) 272
- #define [NUMBEROFELATT](#) 273
- #define [NUMBEROFENUMERATIONSATT](#) 274
- #define [NUMBEROFIDXATT](#) 275
- #define [NUMBEROFITEMSATT](#) 276
- #define [NUMBEROFOBJATT](#) 277
- #define [NUMBEROFOBJECTIVESATT](#) 278
- #define [NUMBEROFOTHERCONSTRAINTRESULTSATT](#) 279
- #define [NUMBEROFOTHEROBJECTIVERESULTSATT](#) 280
- #define [NUMBEROFOTHERRESULTSATT](#) 281
- #define [NUMBEROFOTHERSOLUTIONRESULTSATT](#) 282
- #define [NUMBEROFOTHERVARIABLERESULTSATT](#) 283
- #define [NUMBEROFSOLUTIONSATT](#) 284
- #define [NUMBEROFSOLVEROUTPUTSATT](#) 285
- #define [NUMBEROFSUBSTATUSESATT](#) 286
- #define [NUMBEROFTIMESATT](#) 287
- #define [NUMBEROFVARATT](#) 288
- #define [NUMBEROFVARIABLESATT](#) 289
- #define [NUMBEROFVARIDXATT](#) 290
- #define [TARGETOBJECTIVEIDXATT](#) 291
- #define [IDXATT](#) 292
- #define [INCRATT](#) 293
- #define [MULTATT](#) 294
- #define [SIZEOFATT](#) 295
- #define [CATEGORYATT](#) 296
- #define [EMPTYCATEGORYATT](#) 297
- #define [DESCRIPTIONATT](#) 298
- #define [EMPTYDESCRIPTIONATT](#) 299

- #define NAMEATT 300
- #define EMPTYNAMEATT 301
- #define TYPEATT 302
- #define EMPTYTYPEATT 303
- #define CONTYPEATT 304
- #define EMPTYCONTYPEATT 305
- #define ENUMTYPEATT 306
- #define EMPTYENUMTYPEATT 307
- #define OBJTYPEATT 308
- #define EMPTYOBJTYPEATT 309
- #define VARTYPEATT 310
- #define EMPTYVARTYPEATT 311
- #define UNITATT 312
- #define EMPTYUNITATT 313
- #define VALUEATT 314
- #define EMPTYVALUEATT 315
- #define WEIGHTEDOBJECTIVESATT 316
- #define EMPTYWEIGHTEDOBJECTIVESATT 317
- #define TARGETOBJECTIVENAMEATT 318
- #define EMPTYTARGETOBJECTIVENAMEATT 319
- #define HEADERSTART 320
- #define HEADEREND 321
- #define GENERALSTART 322
- #define GENERALEND 323
- #define SYSTEMSTART 324
- #define SYSTEMEND 325
- #define SERVICESTART 326
- #define SERVICEEND 327
- #define JOBSTART 328
- #define JOBEND 329
- #define OPTIMIZATIONSTART 330
- #define OPTIMIZATIONEND 331
- #define ITEMSTART 332
- #define ITEMEND 333
- #define ITEMSTARTANDEND 334
- #define ITEMEMPTY 335
- #define ACTUALSTARTTIMESTART 336
- #define ACTUALSTARTTIMEEND 337
- #define ATEQUALITYSTART 338
- #define ATEQUALITYEND 339
- #define ATLOWERSTART 340
- #define ATLOWEREND 341
- #define ATUPPERSTART 342
- #define ATUPPEREND 343
- #define AVAILABLECPUNUMBERSTART 344
- #define AVAILABLECPUNUMBEREND 345
- #define AVAILABLECPUSPEEDSTART 346
- #define AVAILABLECPUSPEEDEND 347
- #define AVAILABLEDISKSPACESTART 348
- #define AVAILABLEDISKSPACEEND 349
- #define AVAILABLEMEMORYSTART 350

- #define [AVAILABLEMEMORYEND](#) 351
- #define [BASE64START](#) 352
- #define [BASE64END](#) 353
- #define [BASICSTART](#) 354
- #define [BASICEND](#) 355
- #define [BASISSTATUSSTART](#) 356
- #define [BASISSTATUSEND](#) 357
- #define [BASSTATUSSTART](#) 358
- #define [BASSTATUSEND](#) 359
- #define [CONSTART](#) 360
- #define [CONEND](#) 361
- #define [CONSTRAINTSSTART](#) 362
- #define [CONSTRAINTSEND](#) 363
- #define [CURRENTJOBCOUNTSTART](#) 364
- #define [CURRENTJOBCOUNTEND](#) 365
- #define [CURRENTSTATESTART](#) 366
- #define [CURRENTSTATEEND](#) 367
- #define [DUALVALUESSTART](#) 368
- #define [DUALVALUESEND](#) 369
- #define [ELSTART](#) 370
- #define [EEND](#) 371
- #define [ENUMERATIONSTART](#) 372
- #define [ENUMERATIONEND](#) 373
- #define [ENDTIMESTART](#) 374
- #define [ENDTIMEEND](#) 375
- #define [GENERALSTATUSSTART](#) 376
- #define [GENERALSTATUSEND](#) 377
- #define [GENERALSUBSTATUSSTART](#) 378
- #define [GENERALSUBSTATUSEND](#) 379
- #define [IDXSTART](#) 380
- #define [IDXEND](#) 381
- #define [INSTANCENAMESTART](#) 382
- #define [INSTANCENAMEEND](#) 383
- #define [ISFREESTART](#) 384
- #define [ISFREEEND](#) 385
- #define [JOBIDSTART](#) 386
- #define [JOBIDEND](#) 387
- #define [MESSAGESTART](#) 388
- #define [MESSAGEEND](#) 389
- #define [OBJSTART](#) 390
- #define [OBJEND](#) 391
- #define [OBJECTIVESSTART](#) 392
- #define [OBJECTIVESEND](#) 393
- #define [OPTIMIZATIONSOLUTIONSTATUSSTART](#) 394
- #define [OPTIMIZATIONSOLUTIONSTATUSEND](#) 395
- #define [OPTIMIZATIONSOLUTIONSUBSTATUSSTART](#) 396
- #define [OPTIMIZATIONSOLUTIONSUBSTATUSEND](#) 397
- #define [OTHERSTART](#) 398
- #define [OTHEREND](#) 399
- #define [OTHERRESULTSSTART](#) 400
- #define [OTHERRESULTSEND](#) 401

- #define OTHERSOLUTIONRESULTSTART 402
- #define OTHERSOLUTIONRESULTEND 403
- #define OTHERSOLUTIONRESULTSSTART 404
- #define OTHERSOLUTIONRESULTSEND 405
- #define OTHERSOLVEROUTPUTSTART 406
- #define OTHERSOLVEROUTPUTEND 407
- #define SCHEDULEDSTARTTIMESTART 408
- #define SCHEDULEDSTARTTIMEEND 409
- #define SERVICENAMESTART 410
- #define SERVICENAMEEND 411
- #define SERVICEURISTART 412
- #define SERVICEURIEND 413
- #define SERVICEUTILIZATIONSTART 414
- #define SERVICEUTILIZATIONEND 415
- #define SOLUTIONSTART 416
- #define SOLUTIONEND 417
- #define SOLVERINVOKEDSTART 418
- #define SOLVERINVOKEDEND 419
- #define SOLVEROUTPUTSTART 420
- #define SOLVEROUTPUTEND 421
- #define STATUSSTART 422
- #define STATUSEND 423
- #define SUBMITTIMESTART 424
- #define SUBMITTIMEEND 425
- #define SUBSTATUSSTART 426
- #define SUBSTATUSEND 427
- #define SUPERBASICSTART 428
- #define SUPERBASICEND 429
- #define SYSTEMINFORMATIONSTART 430
- #define SYSTEMINFORMATIONEND 431
- #define TIMESTART 432
- #define TIMEEND 433
- #define TIMESERVICESTARTEDSTART 434
- #define TIMESERVICESTARTEDEND 435
- #define TIMESTAMPSTART 436
- #define TIMESTAMPEND 437
- #define TIMINGINFORMATIONSTART 438
- #define TIMINGINFORMATIONEND 439
- #define TOTALJOBSSOFARSTART 440
- #define TOTALJOBSSOFAREND 441
- #define UNKNOWNSTART 442
- #define UNKNOWNEND 443
- #define USEDCPUNUMBERSTART 444
- #define USEDCPUNUMBEREND 445
- #define USEDCPUSPEEDSTART 446
- #define USEDCPUSPEEDEND 447
- #define USEDDISKSPACESTART 448
- #define USEDDISKSPACEEND 449
- #define USEDMEMORYSTART 450
- #define USEDMEMORYEND 451
- #define VALUESSTRINGSTART 452

- #define [VALUESSTRINGEND](#) 453
- #define [VARSTART](#) 454
- #define [VAREND](#) 455
- #define [VARIABLESSTART](#) 456
- #define [VARIABLESEND](#) 457
- #define [VARIDXSTART](#) 458
- #define [VARIDXEND](#) 459
- #define [MATRIXVARSTART](#) 460
- #define [MATRIXVAREND](#) 461
- #define [MATRIXOBJSTART](#) 462
- #define [MATRIXOBJEND](#) 463
- #define [MATRIXCONSTART](#) 464
- #define [MATRIXCONEND](#) 465
- #define [FILENAMESTART](#) 466
- #define [FILENAMEEND](#) 467
- #define [FILENAMEEMPTY](#) 468
- #define [FILENAMESTARTANDEND](#) 469
- #define [FILESOURCESTART](#) 470
- #define [FILESOURCEEND](#) 471
- #define [FILESOURCEEMPTY](#) 472
- #define [FILESOURCESTARTANDEND](#) 473
- #define [FILEDESCRIPTIONSTART](#) 474
- #define [FILEDESCRIPTIONEND](#) 475
- #define [FILEDESCRIPTIONEMPTY](#) 476
- #define [FILEDESCRIPTIONSTARTANDEND](#) 477
- #define [FILECREATORSTART](#) 478
- #define [FILECREATOREND](#) 479
- #define [FILECREATOREMPTY](#) 480
- #define [FILECREATORSTARTANDEND](#) 481
- #define [FILELICENCESTART](#) 482
- #define [FILELICENCEEND](#) 483
- #define [FILELICENCEEMPTY](#) 484
- #define [FILELICENCESTARTANDEND](#) 485
- #define [MATRIXSTART](#) 486
- #define [MATRIXEND](#) 487
- #define [BASEMATRIXEND](#) 488
- #define [BASEMATRIXSTART](#) 489
- #define [BLOCKSTART](#) 490
- #define [BLOCKEND](#) 491
- #define [BLOCKSSTART](#) 492
- #define [BLOCKSEND](#) 493
- #define [EMPTYSHAPEATT](#) 494
- #define [SHAPEATT](#) 495
- #define [EMPTYSYMMETRYATT](#) 496
- #define [SYMMETRYATT](#) 497
- #define [EMPTYNEGATIVEPATTERNATT](#) 498
- #define [NEGATIVEPATTERNATT](#) 499
- #define [CONSTANTATT](#) 500
- #define [NUMBEROFBLOCKSATT](#) 501
- #define [NUMBEROFCOLUMNSATT](#) 502
- #define [NUMBEROFROWSATT](#) 503

- #define [NUMBEROFVALUESATT](#) 504
- #define [COEFATT](#) 505
- #define [BASEMATRIXIDXATT](#) 506
- #define [TARGETMATRIXFIRSTROWATT](#) 507
- #define [TARGETMATRIXFIRSTCOLATT](#) 508
- #define [BASEMATRIXSTARTROWATT](#) 509
- #define [BASEMATRIXSTARTCOLATT](#) 510
- #define [BASEMATRIXENDROWATT](#) 511
- #define [BASEMATRIXENDCOLATT](#) 512
- #define [SCALARMULTIPLIERATT](#) 513
- #define [EMPTYBASETRANPOSEATT](#) 514
- #define [BASETRANPOSEATT](#) 515
- #define [ELEMENTSSTART](#) 516
- #define [ELEMENTSEND](#) 517
- #define [CONSTANTELEMENTSSTART](#) 518
- #define [CONSTANTELEMENTSEND](#) 519
- #define [STARTVECTORSTART](#) 520
- #define [STARTVECTOREND](#) 521
- #define [NONZEROSSTART](#) 522
- #define [NONZEROSEND](#) 523
- #define [INDEXESSTART](#) 524
- #define [INDEXESEND](#) 525
- #define [VALUESSTART](#) 526
- #define [VALUESEND](#) 527
- #define [VARREFERENCEELEMENTSSTART](#) 528
- #define [VARREFERENCEELEMENTSEND](#) 529
- #define [LINEARELEMENTSSTART](#) 530
- #define [LINEARELEMENTSEND](#) 531
- #define [GENERALELEMENTSSTART](#) 532
- #define [GENERALELEMENTSEND](#) 533
- #define [CONREFERENCEELEMENTSSTART](#) 534
- #define [CONREFERENCEELEMENTSEND](#) 535
- #define [VALUETYPEATT](#) 536
- #define [OBJREFERENCEELEMENTSSTART](#) 537
- #define [OBJREFERENCEELEMENTSEND](#) 538
- #define [PATTERNELEMENTSSTART](#) 539
- #define [PATTERNELEMENTSEND](#) 540
- #define [TRANSFORMATIONSTART](#) 541
- #define [TRANSFORMATIONEND](#) 542
- #define [COLOFFSETSSTART](#) 543
- #define [COLOFFSETSEND](#) 544
- #define [ROWOFFSETSSTART](#) 545
- #define [ROWOFFSETSEND](#) 546
- #define [EMPTYROWMAJORATT](#) 547
- #define [ROWMAJORATT](#) 548
- #define [BLOCKROWIDXATT](#) 549
- #define [BLOCKCOLIDXATT](#) 550
- #define [DUMMY](#) 551
- #define [NONLINEAREXPRESSIONSSTART](#) 552
- #define [NONLINEAREXPRESSIONSEND](#) 553
- #define [NUMBEROFNONLINEAREXPRESSIONS](#) 554

- #define [NLSTART](#) 555
- #define [NLEND](#) 556
- #define [MATRIXEXPRESSIONSSTART](#) 557
- #define [MATRIXEXPRESSIONSEND](#) 558
- #define [NUMBEROFEXPR](#) 559
- #define [EXPRSTART](#) 560
- #define [EXPREND](#) 561
- #define [NUMBEROFMATRIXTERMSATT](#) 562
- #define [MATRIXTERMSTART](#) 563
- #define [MATRIXTERMEND](#) 564
- #define [POWERSTART](#) 565
- #define [POWEREND](#) 566
- #define [PLUSSTART](#) 567
- #define [PLUSEND](#) 568
- #define [MINUSSTART](#) 569
- #define [MINUSEND](#) 570
- #define [DIVIDESTART](#) 571
- #define [DIVIDEEND](#) 572
- #define [LNSTART](#) 573
- #define [LNEND](#) 574
- #define [SQRTSTART](#) 575
- #define [SQRTEND](#) 576
- #define [SUMSTART](#) 577
- #define [SUMEND](#) 578
- #define [PRODUCTSTART](#) 579
- #define [PRODUCTEND](#) 580
- #define [EXPSTART](#) 581
- #define [EXPEND](#) 582
- #define [NEGATESTART](#) 583
- #define [NEGATEEND](#) 584
- #define [IFSTART](#) 585
- #define [IFEND](#) 586
- #define [SQUARESTART](#) 587
- #define [SQUAREEND](#) 588
- #define [COSSTART](#) 589
- #define [COSEND](#) 590
- #define [SINSTART](#) 591
- #define [SINEND](#) 592
- #define [VARIABLESTART](#) 593
- #define [VARIABLEEND](#) 594
- #define [ABSSTART](#) 595
- #define [ABSEND](#) 596
- #define [ERFSTART](#) 597
- #define [ERFEND](#) 598
- #define [MAXSTART](#) 599
- #define [MAXEND](#) 600
- #define [ALLDIFFSTART](#) 601
- #define [ALLDIFFEND](#) 602
- #define [MINSTART](#) 603
- #define [MINEND](#) 604
- #define [ESTART](#) 605

- #define EEND 606
- #define PISTART 607
- #define PIEND 608
- #define TIMESSTART 609
- #define TIMESEND 610
- #define NUMBERSTART 611
- #define NUMBEREND 612
- #define MATRIXDETERMINANTSTART 613
- #define MATRIXDETERMINANTEND 614
- #define MATRIXTRACESTART 615
- #define MATRIXTRACEEND 616
- #define MATRXTOSCALARSTART 617
- #define MATRXTOSCALAREND 618
- #define MATRIXDIAGONALSTART 619
- #define MATRIXDIAGONALEND 620
- #define MATRIXDOTTIMESSTART 621
- #define MATRIXDOTTIMESEND 622
- #define MATRIXLOWERTRIANGLESTART 623
- #define MATRIXLOWERTRIANGLEEND 624
- #define MATRIXUPPERTRIANGLESTART 625
- #define MATRIXUPPERTRIANGLEEND 626
- #define MATRIXMERGESTART 627
- #define MATRIXMERGEEND 628
- #define MATRIXMINUSSTART 629
- #define MATRIXMINUSEND 630
- #define MATRIXNEGATESTART 631
- #define MATRIXNEGATEEND 632
- #define MATRIXPLUSSTART 633
- #define MATRIXPLUSEND 634
- #define MATRXTIMESSTART 635
- #define MATRXTIMESEND 636
- #define MATRIXPRODUCTSTART 637
- #define MATRIXPRODUCTEND 638
- #define MATRIXSCALARTIMESSTART 639
- #define MATRIXSCALARTIMESEND 640
- #define MATRIXSUBMATRIXATSTART 641
- #define MATRIXSUBMATRIXATEND 642
- #define MATRIXTRANSPOSESTART 643
- #define MATRIXTRANSPOSEEND 644
- #define MATRIXREFERENCESTART 645
- #define MATRIXREFERENCEEND 646
- #define IDENTITYMATRIXSTART 647
- #define IDENTITYMATRIXEND 648
- #define MATRIXINVERSESTART 649
- #define MATRIXINVERSEEND 650
- #define EMPTYINCLUDEDIAGONALATT 651
- #define INCLUDEDIAGONALATT 652
- #define IDATT 653
- #define YYSTYPE_IS_TRIVIAL 1
- #define yystype YYSTYPE /* obsolescent; will be withdrawn */
- #define YYSTYPE_IS_DECLARED 1
- #define yylytype YYLTYPE /* obsolescent; will be withdrawn */
- #define YYLTYPE_IS_DECLARED 1
- #define YYLTYPE_IS_TRIVIAL 1

Typedefs

- typedef union [YYSTYPE](#) YYSTYPE
- typedef struct [YYLTYPE](#) YYLTYPE

Enumerations

- enum [yytokentype](#) {
[QUOTE](#) = 258, [ATTRIBUTETEXT](#) = 259, [ELEMENTTEXT](#) = 260, [ITEMTEXT](#) = 261,
[INTEGER](#) = 262, [DOUBLE](#) = 263, [TWOQUOTES](#) = 264, [ENDOFELEMENT](#) = 265,
[GREATERTHAN](#) = 266, [OSILEND](#) = 267, [INSTANCEDATAEND](#) = 268, [INSTANCEDATASTARTEND](#) = 269,
[EMPTYIDATT](#) = 270, [IDXONEATT](#) = 271, [IDXTWOATT](#) = 272, [VALUEATT](#) = 273,
[QUADRATICCOEFFICIENTSSTART](#) = 274, [QUADRATICCOEFFICIENTSEND](#) = 275, [NUMBEROFQTERMSA-](#)
[TT](#) = 276, [QTERMSTART](#) = 277,
[QTERMEND](#) = 278, [MATRICESSTART](#) = 279, [MATRICESEND](#) = 280, [NUMBEROFMATRICESATT](#) = 281,
[CONESSTART](#) = 282, [CONESEND](#) = 283, [NUMBEROFCONESATT](#) = 284, [NONNEGATIVECONESTART](#) =
285,
[NONNEGATIVECONEEND](#) = 286, [NONPOSITIVECONESTART](#) = 287, [NONPOSITIVECONEEND](#) = 288, [ORT-](#)
[HANTCONESTART](#) = 289,
[ORTHANTCONEEND](#) = 290, [POLYHEDRALCONESTART](#) = 291, [POLYHEDRALCONEEND](#) = 292, [QUADRAT-](#)
[ICCONESTART](#) = 293,
[QUADRATICCONEEND](#) = 294, [ROTATEDQUADRATICCONESTART](#) = 295, [ROTATEDQUADRATICCONEEN-](#)
[D](#) = 296, [SEMIDEFINITECONESTART](#) = 297,
[SEMIDEFINITECONEEND](#) = 298, [PRODUCTCONESTART](#) = 299, [PRODUCTCONEEND](#) = 300, [INTERSECTI-](#)
[ONCONESTART](#) = 301,
[INTERSECTIONCONEEND](#) = 302, [DUALCONESTART](#) = 303, [DUALCONEEND](#) = 304, [POLARCONESTART](#) =
305,
[POLARCONEEND](#) = 306, [DIRECTIONSTART](#) = 307, [DIRECTIONEND](#) = 308, [FACTORSSTART](#) = 309,
[FACTORSEND](#) = 310, [COMPONENTSSTART](#) = 311, [COMPONENTSEND](#) = 312, [NORMSCALEFACTORATT](#) =
313,
[DISTORTIONMATRIXIDXATT](#) = 314, [AXISDIRECTIONATT](#) = 315, [FIRSTAXISDIRECTIONATT](#) = 316, [SECON-](#)
[DAXISDIRECTIONATT](#) = 317,
[EMPTYSEMIDEFINITENESSATT](#) = 318, [SEMIDEFINITENESSATT](#) = 319, [REFERENCEMATRIXIDXATT](#) = 320,
[MATRIXPROGRAMMINGSTART](#) = 321,
[MATRIXPROGRAMMINGEND](#) = 322, [VARTYPEATT](#) = 323, [MATRIXVARIABLESSTART](#) = 324, [MATRIXVARI-](#)
[ABLESEND](#) = 325,
[NUMBEROFMATRIXVARATT](#) = 326, [MATRIXVARSTART](#) = 327, [MATRIXVAREND](#) = 328, [MATRIXOBJECTIV-](#)
[ESSTART](#) = 329,
[MATRIXOBJECTIVESEND](#) = 330, [NUMBEROFMATRIXOBJATT](#) = 331, [MATRIXOBJSTART](#) = 332, [MATRIXO-](#)
[BJEND](#) = 333,
[MATRIXCONSTRAINTSSTART](#) = 334, [MATRIXCONSTRAINTSEND](#) = 335, [NUMBEROFMATRIXCONATT](#) =
336, [MATRIXCONSTART](#) = 337,
[MATRIXCONEND](#) = 338, [MATRIXIDXATT](#) = 339, [LBMATRIXIDXATT](#) = 340, [LBCONEIDXATT](#) = 341,
[UBMATRIXIDXATT](#) = 342, [UBCONEIDXATT](#) = 343, [TEMPLATEMATRIXIDXATT](#) = 344, [VARREFERENCEMA-](#)
[TRIXIDXATT](#) = 345,
[OBJREFERENCEMATRIXIDXATT](#) = 346, [CONREFERENCEMATRIXIDXATT](#) = 347, [ORDERCONEIDXATT](#) =
348, [CONSTANTMATRIXIDXATT](#) = 349,
[TIMEDOMAINSTART](#) = 350, [TIMEDOMAINEND](#) = 351, [STAGESSTART](#) = 352, [STAGESEND](#) = 353,
[STAGESTART](#) = 354, [STAGEEND](#) = 355, [NUMBEROFSTAGESATT](#) = 356, [HORIZONATT](#) = 357,
[STARTATT](#) = 358, [VARIABLESSTART](#) = 359, [CONSTRAINTSSTART](#) = 360, [OBJECTIVESSTART](#) = 361,
[VARIABLESEND](#) = 362, [CONSTRAINTSEND](#) = 363, [OBJECTIVESEND](#) = 364, [NUMBEROFVARIABLESATT](#) =
365,
[NUMBEROFCONSTRAINTSATT](#) = 366, [NUMBEROFOBJECTIVESATT](#) = 367, [STARTIDXATT](#) = 368, [VARST-](#)

ART = 369,
VAREND = 370, CONSTART = 371, CONEND = 372, OBJSTART = 373,
OBJEND = 374, INTERVALSTART = 375, INTERVALEND = 376, HEADERSTART = 377,
HEADEREND = 378, FILENAMESTART = 379, FILENAMEEND = 380, FILENAMEEMPTY = 381,
FILENAMESTARTANDEND = 382, FILESOURCESTART = 383, FILESOURCEEND = 384, FILESOURCEEMP-
TY = 385,
FILESOURCESTARTANDEND = 386, FILEDESCRIPTIONSTART = 387, FILEDESCRIPTIONEND = 388, FILE-
DESCRIPTIONEMPTY = 389,
FILEDESCRIPTIONSTARTANDEND = 390, FILECREATORSTART = 391, FILECREATOREND = 392, FILECR-
EATOREMPTY = 393,
FILECREATORSTARTANDEND = 394, FILELICENCESTART = 395, FILELICENCEEND = 396, FILELICENCE-
EMPTY = 397,
FILELICENCESTARTANDEND = 398, ENUMERATIONSTART = 399, ENUMERATIONEND = 400, NUMERO-
FELATT = 401,
ITEMEMPTY = 402, ITEMSTART = 403, ITEMEND = 404, ITEMSTARTANDEND = 405,
BASE64START = 406, BASE64END = 407, INCRATT = 408, MULTATT = 409,
SIZEOFATT = 410, ELSTART = 411, ELEND = 412, MATRIXSTART = 413,
MATRIXEND = 414, BASEMATRIXEND = 415, BASEMATRIXSTART = 416, BLOCKSTART = 417,
BLOCKEND = 418, BLOCKSSTART = 419, BLOCKSEND = 420, EMPTYNAMEATT = 421,
NAMEATT = 422, EMPTYTYPEATT = 423, TYPEATT = 424, EMPTYSHAPEATT = 425,
SHAPEATT = 426, EMPTYSYMMETRYATT = 427, SYMMETRYATT = 428, EMPTYNEGATIVEPATTERNATT =
429,
NEGATIVEPATTERNATT = 430, CONSTANTATT = 431, NUMBEROFBLOCKSATT = 432, NUMBEROFCOLU-
MNSATT = 433,
NUMBEROFROWSATT = 434, NUMBEROFVALUESATT = 435, NUMBEROFVARIDXATT = 436, IDXATT =
437,
COEFATT = 438, BASEMATRIXIDXATT = 439, TARGETMATRIXFIRSTROWATT = 440, TARGETMATRIXFIR-
STCOLATT = 441,
BASEMATRIXSTARTROWATT = 442, BASEMATRIXSTARTCOLATT = 443, BASEMATRIXENDROWATT =
444, BASEMATRIXENDCOLATT = 445,
SCALARMULTIPLIERATT = 446, EMPTYBASETRANSPOSEATT = 447, BASETRANSPOSEATT = 448, ELEM-
ENTSSTART = 449,
ELEMENTSEND = 450, CONSTANTELEMENTSSTART = 451, CONSTANTELEMENTSEND = 452, STARTVE-
CTORSTART = 453,
STARTVECTOREND = 454, NONZEROSSTART = 455, NONZEROSSEND = 456, INDEXESSTART = 457,
INDEXESEND = 458, VALUESSTART = 459, VALUESEND = 460, VARREFERENCEELEMENTSSTART = 461,
VARREFERENCEELEMENTSEND = 462, LINEARELEMENTSSTART = 463, LINEARELEMENTSEND = 464,
GENERALELEMENTSSTART = 465,
GENERALELEMENTSEND = 466, CONREFERENCEELEMENTSSTART = 467, CONREFERENCEELEMENT-
SEND = 468, VALUETYPEATT = 469,
OBJREFERENCEELEMENTSSTART = 470, OBJREFERENCEELEMENTSEND = 471, PATTERNELEMENTS-
START = 472, PATTERNELEMENTSEND = 473,
VARIDXSTART = 474, VARIDXEND = 475, TRANSFORMATIONSTART = 476, TRANSFORMATIONEND = 477,
COLOFFSETSSTART = 478, COLOFFSETSEND = 479, ROWOFFSETSSTART = 480, ROWOFFSETSEND =
481,
EMPTYROWMAJORATT = 482, ROWMAJORATT = 483, BLOCKROWIDXATT = 484, BLOCKCOLIDXATT =
485,
DUMMY = 486, NONLINEAREXPRESSIONSSTART = 487, NONLINEAREXPRESSIONSEND = 488, NUMBE-

ROFNONLINEAREXPRESSIONS = 489,
NLSTART = 490, NLEND = 491, MATRIXEXPRESSIONSSTART = 492, MATRIXEXPRESSIONSEND = 493,
NUMBEROFEXPR = 494, EXPRSTART = 495, EXPREND = 496, NUMBEROFMATRIXTERMSATT = 497,
MATRIXTERMSTART = 498, MATRIXTERMEND = 499, POWERSTART = 500, POWEREND = 501,
PLUSSTART = 502, PLUSEND = 503, MINUSSTART = 504, MINUSEND = 505,
DIVIDESTART = 506, DIVIDEEND = 507, LNSTART = 508, LLEND = 509,
SQRTSTART = 510, SQRTEND = 511, SUMSTART = 512, SUMEND = 513,
PRODUCTSTART = 514, PRODUCTEND = 515, EXPSTART = 516, EXPEND = 517,
NEGATESTART = 518, NEGATEEND = 519, IFSTART = 520, IFEND = 521,
SQUARESTART = 522, SQUAREEND = 523, COSSTART = 524, COSEND = 525,
SINSTART = 526, SINEND = 527, VARIABLESTART = 528, VARIABLEEND = 529,
ABSSTART = 530, ABSEND = 531, ERFSTART = 532, ERFEND = 533,
MAXSTART = 534, MAXEND = 535, ALLDIFFSTART = 536, ALLDIFFEND = 537,
MINSTART = 538, MINEND = 539, ESTART = 540, EEND = 541,
PISTART = 542, PIEND = 543, TIMESSTART = 544, TIMESEND = 545,
NUMBERSTART = 546, NUMBEREND = 547, MATRIXDETERMINANTSTART = 548, MATRIXDETERMINANT-
END = 549,
MATRIXTRACESTART = 550, MATRIXTRACEEND = 551, MATRIXTOSCALARSTART = 552, MATRIXTOSCA-
LAREND = 553,
MATRIXDIAGONALSTART = 554, MATRIXDIAGONALEND = 555, MATRIXDOTTIMESSTART = 556, MATRIX-
DOTTIMSEND = 557,
MATRIXLOWERTRIANGLESTART = 558, MATRIXLOWERTRIANGLEEND = 559, MATRIXUPPERTRIANGLE-
START = 560, MATRIXUPPERTRIANGLEEND = 561,
MATRIXMERGESTART = 562, MATRIXMERGEEND = 563, MATRIXMINUSSTART = 564, MATRIXMINUSEND
= 565,
MATRIXNEGATESTART = 566, MATRIXNEGATEEND = 567, MATRIXPLUSSTART = 568, MATRIXPLUSEND
= 569,
MATRIXTIMESSTART = 570, MATRIXTIMSEND = 571, MATRIXPRODUCTSTART = 572, MATRIXPRODUC-
TEND = 573,
MATRIXSCALARTIMESSTART = 574, MATRIXSCALARTIMSEND = 575, MATRIXSUBMATRIXATSTART =
576, MATRIXSUBMATRIXATEND = 577,
MATRIXTRANPOSESTART = 578, MATRIXTRANPOSEEND = 579, MATRIXREFERENCESTART = 580, M-
ATRIXREFERENCEEND = 581,
IDENTITYMATRIXSTART = 582, IDENTITYMATRIXEND = 583, MATRIXINVERSESTART = 584, MATRIXINV-
ERSEEND = 585,
EMPTYINCLUDEDIAGONALATT = 586, INCLUDEDIAGONALATT = 587, IDATT = 588, ATTRIBUTETEXT =
258,
ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
OSOLSTART = 267, OSOLSTARTEMPT = 268, OSOLATTRIBUTETEXT = 269, OSOLEND = 270,
NUMBEROFOTHEROPTIONSATT = 271, NUMBEROFENUMERATIONSATT = 272, NUMBEROFJOBIDSATT
= 273, NUMBEROFPATHSATT = 274,
NUMBEROFPATHPAIRSATT = 275, FROMATT = 276, TOATT = 277, MAKECOPYATT = 278,
CATEGORYATT = 279, TYPEATT = 280, GROUPWEIGHTATT = 281, NUMBEROFPROCESSESATT = 282,
NUMBEROF SOLVEROPTIONSATT = 283, NUMBEROF SOSATT = 284, NUMBEROFVARIABLESATT = 285,
NUMBEROF OBJECTIVESATT = 286,
NUMBEROF CONSTRAINTSATT = 287, NUMBEROF OTHERVARIABLEOPTIONSATT = 288, NUMBEROFOT-
HEROBJECTIVEOPTIONSATT = 289, NUMBEROF OTHERCONSTRAINTOPTIONSATT = 290,
NUMBEROF ITEMSATT = 291, NUMBEROFVARATT = 292, NUMBEROF OBJATT = 293, NUMBEROF CONATT

```

= 294,
NUMBEROFELATT = 295, NAMEATT = 296, IDXATT = 297, SOSIDXATT = 298,
VALUEATT = 299, UNITATT = 300, DESCRIPTIONATT = 301, CONTYPEATT = 302,
EMPTYCONTYPEATT = 303, ENUMTYPEATT = 304, EMPTYENUMTYPEATT = 305, OBJTYPEATT = 306,
EMPTYOBJTYPEATT = 307, VARTYPEATT = 308, EMPTYVARTYPEATT = 309, EMPTYTYPEATT = 310,
EMPTYNAMEATT = 311, EMPTYCATEGORYATT = 312, EMPTYDESCRIPTIONATT = 313, EMPTYUNITATT
= 314,
EMPTYVALUEATT = 315, EMPTYLBVALUEATT = 316, EMPTYUBVALUEATT = 317, LBVALUEATT = 318,
UBVALUEATT = 319, EMPTYLBDUALVALUEATT = 320, EMPTYUBDUALVALUEATT = 321, LBDUALVALUE-
ATT = 322,
UBDUALVALUEATT = 323, SOLVERATT = 324, EMPTYSOLVERATT = 325, WEIGHTATT = 326,
EMPTYWEIGHTATT = 327, TRANSPORTTYPEATT = 328, LOCATIONTYPEATT = 329, GENERALSTART =
330,
GENERALEND = 331, SYSTEMSTART = 332, SYSTEMEND = 333, SERVICESTART = 334,
SERVICEEND = 335, JOBSTART = 336, JOBEND = 337, OPTIMIZATIONSTART = 338,
OPTIMIZATIONEND = 339, SERVICEURISTART = 340, SERVICEURIEND = 341, SERVICENAMESTART =
342,
SERVICENAMEEND = 343, INSTANCENAMESTART = 344, INSTANCENAMEEND = 345, INSTANCELOCATI-
ONSTART = 346,
INSTANCELOCATIONEND = 347, JOBIDSTART = 348, JOBIDEND = 349, SOLVERTOINVOKESTART = 350,
SOLVERTOINVOKEEND = 351, LICENSESTART = 352, LICENSEEND = 353, USERNAMESTART = 354,
USERNAMEEND = 355, PASSWORDSTART = 356, PASSWORDEND = 357, CONTACTSTART = 358,
CONTACTEND = 359, OTHEROPTIONSSTART = 360, OTHEROPTIONSSEND = 361, OTHERSTART = 362,
OTHEREND = 363, MINDISKSPACESTART = 364, MINDISKSPACEEND = 365, MINMEMORYSTART = 366,
MINMEMORYEND = 367, MINCPUSPEEDSTART = 368, MINCPUSPEEDEND = 369, MINCPUNUMBERSTA-
RT = 370,
MINCPUNUMBEREND = 371, SERVICYPESTART = 372, SERVICYPEEND = 373, MAXTIMESTART =
374,
MAXTIMEEND = 375, REQUESTEDSTARTTIMESTART = 376, REQUESTEDSTARTTIMEEND = 377, DEPEN-
DENCIESSTART = 378,
DEPENDENCIESEND = 379, REQUIREDDIRECTORIESSTART = 380, REQUIREDDIRECTORIESEND = 381,
REQUIREDFILESSTART = 382,
REQUIREDFILESEND = 383, PATHSTART = 384, PATHEND = 385, PATHPAIRSTART = 386,
PATHPAIREND = 387, DIRECTORIESTOMAKESTART = 388, DIRECTORIESTOMAKEEND = 389, FILESTO-
MAKESTART = 390,
FILESTOMAKEEND = 391, DIRECTORIESTODELETESTART = 392, DIRECTORIESTODELETEEND = 393, F-
ILESTODELETESTART = 394,
FILESTODELETEEND = 395, INPUTDIRECTORIESTOMOVESTART = 396, INPUTDIRECTORIESTOMOVEE-
ND = 397, INPUTFILESTOMOVESTART = 398,
INPUTFILESTOMOVEEND = 399, OUTPUTDIRECTORIESTOMOVESTART = 400, OUTPUTDIRECTORIEST-
OMOVEEND = 401, OUTPUTFILESTOMOVESTART = 402,
OUTPUTFILESTOMOVEEND = 403, PROCESSESTOKILLSTART = 404, PROCESSESTOKILLEND = 405, P-
ROCESSSTART = 406,
PROCESSEND = 407, VARIABLESSTART = 408, VARIABLESEND = 409, INITIALVARIABLEVALUESSTART
= 410,
INITIALVARIABLEVALUESSEND = 411, VARSTART = 412, VAREND = 413, INITIALVARIABLEVALUESSTRIN-
GSTART = 414,
INITIALVARIABLEVALUESSTRINGEND = 415, INITIALBASISSTATUSSTART = 416, INITIALBASISSTATUSE-
ND = 417, BASICSTART = 418,
BASICEND = 419, ATUPPERSTART = 420, ATUPPEREND = 421, ATLOWERSTART = 422,
ATLOWEREND = 423, ATEQUALITYSTART = 424, ATEQUALITYEND = 425, SUPERBASICSTART = 426,
SUPERBASICEND = 427, ISFREESTART = 428, ISFREEEND = 429, UNKNOWNSTART = 430,
UNKNOWNEND = 431, INTEGERVARIABLEBRANCHINGWEIGHTSSTART = 432, INTEGERVARIABLEBRAN-

```

CHINGWEIGHTSEND = 433, SOSVARIABLEBRANCHINGWEIGHTSSTART = 434,
 SOSVARIABLEBRANCHINGWEIGHTSEND = 435, SOSSTART = 436, SOSEND = 437, OBJECTIVESSTART =
 438,
 OBJECTIVESEND = 439, INITIALOBJECTIVEVALUESSTART = 440, INITIALOBJECTIVEVALUESEND = 441,
 OBJSTART = 442,
 OBJEND = 443, INITIALOBJECTIVEBOUNDSSTART = 444, INITIALOBJECTIVEBOUNDSEND = 445, CONST-
 RAINTSSTART = 446,
 CONSTRAINTSEND = 447, INITIALCONSTRAINTVALUESSTART = 448, INITIALCONSTRAINTVALUESEND
 = 449, CONSTART = 450,
 CONEND = 451, INITIALDUALVALUESSTART = 452, INITIALDUALVALUESEND = 453, SOLVEROPTIONSS-
 TART = 454,
 SOLVEROPTIONSEND = 455, SOLVEROPTIONSTART = 456, SOLVEROPTIONEND = 457, ENUMERATION-
 START = 458,
 ENUMERATIONEND = 459, ITEMEMPTY = 460, ITEMSTART = 461, ITEMEND = 462,
 ITEMSTARTANDEND = 463, BASE64START = 464, BASE64END = 465, INCRATT = 466,
 MULTATT = 467, SIZEOFATT = 468, ELSTART = 469, ELEND = 470,
 MATRIXVARSTART = 471, MATRIXVAREND = 472, MATRIXOBJSTART = 473, MATRIXOBJEND = 474,
 MATRIXCONSTART = 475, MATRIXCONEND = 476, HEADERSTART = 477, HEADEREND = 478,
 FILENAMESTART = 479, FILENAMEEND = 480, FILENAMEEMPTY = 481, FILENAMESTARTANDEND = 482,
 FILESOURCESTART = 483, FILESOURCEEND = 484, FILESOURCEEMPTY = 485, FILESOURCESTARTAN-
 DEND = 486,
 FILEDESCRIPTIONSTART = 487, FILEDESCRIPTIONEND = 488, FILEDESCRIPTIONEMPTY = 489, FILEDE-
 SCRIPTIONSTARTANDEND = 490,
 FILECREATORSTART = 491, FILECREATOREND = 492, FILECREATOREMPTY = 493, FILECREATORSTA-
 RTANDEND = 494,
 FILELICENCESTART = 495, FILELICENCEEND = 496, FILELICENCEEMPTY = 497, FILELICENCESTARTAN-
 DEND = 498,
 MATRIXSTART = 499, MATRIXEND = 500, BASEMATRIXEND = 501, BASEMATRIXSTART = 502,
 BLOCKSTART = 503, BLOCKEND = 504, BLOCKSSTART = 505, BLOCKSEND = 506,
 EMPTYSHAPEATT = 507, SHAPEATT = 508, EMPTYSYMMETRYATT = 509, SYMMETRYATT = 510,
 EMPTYNEGATIVEPATTERNATT = 511, NEGATIVEPATTERNATT = 512, CONSTANTATT = 513, NUMBERO-
 FBLOCKSATT = 514,
 NUMBEROFCOLUMNSATT = 515, NUMBEROFROWSATT = 516, NUMBEROFVALUESATT = 517, NUMBER-
 OFVARIDXATT = 518,
 COEFATT = 519, BASEMATRIXIDXATT = 520, TARGETMATRIXFIRSTROWATT = 521, TARGETMATRIXFIR-
 STCOLATT = 522,
 BASEMATRIXSTARTROWATT = 523, BASEMATRIXSTARTCOLATT = 524, BASEMATRIXENDROWATT =
 525, BASEMATRIXENDCOLATT = 526,
 SCALARMULTIPLIERATT = 527, EMPTYBASETRANSPPOSEATT = 528, BASETRANSPPOSEATT = 529, ELEM-
 ENTSSTART = 530,
 ELEMENTSEND = 531, CONSTANTELEMENTSSTART = 532, CONSTANTELEMENTSEND = 533, STARTVE-
 CTORSTART = 534,
 STARTVECTOREND = 535, NONZEROSSTART = 536, NONZEROSSEND = 537, INDEXESSTART = 538,
 INDEXESEND = 539, VALUESSTART = 540, VALUESEND = 541, VARREFERENCEELEMENTSSTART = 542,
 VARREFERENCEELEMENTSEND = 543, LINEARELEMENTSSTART = 544, LINEARELEMENTSEND = 545,
 GENERALELEMENTSSTART = 546,
 GENERALELEMENTSEND = 547, CONREFERENCEELEMENTSSTART = 548, CONREFERENCEELEMENT-
 SEND = 549, VALUETYPEATT = 550,
 OBJREFERENCEELEMENTSSTART = 551, OBJREFERENCEELEMENTSEND = 552, PATTERNELEMENTS-
 START = 553, PATTERNELEMENTSEND = 554,
 VARIDXSTART = 555, VARIDXEND = 556, TRANSFORMATIONSTART = 557, TRANSFORMATIONEND = 558,
 COLOFFSETSSTART = 559, COLOFFSETSEND = 560, ROWOFFSETSSTART = 561, ROWOFFSETSEND =
 562,
 EMPTYROWMAJORATT = 563, ROWMAJORATT = 564, BLOCKROWIDXATT = 565, BLOCKCOLIDXATT =

566,
 DUMMY = 567, NONLINEAREXPRESSIONSSTART = 568, NONLINEAREXPRESSIONSEND = 569, NUMBEROFNONLINEAREXPRESSIONS = 570,
 NLSTART = 571, NLEND = 572, MATRIXEXPRESSIONSSTART = 573, MATRIXEXPRESSIONSEND = 574,
 NUMBEROFEXPR = 575, EXPRSTART = 576, EXPREND = 577, NUMBEROFMATRIXTERMSATT = 578,
 MATRIXTERMSTART = 579, MATRIXTERMEND = 580, POWERSTART = 581, POWEREND = 582,
 PLUSSTART = 583, PLUSEND = 584, MINUSSTART = 585, MINUSEND = 586,
 DIVIDESTART = 587, DIVIDEEND = 588, LNSTART = 589, LNEEND = 590,
 SQRTSTART = 591, SQRTEND = 592, SUMSTART = 593, SUMEND = 594,
 PRODUCTSTART = 595, PRODUCTEND = 596, EXPSTART = 597, EXPEND = 598,
 NEGATESTART = 599, NEGATEEND = 600, IFSTART = 601, IFEND = 602,
 SQUARESTART = 603, SQUAREEND = 604, COSSTART = 605, COSEND = 606,
 SINSTART = 607, SINEND = 608, VARIABLESTART = 609, VARIABLEEND = 610,
 ABSSTART = 611, ABSEND = 612, ERFSTART = 613, ERFEND = 614,
 MAXSTART = 615, MAXEND = 616, ALLDIFFSTART = 617, ALLDIFFEND = 618,
 MINSTART = 619, MINEND = 620, ESTART = 621, EEND = 622,
 PISTART = 623, PIEND = 624, TIMESSTART = 625, TIMESEND = 626,
 NUMBERSTART = 627, NUMBEREND = 628, MATRIXDETERMINANTSTART = 629, MATRIXDETERMINANTEND = 630,
 MATRIXTRACESTART = 631, MATRIXTRACEEND = 632, MATRIXTOSCALARSTART = 633, MATRIXTOSCALAREND = 634,
 MATRIXDIAGONALSTART = 635, MATRIXDIAGONALEND = 636, MATRIXDOTTIMESSTART = 637, MATRIXDOTTIMESEND = 638,
 MATRIXLOWERTRIANGLESTART = 639, MATRIXLOWERTRIANGLEEND = 640, MATRIXUPPERTRIANGLESTART = 641, MATRIXUPPERTRIANGLEEND = 642,
 MATRIXMERGESTART = 643, MATRIXMERGEEND = 644, MATRIXMINUSSTART = 645, MATRIXMINUSEND = 646,
 MATRIXNEGATESTART = 647, MATRIXNEGATEEND = 648, MATRIXPLUSSTART = 649, MATRIXPLUSEND = 650,
 MATRIXTIMESSTART = 651, MATRIXTIMESEND = 652, MATRIXPRODUCTSTART = 653, MATRIXPRODUCTEND = 654,
 MATRIXSCALARTIMESSTART = 655, MATRIXSCALARTIMESEND = 656, MATRIXSUBMATRIXATSTART = 657, MATRIXSUBMATRIXATEND = 658,
 MATRIXTRANPOSESTART = 659, MATRIXTRANPOSEEND = 660, MATRIXREFERENCESTART = 661, MATRIXREFERENCEEND = 662,
 IDENTITYMATRIXSTART = 663, IDENTITYMATRIXEND = 664, MATRIXINVERSESTART = 665, MATRIXINVERSEEND = 666,
 EMPTYINCLUDEDIAGONALATT = 667, INCLUDEDIAGONALATT = 668, IDATT = 669, ATTRIBUTETEXT = 258,
 ELEMENTTEXT = 259, ITEMTEXT = 260, INTEGER = 261, DOUBLE = 262,
 QUOTE = 263, TWOQUOTES = 264, GREATERTHAN = 265, ENDOFELEMENT = 266,
 OSRLSTART = 267, OSRLSTARTEMPT = 268, OSRLATTRIBUTETEXT = 269, OSRLEND = 270,
 NUMBEROFCONATT = 271, NUMBEROFCONSTRAINTSATT = 272, NUMBEROFELATT = 273, NUMBEROFENUMERATIONSATT = 274,
 NUMBEROFIDXATT = 275, NUMBEROFITEMSATT = 276, NUMBEROFOBJATT = 277, NUMBEROFOBJECTIVESATT = 278,
 NUMBEROFOTHERCONSTRAINTRESULTSATT = 279, NUMBEROFOTHEROBJECTIVERESULTSATT = 280, NUMBEROFOTHERRESULTSATT = 281, NUMBEROFOTHERSOLUTIONRESULTSATT = 282,
 NUMBEROFOTHERVARIABLERESULTSATT = 283, NUMBEROFSOLUTIONSATT = 284, NUMBEROFSOLVEROUTPUTSATT = 285, NUMBEROFSUBSTATUSESATT = 286,
 NUMBEROFTIMESATT = 287, NUMBEROFVARATT = 288, NUMBEROFVARIABLESATT = 289, NUMBEROF-

VARIDXATT = 290,
 TARGETOBJECTIVEIDXATT = 291, IDXATT = 292, INCRATT = 293, MULTATT = 294,
 SIZEOFATT = 295, CATEGORYATT = 296, EMPTYCATEGORYATT = 297, DESCRIPTIONATT = 298,
 EMPTYDESCRIPTIONATT = 299, NAMEATT = 300, EMPTYNAMEATT = 301, TYPEATT = 302,
 EMPTYTYPEATT = 303, CONTYPEATT = 304, EMPTYCONTYPEATT = 305, ENUMTYPEATT = 306,
 EMPTYENUMTYPEATT = 307, OBJTYPEATT = 308, EMPTYOBJTYPEATT = 309, VARTYPEATT = 310,
 EMPTYVARTYPEATT = 311, UNITATT = 312, EMPTYUNITATT = 313, VALUEATT = 314,
 EMPTYVALUEATT = 315, WEIGHTEDOBJECTIVESATT = 316, EMPTYWEIGHTEDOBJECTIVESATT = 317,
 TARGETOBJECTIVENAMEATT = 318,
 EMPTYTARGETOBJECTIVENAMEATT = 319, HEADERSTART = 320, HEADEREND = 321, GENERALSTART
 = 322,
 GENERALEND = 323, SYSTEMSTART = 324, SYSTEMEND = 325, SERVICESTART = 326,
 SERVICEEND = 327, JOBSTART = 328, JOBEND = 329, OPTIMIZATIONSTART = 330,
 OPTIMIZATIONEND = 331, ITEMSTART = 332, ITEMEND = 333, ITEMSTARTANDEND = 334,
 ITEMEMPTY = 335, ACTUALSTARTTIMESTART = 336, ACTUALSTARTTIMEEND = 337, ATEQUALITYSTART
 = 338,
 ATEQUALITYEND = 339, ATLOWERSTART = 340, ATLOWEREND = 341, ATUPPERSTART = 342,
 ATUPPEREND = 343, AVAILABLECPUNUMBERSTART = 344, AVAILABLECPUNUMBEREND = 345, AVAILA-
 BLECPUSPEEDSTART = 346,
 AVAILABLECPUSPEEDEND = 347, AVAILABLEDISKSPACESTART = 348, AVAILABLEDISKSPACEEND =
 349, AVAILABLEMEMORYSTART = 350,
 AVAILABLEMEMORYEND = 351, BASE64START = 352, BASE64END = 353, BASICSTART = 354,
 BASICEND = 355, BASISSTATUSSTART = 356, BASISSTATUSEND = 357, BASSTATUSSTART = 358,
 BASSTATUSEND = 359, CONSTART = 360, CONEND = 361, CONSTRAINTSSTART = 362,
 CONSTRAINTSEND = 363, CURRENTJOBCOUNTSTART = 364, CURRENTJOBCOUNTEND = 365, CURRE-
 NTSTATESTART = 366,
 CURRENTSTATEEND = 367, DUALVALUESSTART = 368, DUALVALUESEND = 369, ELSTART = 370,
 EEND = 371, ENUMERATIONSTART = 372, ENUMERATIONEND = 373, ENDTIMESTART = 374,
 ENDTIMEEND = 375, GENERALSTATUSSTART = 376, GENERALSTATUSEND = 377, GENERALSUBSTAT-
 USSTART = 378,
 GENERALSUBSTATUSEND = 379, IDXSTART = 380, IDXEND = 381, INSTANCENAMESTART = 382,
 INSTANCENAMEEND = 383, ISFREESTART = 384, ISFREEEND = 385, JOBIDSTART = 386,
 JOBIDEND = 387, MESSAGESTART = 388, MESSAGEEND = 389, OBJSTART = 390,
 OBJEND = 391, OBJECTIVESSTART = 392, OBJECTIVESEND = 393, OPTIMIZATIONSOLUTIONSTATUSST-
 ART = 394,
 OPTIMIZATIONSOLUTIONSTATUSEND = 395, OPTIMIZATIONSOLUTIONSUBSTATUSSTART = 396, OPTI-
 MIZATIONSOLUTIONSUBSTATUSEND = 397, OTHERSTART = 398,
 OTHEREND = 399, OTHERRESULTSSTART = 400, OTHERRESULTSEND = 401, OTHERSOLUTIONRESUL-
 TSTART = 402,
 OTHERSOLUTIONRESULTEND = 403, OTHERSOLUTIONRESULTSSTART = 404, OTHERSOLUTIONRESUL-
 TSEND = 405, OTHERSOLVEROUTPUTSTART = 406,
 OTHERSOLVEROUTPUTEND = 407, SCHEDULEDSTARTTIMESTART = 408, SCHEDULEDSTARTTIMEEND
 = 409, SERVICENAMESTART = 410,
 SERVICENAMEEND = 411, SERVICEURISTART = 412, SERVICEURIEND = 413, SERVICEUTILIZATIONST-
 ART = 414,
 SERVICEUTILIZATIONEND = 415, SOLUTIONSTART = 416, SOLUTIONEND = 417, SOLVERINVOKEDSTA-
 RT = 418,
 SOLVERINVOKEDEND = 419, SOLVEROUTPUTSTART = 420, SOLVEROUTPUTEND = 421, STATUSSTART
 = 422,
 STATUSEND = 423, SUBMITTIMESTART = 424, SUBMITTIMEEND = 425, SUBSTATUSSTART = 426,
 SUBSTATUSEND = 427, SUPERBASICSTART = 428, SUPERBASICEND = 429, SYSTEMINFORMATIONST-
 ART = 430,
 SYSTEMINFORMATIONEND = 431, TIMESTART = 432, TIMEEND = 433, TIMESERVICESTARTEDSTART =

434,
TIMESERVICESTARTEDEND = 435, TIMESTAMPSTART = 436, TIMESTAMPEND = 437, TIMINGINFORMATIONSTART = 438,
TIMINGINFORMATIONEND = 439, TOTALJOBSSOFARSTART = 440, TOTALJOBSSOFAREND = 441, UNKNOWNSTART = 442,
UNKNOWNEND = 443, USEDCPUNUMBERSTART = 444, USEDCPUNUMBEREND = 445, USEDCPUSPEEDSTART = 446,
USEDCPUSPEEDEND = 447, USEDDISKSPACESTART = 448, USEDDISKSPACEEND = 449, USEDMEMORYSTART = 450,
USEDMEMORYEND = 451, VALUESSTRINGSTART = 452, VALUESSTRINGEND = 453, VARSTART = 454, VAREND = 455, VARIABLESSTART = 456, VARIABLESEND = 457, VARIDXSTART = 458,
VARIDXEND = 459, MATRIXVARSTART = 460, MATRIXVAREND = 461, MATRIXOBJSTART = 462, MATRIXOBJEND = 463, MATRIXCONSTART = 464, MATRIXCONEND = 465, FILENAMESTART = 466,
FILENAMEEND = 467, FILENAMEEMPTY = 468, FILENAMESTARTANDEND = 469, FILESOURCESTART = 470,
FILESOURCEEND = 471, FILESOURCEEMPTY = 472, FILESOURCESTARTANDEND = 473, FILEDESCRIPTIONSTART = 474,
FILEDESCRIPTIONEND = 475, FILEDESCRIPTIONEMPTY = 476, FILEDESCRIPTIONSTARTANDEND = 477, FILECREATORSTART = 478,
FILECREATOREND = 479, FILECREATOREMPTY = 480, FILECREATORSTARTANDEND = 481, FILELICENCESTART = 482,
FILELICENCEEND = 483, FILELICENCEEMPTY = 484, FILELICENCESTARTANDEND = 485, MATRIXSTART = 486,
MATRIXEND = 487, BASEMATRIXEND = 488, BASEMATRIXSTART = 489, BLOCKSTART = 490, BLOCKEND = 491, BLOCKSSTART = 492, BLOCKSEND = 493, EMPTYSHAPEATT = 494,
SHAPEATT = 495, EMPTYSYMMETRYATT = 496, SYMMETRYATT = 497, EMPTYNEGATIVEPATTERNATT = 498,
NEGATIVEPATTERNATT = 499, CONSTANTATT = 500, NUMBEROFBLOCKSATT = 501, NUMBEROFCOLUMNSSATT = 502,
NUMBEROFFROWSATT = 503, NUMBEROFVALUESATT = 504, COEFATT = 505, BASEMATRIXIDXATT = 506, TARGETMATRIXFIRSTROWATT = 507, TARGETMATRIXFIRSTCOLATT = 508, BASEMATRIXSTARTROWATT = 509, BASEMATRIXSTARTCOLATT = 510,
BASEMATRIXENDROWATT = 511, BASEMATRIXENDCOLATT = 512, SCALARMULTIPLIERATT = 513, EMPTYBASETRANPOSEATT = 514,
BASETRANPOSEATT = 515, ELEMENTSSTART = 516, ELEMENTSEND = 517, CONSTANTELEMENTSSTART = 518,
CONSTANTELEMENTSEND = 519, STARTVECTORSTART = 520, STARTVECTOREND = 521, NONZEROSTART = 522,
NONZEROSSEND = 523, INDEXESSTART = 524, INDEXESEND = 525, VALUESSTART = 526, VALUESEND = 527, VARREFERENCEELEMENTSSTART = 528, VARREFERENCEELEMENTSEND = 529, LINEARELEMENTSSTART = 530,
LINEARELEMENTSEND = 531, GENERALELEMENTSSTART = 532, GENERALELEMENTSEND = 533, CONREFERENCEELEMENTSSTART = 534,
CONREFERENCEELEMENTSEND = 535, VALUETYPEATT = 536, OBJREFERENCEELEMENTSSTART = 537, OBJREFERENCEELEMENTSEND = 538,
PATTERNELEMENTSSTART = 539, PATTERNELEMENTSEND = 540, TRANSFORMATIONSTART = 541, TRANSFORMATIONEND = 542,
COLOFFSETSSTART = 543, COLOFFSETSEND = 544, ROWOFFSETSSTART = 545, ROWOFFSETSEND = 546,
EMPTYROWMAJORATT = 547, ROWMAJORATT = 548, BLOCKROWIDXATT = 549, BLOCKCOLIDXATT = 550,
DUMMY = 551, NONLINEAREXPRESSIONSSTART = 552, NONLINEAREXPRESSIONSEND = 553, NUMBE-


```

ROFNONLINEAREXPRESSIONS = 554,
NLSTART = 555, NLEND = 556, MATRIXEXPRESSIONSSTART = 557, MATRIXEXPRESSIONSEND = 558,
NUMBEROFEXPR = 559, EXPRSTART = 560, EXPREND = 561, NUMBEROFMATRIXTERMSATT = 562,
MATRIXTERMSTART = 563, MATRIXTERMEND = 564, POWERSTART = 565, POWEREND = 566,
PLUSSTART = 567, PLUSEND = 568, MINUSSTART = 569, MINUSEND = 570,
DIVIDESTART = 571, DIVIDEEND = 572, LNSTART = 573, LNEEND = 574,
SQRTSTART = 575, SQRTEND = 576, SUMSTART = 577, SUMEND = 578,
PRODUCTSTART = 579, PRODUCTEND = 580, EXPSTART = 581, EXPEND = 582,
NEGATESTART = 583, NEGATEEND = 584, IFSTART = 585, IFEND = 586,
SQUARESTART = 587, SQUAREEND = 588, COSSTART = 589, COSEND = 590,
SINSTART = 591, SINEND = 592, VARIABLESTART = 593, VARIABLEEND = 594,
ABSSTART = 595, ABSEND = 596, ERFSTART = 597, ERFEND = 598,
MAXSTART = 599, MAXEND = 600, ALLDIFFSTART = 601, ALLDIFFEND = 602,
MINSTART = 603, MINEND = 604, ESTART = 605, EEND = 606,
PISTART = 607, PIEND = 608, TIMESSTART = 609, TIMESEND = 610,
NUMBERSTART = 611, NUMBEREND = 612, MATRIXDETERMINANTSTART = 613, MATRIXDETERMINANT-
END = 614,
MATRIXTRACESTART = 615, MATRIXTRACEEND = 616, MATRIXTOSCALARSTART = 617, MATRIXTOSCA-
LAREND = 618,
MATRIXDIAGONALSTART = 619, MATRIXDIAGONALEND = 620, MATRIXDOTTIMESSTART = 621, MATRIX-
DOTTIMESEND = 622,
MATRIXLOWERTRIANGLESTART = 623, MATRIXLOWERTRIANGLEEND = 624, MATRIXUPPERTRIANGLE-
START = 625, MATRIXUPPERTRIANGLEEND = 626,
MATRIXMERGESTART = 627, MATRIXMERGEEND = 628, MATRIXMINUSSTART = 629, MATRIXMINUSEND
= 630,
MATRIXNEGATESTART = 631, MATRIXNEGATEEND = 632, MATRIXPLUSSTART = 633, MATRIXPLUSEND
= 634,
MATRIXTIMESSTART = 635, MATRIXTIMESEND = 636, MATRIXPRODUCTSTART = 637, MATRIXPRODUC-
TEND = 638,
MATRIXSCALARTIMESSTART = 639, MATRIXSCALARTIMESEND = 640, MATRIXSUBMATRIXATSTART =
641, MATRIXSUBMATRIXATEND = 642,
MATRIXTRANPOSESTART = 643, MATRIXTRANPOSEEND = 644, MATRIXREFERENCESTART = 645, M-
ATRIXREFERENCEEND = 646,
IDENTITYMATRIXSTART = 647, IDENTITYMATRIXEND = 648, MATRIXINVERSESTART = 649, MATRIXINV-
ERSEEND = 650,
EMPTYINCLUDEDIAGONALATT = 651, INCLUDEDIAGONALATT = 652, IDATT = 653 }

```

7.37.1 Macro Definition Documentation

7.37.1.1 #define ATTRIBUTETEXT 258

Definition at line 441 of file OSParseosrl.tab.hpp.

7.37.1.2 #define ELEMENTTEXT 259

Definition at line 442 of file OSParseosrl.tab.hpp.

7.37.1.3 #define ITEMTEXT 260

Definition at line 443 of file OSParseosrl.tab.hpp.

7.37.1.4 #define INTEGER 261

Definition at line 444 of file OSParseosrl.tab.hpp.

7.37.1.5 #define DOUBLE 262

Definition at line 445 of file OSParseosrl.tab.hpp.

7.37.1.6 #define QUOTE 263

Definition at line 446 of file OSParseosrl.tab.hpp.

7.37.1.7 #define TWOQUOTES 264

Definition at line 447 of file OSParseosrl.tab.hpp.

7.37.1.8 #define GREATERTHAN 265

Definition at line 448 of file OSParseosrl.tab.hpp.

7.37.1.9 #define ENDOFELEMENT 266

Definition at line 449 of file OSParseosrl.tab.hpp.

7.37.1.10 #define OSRLSTART 267

Definition at line 450 of file OSParseosrl.tab.hpp.

7.37.1.11 #define OSRLSTARTEMPT 268

Definition at line 451 of file OSParseosrl.tab.hpp.

7.37.1.12 #define OSRLATTRIBUTETEXT 269

Definition at line 452 of file OSParseosrl.tab.hpp.

7.37.1.13 #define OSRLEND 270

Definition at line 453 of file OSParseosrl.tab.hpp.

7.37.1.14 #define NUMBEROFCONATT 271

Definition at line 454 of file OSParseosrl.tab.hpp.

7.37.1.15 #define NUMBEROFCONSTRAINTSATT 272

Definition at line 455 of file OSParseosrl.tab.hpp.

7.37.1.16 #define NUMBEROFELATT 273

Definition at line 456 of file OSParseosrl.tab.hpp.

7.37.1.17 #define NUMBEROFENUMERATIONSATT 274

Definition at line 457 of file OSParseosrl.tab.hpp.

7.37.1.18 #define NUMBEROFIDXATT 275

Definition at line 458 of file OSParseosrl.tab.hpp.

7.37.1.19 #define NUMBEROFITEMSATT 276

Definition at line 459 of file OSParseosrl.tab.hpp.

7.37.1.20 #define NUMBEROFOBJATT 277

Definition at line 460 of file OSParseosrl.tab.hpp.

7.37.1.21 #define NUMBEROFOBJECTIVESATT 278

Definition at line 461 of file OSParseosrl.tab.hpp.

7.37.1.22 #define NUMBEROFOTHERCONSTRAINTRESULTSATT 279

Definition at line 462 of file OSParseosrl.tab.hpp.

7.37.1.23 #define NUMBEROFOTHEROBJECTIVERESULTSATT 280

Definition at line 463 of file OSParseosrl.tab.hpp.

7.37.1.24 #define NUMBEROFOTHERRESULTSATT 281

Definition at line 464 of file OSParseosrl.tab.hpp.

7.37.1.25 #define NUMBEROFOTHERSOLUTIONRESULTSATT 282

Definition at line 465 of file OSParseosrl.tab.hpp.

7.37.1.26 #define NUMBEROFOTHERVARIABLERESULTSATT 283

Definition at line 466 of file OSParseosrl.tab.hpp.

7.37.1.27 #define NUMBEROFSOLUTIONSATT 284

Definition at line 467 of file OSParseosrl.tab.hpp.

7.37.1.28 #define NUMBEROFSOLVEROUTPUTSATT 285

Definition at line 468 of file OSParseosrl.tab.hpp.

7.37.1.29 #define NUMBEROFSUBSTATUSESATT 286

Definition at line 469 of file OSParseosrl.tab.hpp.

7.37.1.30 #define NUMBEROFTIMESATT 287

Definition at line 470 of file OSParseosrl.tab.hpp.

7.37.1.31 #define NUMBEROFVARATT 288

Definition at line 471 of file OSParseosrl.tab.hpp.

7.37.1.32 #define NUMBEROFVARIABLESATT 289

Definition at line 472 of file OSParseosrl.tab.hpp.

7.37.1.33 #define NUMBEROFVARIDXATT 290

Definition at line 473 of file OSParseosrl.tab.hpp.

7.37.1.34 #define TARGETOBJECTIVEIDXATT 291

Definition at line 474 of file OSParseosrl.tab.hpp.

7.37.1.35 #define IDXATT 292

Definition at line 475 of file OSParseosrl.tab.hpp.

7.37.1.36 #define INCRATT 293

Definition at line 476 of file OSParseosrl.tab.hpp.

7.37.1.37 #define MULTATT 294

Definition at line 477 of file OSParseosrl.tab.hpp.

7.37.1.38 #define SIZEOFATT 295

Definition at line 478 of file OSParseosrl.tab.hpp.

7.37.1.39 #define CATEGORYATT 296

Definition at line 479 of file OSParseosrl.tab.hpp.

7.37.1.40 #define EMPTYCATEGORYATT 297

Definition at line 480 of file OSParseosrl.tab.hpp.

7.37.1.41 #define DESCRIPTIONATT 298

Definition at line 481 of file OSParseosrl.tab.hpp.

7.37.1.42 #define EMPTYDESCRIPTIONATT 299

Definition at line 482 of file OSParseosrl.tab.hpp.

7.37.1.43 #define NAMEATT 300

Definition at line 483 of file OSParseosrl.tab.hpp.

7.37.1.44 #define EMPTYNAMEATT 301

Definition at line 484 of file OSParseosrl.tab.hpp.

7.37.1.45 #define TYPEATT 302

Definition at line 485 of file OSParseosrl.tab.hpp.

7.37.1.46 #define EMPTYTYPEATT 303

Definition at line 486 of file OSParseosrl.tab.hpp.

7.37.1.47 #define CONTYPEATT 304

Definition at line 487 of file OSParseosrl.tab.hpp.

7.37.1.48 #define EMPTYCONTYPEATT 305

Definition at line 488 of file OSParseosrl.tab.hpp.

7.37.1.49 #define ENUMTYPEATT 306

Definition at line 489 of file OSParseosrl.tab.hpp.

7.37.1.50 #define EMPTYENUMTYPEATT 307

Definition at line 490 of file OSParseosrl.tab.hpp.

7.37.1.51 #define OBJTYPEATT 308

Definition at line 491 of file OSParseosrl.tab.hpp.

7.37.1.52 #define EMPTYOBJTYPEATT 309

Definition at line 492 of file OSParseosrl.tab.hpp.

7.37.1.53 #define VARTYPEATT 310

Definition at line 493 of file OSParseosrl.tab.hpp.

7.37.1.54 #define EMPTYVARTYPEATT 311

Definition at line 494 of file OSParseosrl.tab.hpp.

7.37.1.55 #define UNITATT 312

Definition at line 495 of file OSParseosrl.tab.hpp.

7.37.1.56 #define EMPTYUNITATT 313

Definition at line 496 of file OSParseosrl.tab.hpp.

7.37.1.57 #define VALUEATT 314

Definition at line 497 of file OSParseosrl.tab.hpp.

7.37.1.58 #define EMPTYVALUEATT 315

Definition at line 498 of file OSParseosrl.tab.hpp.

7.37.1.59 #define WEIGHTEDOBJECTIVESATT 316

Definition at line 499 of file OSParseosrl.tab.hpp.

7.37.1.60 #define EMPTYWEIGHTEDOBJECTIVESATT 317

Definition at line 500 of file OSParseosrl.tab.hpp.

7.37.1.61 #define TARGETOBJECTIVENAMEATT 318

Definition at line 501 of file OSParseosrl.tab.hpp.

7.37.1.62 #define EMPTYTARGETOBJECTIVENAMEATT 319

Definition at line 502 of file OSParseosrl.tab.hpp.

7.37.1.63 #define HEADERSTART 320

Definition at line 503 of file OSParseosrl.tab.hpp.

7.37.1.64 #define HEADEREND 321

Definition at line 504 of file OSParseosrl.tab.hpp.

7.37.1.65 #define GENERALSTART 322

Definition at line 505 of file OSParseosrl.tab.hpp.

7.37.1.66 #define GENERALEND 323

Definition at line 506 of file OSParseosrl.tab.hpp.

7.37.1.67 #define SYSTEMSTART 324

Definition at line 507 of file OSParseosrl.tab.hpp.

7.37.1.68 #define SYSTEMEND 325

Definition at line 508 of file OSParseosrl.tab.hpp.

7.37.1.69 #define SERVICESTART 326

Definition at line 509 of file OSParseosrl.tab.hpp.

7.37.1.70 #define SERVICEEND 327

Definition at line 510 of file OSParseosrl.tab.hpp.

7.37.1.71 #define JOBSTART 328

Definition at line 511 of file OSParseosrl.tab.hpp.

7.37.1.72 #define JOBEND 329

Definition at line 512 of file OSParseosrl.tab.hpp.

7.37.1.73 #define OPTIMIZATIONSTART 330

Definition at line 513 of file OSParseosrl.tab.hpp.

7.37.1.74 #define OPTIMIZATIONEND 331

Definition at line 514 of file OSParseosrl.tab.hpp.

7.37.1.75 #define ITEMSTART 332

Definition at line 515 of file OSParseosrl.tab.hpp.

7.37.1.76 #define ITEMEND 333

Definition at line 516 of file OSParseosrl.tab.hpp.

7.37.1.77 #define ITEMSTARTANDEND 334

Definition at line 517 of file OSParseosrl.tab.hpp.

7.37.1.78 #define ITEMEMPTY 335

Definition at line 518 of file OSParseosrl.tab.hpp.

7.37.1.79 #define ACTUALSTARTTIMESTART 336

Definition at line 519 of file OSParseosrl.tab.hpp.

7.37.1.80 #define ACTUALSTARTTIMEEND 337

Definition at line 520 of file OSParseosrl.tab.hpp.

7.37.1.81 #define ATEQUALITYSTART 338

Definition at line 521 of file OSParseosrl.tab.hpp.

7.37.1.82 #define ATEQUALITYEND 339

Definition at line 522 of file OSParseosrl.tab.hpp.

7.37.1.83 #define ATLOWERSTART 340

Definition at line 523 of file OSParseosrl.tab.hpp.

7.37.1.84 #define ATLOWEREND 341

Definition at line 524 of file OSParseosrl.tab.hpp.

7.37.1.85 #define ATUPPERSTART 342

Definition at line 525 of file OSParseosrl.tab.hpp.

7.37.1.86 #define ATUPPEREND 343

Definition at line 526 of file OSParseosrl.tab.hpp.

7.37.1.87 #define AVAILABLECPUNUMBERSTART 344

Definition at line 527 of file OSParseosrl.tab.hpp.

7.37.1.88 #define AVAILABLECPUNUMBEREND 345

Definition at line 528 of file OSParseosrl.tab.hpp.

7.37.1.89 **#define AVAILABLECPUSPEEDSTART 346**

Definition at line 529 of file OSParseosrl.tab.hpp.

7.37.1.90 **#define AVAILABLECPUSPEEDEND 347**

Definition at line 530 of file OSParseosrl.tab.hpp.

7.37.1.91 **#define AVAILABLEDISKSPACESTART 348**

Definition at line 531 of file OSParseosrl.tab.hpp.

7.37.1.92 **#define AVAILABLEDISKSPACEEND 349**

Definition at line 532 of file OSParseosrl.tab.hpp.

7.37.1.93 **#define AVAILABLEMEMORYSTART 350**

Definition at line 533 of file OSParseosrl.tab.hpp.

7.37.1.94 **#define AVAILABLEMEMORYEND 351**

Definition at line 534 of file OSParseosrl.tab.hpp.

7.37.1.95 **#define BASE64START 352**

Definition at line 535 of file OSParseosrl.tab.hpp.

7.37.1.96 **#define BASE64END 353**

Definition at line 536 of file OSParseosrl.tab.hpp.

7.37.1.97 **#define BASICSTART 354**

Definition at line 537 of file OSParseosrl.tab.hpp.

7.37.1.98 **#define BASICEND 355**

Definition at line 538 of file OSParseosrl.tab.hpp.

7.37.1.99 **#define BASISSTATUSSTART 356**

Definition at line 539 of file OSParseosrl.tab.hpp.

7.37.1.100 **#define BASISSTATUSEND 357**

Definition at line 540 of file OSParseosrl.tab.hpp.

7.37.1.101 **#define BASSTATUSSTART 358**

Definition at line 541 of file OSParseosrl.tab.hpp.

7.37.1.102 **#define BASSTATUSEND 359**

Definition at line 542 of file OSParseosrl.tab.hpp.

7.37.1.103 #define CONSTART 360

Definition at line 543 of file OSParseosrl.tab.hpp.

7.37.1.104 #define CONEND 361

Definition at line 544 of file OSParseosrl.tab.hpp.

7.37.1.105 #define CONSTRAINTSSTART 362

Definition at line 545 of file OSParseosrl.tab.hpp.

7.37.1.106 #define CONSTRAINTSEND 363

Definition at line 546 of file OSParseosrl.tab.hpp.

7.37.1.107 #define CURRENTJOBCOUNTSTART 364

Definition at line 547 of file OSParseosrl.tab.hpp.

7.37.1.108 #define CURRENTJOBCOUNTEND 365

Definition at line 548 of file OSParseosrl.tab.hpp.

7.37.1.109 #define CURRENTSTATESTART 366

Definition at line 549 of file OSParseosrl.tab.hpp.

7.37.1.110 #define CURRENTSTATEEND 367

Definition at line 550 of file OSParseosrl.tab.hpp.

7.37.1.111 #define DUALVALUESSTART 368

Definition at line 551 of file OSParseosrl.tab.hpp.

7.37.1.112 #define DUALVALUESEND 369

Definition at line 552 of file OSParseosrl.tab.hpp.

7.37.1.113 #define ELSTART 370

Definition at line 553 of file OSParseosrl.tab.hpp.

7.37.1.114 #define ELEND 371

Definition at line 554 of file OSParseosrl.tab.hpp.

7.37.1.115 #define ENUMERATIONSTART 372

Definition at line 555 of file OSParseosrl.tab.hpp.

7.37.1.116 #define ENUMERATIONEND 373

Definition at line 556 of file OSParseosrl.tab.hpp.

7.37.1.117 #define ENDTIMESTART 374

Definition at line 557 of file OSParseosrl.tab.hpp.

7.37.1.118 #define ENDTIMEEND 375

Definition at line 558 of file OSParseosrl.tab.hpp.

7.37.1.119 #define GENERALSTATUSSTART 376

Definition at line 559 of file OSParseosrl.tab.hpp.

7.37.1.120 #define GENERALSTATUSEND 377

Definition at line 560 of file OSParseosrl.tab.hpp.

7.37.1.121 #define GENERALSUBSTATUSSTART 378

Definition at line 561 of file OSParseosrl.tab.hpp.

7.37.1.122 #define GENERALSUBSTATUSEND 379

Definition at line 562 of file OSParseosrl.tab.hpp.

7.37.1.123 #define IDXSTART 380

Definition at line 563 of file OSParseosrl.tab.hpp.

7.37.1.124 #define IDXEND 381

Definition at line 564 of file OSParseosrl.tab.hpp.

7.37.1.125 #define INSTANCENAMESTART 382

Definition at line 565 of file OSParseosrl.tab.hpp.

7.37.1.126 #define INSTANCENAMEEND 383

Definition at line 566 of file OSParseosrl.tab.hpp.

7.37.1.127 #define ISFREESTART 384

Definition at line 567 of file OSParseosrl.tab.hpp.

7.37.1.128 #define ISFREEEND 385

Definition at line 568 of file OSParseosrl.tab.hpp.

7.37.1.129 #define JOBIDSTART 386

Definition at line 569 of file OSParseosrl.tab.hpp.

7.37.1.130 #define JOBIDEND 387

Definition at line 570 of file OSParseosrl.tab.hpp.

7.37.1.131 **#define MESSAGESTART 388**

Definition at line 571 of file OSParseosrl.tab.hpp.

7.37.1.132 **#define MESSAGEEND 389**

Definition at line 572 of file OSParseosrl.tab.hpp.

7.37.1.133 **#define OBJSTART 390**

Definition at line 573 of file OSParseosrl.tab.hpp.

7.37.1.134 **#define OBJEND 391**

Definition at line 574 of file OSParseosrl.tab.hpp.

7.37.1.135 **#define OBJECTIVESSTART 392**

Definition at line 575 of file OSParseosrl.tab.hpp.

7.37.1.136 **#define OBJECTIVESEND 393**

Definition at line 576 of file OSParseosrl.tab.hpp.

7.37.1.137 **#define OPTIMIZATIONSOLUTIONSTATUSSTART 394**

Definition at line 577 of file OSParseosrl.tab.hpp.

7.37.1.138 **#define OPTIMIZATIONSOLUTIONSTATUSEND 395**

Definition at line 578 of file OSParseosrl.tab.hpp.

7.37.1.139 **#define OPTIMIZATIONSOLUTIONSUBSTATUSSTART 396**

Definition at line 579 of file OSParseosrl.tab.hpp.

7.37.1.140 **#define OPTIMIZATIONSOLUTIONSUBSTATUSEND 397**

Definition at line 580 of file OSParseosrl.tab.hpp.

7.37.1.141 **#define OTHERSTART 398**

Definition at line 581 of file OSParseosrl.tab.hpp.

7.37.1.142 **#define OTHEREND 399**

Definition at line 582 of file OSParseosrl.tab.hpp.

7.37.1.143 **#define OTHERRESULTSSTART 400**

Definition at line 583 of file OSParseosrl.tab.hpp.

7.37.1.144 **#define OTHERRESULTSEND 401**

Definition at line 584 of file OSParseosrl.tab.hpp.

7.37.1.145 **#define OTHERSOLUTIONRESULTSTART 402**

Definition at line 585 of file OSParseosrl.tab.hpp.

7.37.1.146 **#define OTHERSOLUTIONRESULTEND 403**

Definition at line 586 of file OSParseosrl.tab.hpp.

7.37.1.147 **#define OTHERSOLUTIONRESULTSSTART 404**

Definition at line 587 of file OSParseosrl.tab.hpp.

7.37.1.148 **#define OTHERSOLUTIONRESULTSEND 405**

Definition at line 588 of file OSParseosrl.tab.hpp.

7.37.1.149 **#define OTHERSOLVEROUTPUTSTART 406**

Definition at line 589 of file OSParseosrl.tab.hpp.

7.37.1.150 **#define OTHERSOLVEROUTPUTEND 407**

Definition at line 590 of file OSParseosrl.tab.hpp.

7.37.1.151 **#define SCHEDULEDSTARTTIMESTART 408**

Definition at line 591 of file OSParseosrl.tab.hpp.

7.37.1.152 **#define SCHEDULEDSTARTTIMEEND 409**

Definition at line 592 of file OSParseosrl.tab.hpp.

7.37.1.153 **#define SERVICENAMESTART 410**

Definition at line 593 of file OSParseosrl.tab.hpp.

7.37.1.154 **#define SERVICENAMEEND 411**

Definition at line 594 of file OSParseosrl.tab.hpp.

7.37.1.155 **#define SERVICEURISTART 412**

Definition at line 595 of file OSParseosrl.tab.hpp.

7.37.1.156 **#define SERVICEURIEND 413**

Definition at line 596 of file OSParseosrl.tab.hpp.

7.37.1.157 **#define SERVICEUTILIZATIONSTART 414**

Definition at line 597 of file OSParseosrl.tab.hpp.

7.37.1.158 **#define SERVICEUTILIZATIONEND 415**

Definition at line 598 of file OSParseosrl.tab.hpp.

7.37.1.159 **#define SOLUTIONSTART 416**

Definition at line 599 of file OSParseosrl.tab.hpp.

7.37.1.160 **#define SOLUTIONEND 417**

Definition at line 600 of file OSParseosrl.tab.hpp.

7.37.1.161 **#define SOLVERINVOKEDSTART 418**

Definition at line 601 of file OSParseosrl.tab.hpp.

7.37.1.162 **#define SOLVERINVOKEDEND 419**

Definition at line 602 of file OSParseosrl.tab.hpp.

7.37.1.163 **#define SOLVEROUTPUTSTART 420**

Definition at line 603 of file OSParseosrl.tab.hpp.

7.37.1.164 **#define SOLVEROUTPUTEND 421**

Definition at line 604 of file OSParseosrl.tab.hpp.

7.37.1.165 **#define STATUSSTART 422**

Definition at line 605 of file OSParseosrl.tab.hpp.

7.37.1.166 **#define STATUSEND 423**

Definition at line 606 of file OSParseosrl.tab.hpp.

7.37.1.167 **#define SUBMITTIMESTART 424**

Definition at line 607 of file OSParseosrl.tab.hpp.

7.37.1.168 **#define SUBMITTIMEEND 425**

Definition at line 608 of file OSParseosrl.tab.hpp.

7.37.1.169 **#define SUBSTATUSSTART 426**

Definition at line 609 of file OSParseosrl.tab.hpp.

7.37.1.170 **#define SUBSTATUSEND 427**

Definition at line 610 of file OSParseosrl.tab.hpp.

7.37.1.171 **#define SUPERBASICSTART 428**

Definition at line 611 of file OSParseosrl.tab.hpp.

7.37.1.172 **#define SUPERBASICEND 429**

Definition at line 612 of file OSParseosrl.tab.hpp.

7.37.1.173 **#define SYSTEMINFORMATIONSTART 430**

Definition at line 613 of file OSParseosrl.tab.hpp.

7.37.1.174 **#define SYSTEMINFORMATIONEND 431**

Definition at line 614 of file OSParseosrl.tab.hpp.

7.37.1.175 **#define TIMESTART 432**

Definition at line 615 of file OSParseosrl.tab.hpp.

7.37.1.176 **#define TIMEEND 433**

Definition at line 616 of file OSParseosrl.tab.hpp.

7.37.1.177 **#define TIMESERVICESTARTEDSTART 434**

Definition at line 617 of file OSParseosrl.tab.hpp.

7.37.1.178 **#define TIMESERVICESTARTEDEND 435**

Definition at line 618 of file OSParseosrl.tab.hpp.

7.37.1.179 **#define TIMESTAMPSTART 436**

Definition at line 619 of file OSParseosrl.tab.hpp.

7.37.1.180 **#define TIMESTAMPEND 437**

Definition at line 620 of file OSParseosrl.tab.hpp.

7.37.1.181 **#define TIMINGINFORMATIONSTART 438**

Definition at line 621 of file OSParseosrl.tab.hpp.

7.37.1.182 **#define TIMINGINFORMATIONEND 439**

Definition at line 622 of file OSParseosrl.tab.hpp.

7.37.1.183 **#define TOTALJOBSSOFARSTART 440**

Definition at line 623 of file OSParseosrl.tab.hpp.

7.37.1.184 **#define TOTALJOBSSOFAREND 441**

Definition at line 624 of file OSParseosrl.tab.hpp.

7.37.1.185 **#define UNKNOWNSTART 442**

Definition at line 625 of file OSParseosrl.tab.hpp.

7.37.1.186 **#define UNKNOWNEND 443**

Definition at line 626 of file OSParseosrl.tab.hpp.

7.37.1.187 **#define USEDPCUNUMBERSTART 444**

Definition at line 627 of file OSParseosrl.tab.hpp.

7.37.1.188 **#define USEDPCUNUMBEREND 445**

Definition at line 628 of file OSParseosrl.tab.hpp.

7.37.1.189 **#define USEDPCUSPEEDSTART 446**

Definition at line 629 of file OSParseosrl.tab.hpp.

7.37.1.190 **#define USEDPCUSPEEDEND 447**

Definition at line 630 of file OSParseosrl.tab.hpp.

7.37.1.191 **#define USEDDISKSPACESTART 448**

Definition at line 631 of file OSParseosrl.tab.hpp.

7.37.1.192 **#define USEDDISKSPACEEND 449**

Definition at line 632 of file OSParseosrl.tab.hpp.

7.37.1.193 **#define USEDMEMORYSTART 450**

Definition at line 633 of file OSParseosrl.tab.hpp.

7.37.1.194 **#define USEDMEMORYEND 451**

Definition at line 634 of file OSParseosrl.tab.hpp.

7.37.1.195 **#define VALUESSTRINGSTART 452**

Definition at line 635 of file OSParseosrl.tab.hpp.

7.37.1.196 **#define VALUESSTRINGEND 453**

Definition at line 636 of file OSParseosrl.tab.hpp.

7.37.1.197 **#define VARSTART 454**

Definition at line 637 of file OSParseosrl.tab.hpp.

7.37.1.198 **#define VAREND 455**

Definition at line 638 of file OSParseosrl.tab.hpp.

7.37.1.199 **#define VARIABLESSTART 456**

Definition at line 639 of file OSParseosrl.tab.hpp.

7.37.1.200 **#define VARIABLESEND 457**

Definition at line 640 of file OSParseosrl.tab.hpp.

7.37.1.201 #define VARIDXSTART 458

Definition at line 641 of file OSParseosrl.tab.hpp.

7.37.1.202 #define VARIDXEND 459

Definition at line 642 of file OSParseosrl.tab.hpp.

7.37.1.203 #define MATRIXVARSTART 460

Definition at line 643 of file OSParseosrl.tab.hpp.

7.37.1.204 #define MATRIXVAREND 461

Definition at line 644 of file OSParseosrl.tab.hpp.

7.37.1.205 #define MATRIXOBJSTART 462

Definition at line 645 of file OSParseosrl.tab.hpp.

7.37.1.206 #define MATRIXOBJEND 463

Definition at line 646 of file OSParseosrl.tab.hpp.

7.37.1.207 #define MATRIXCONSTART 464

Definition at line 647 of file OSParseosrl.tab.hpp.

7.37.1.208 #define MATRIXCONEND 465

Definition at line 648 of file OSParseosrl.tab.hpp.

7.37.1.209 #define FILENAMESTART 466

Definition at line 649 of file OSParseosrl.tab.hpp.

7.37.1.210 #define FILENAMEEND 467

Definition at line 650 of file OSParseosrl.tab.hpp.

7.37.1.211 #define FILENAMEEMPTY 468

Definition at line 651 of file OSParseosrl.tab.hpp.

7.37.1.212 #define FILENAMESTARTANDEND 469

Definition at line 652 of file OSParseosrl.tab.hpp.

7.37.1.213 #define FILESOURCESTART 470

Definition at line 653 of file OSParseosrl.tab.hpp.

7.37.1.214 #define FILESOURCEEND 471

Definition at line 654 of file OSParseosrl.tab.hpp.

7.37.1.215 **#define** FILESOURCEEMPTY 472

Definition at line 655 of file OSParseosrl.tab.hpp.

7.37.1.216 **#define** FILESOURCESTARTANDEND 473

Definition at line 656 of file OSParseosrl.tab.hpp.

7.37.1.217 **#define** FILEDESCRIPTIONSTART 474

Definition at line 657 of file OSParseosrl.tab.hpp.

7.37.1.218 **#define** FILEDESCRIPTIONEND 475

Definition at line 658 of file OSParseosrl.tab.hpp.

7.37.1.219 **#define** FILEDESCRIPTIONEMPTY 476

Definition at line 659 of file OSParseosrl.tab.hpp.

7.37.1.220 **#define** FILEDESCRIPTIONSTARTANDEND 477

Definition at line 660 of file OSParseosrl.tab.hpp.

7.37.1.221 **#define** FILECREATORSTART 478

Definition at line 661 of file OSParseosrl.tab.hpp.

7.37.1.222 **#define** FILECREATOREND 479

Definition at line 662 of file OSParseosrl.tab.hpp.

7.37.1.223 **#define** FILECREATOREMPTY 480

Definition at line 663 of file OSParseosrl.tab.hpp.

7.37.1.224 **#define** FILECREATORSTARTANDEND 481

Definition at line 664 of file OSParseosrl.tab.hpp.

7.37.1.225 **#define** FILELICENCESTART 482

Definition at line 665 of file OSParseosrl.tab.hpp.

7.37.1.226 **#define** FILELICENCEEND 483

Definition at line 666 of file OSParseosrl.tab.hpp.

7.37.1.227 **#define** FILELICENCEEMPTY 484

Definition at line 667 of file OSParseosrl.tab.hpp.

7.37.1.228 **#define** FILELICENCESTARTANDEND 485

Definition at line 668 of file OSParseosrl.tab.hpp.

7.37.1.229 **#define MATRIXSTART 486**

Definition at line 669 of file OSParseosrl.tab.hpp.

7.37.1.230 **#define MATRIXEND 487**

Definition at line 670 of file OSParseosrl.tab.hpp.

7.37.1.231 **#define BASEMATRIXEND 488**

Definition at line 671 of file OSParseosrl.tab.hpp.

7.37.1.232 **#define BASEMATRIXSTART 489**

Definition at line 672 of file OSParseosrl.tab.hpp.

7.37.1.233 **#define BLOCKSTART 490**

Definition at line 673 of file OSParseosrl.tab.hpp.

7.37.1.234 **#define BLOCKEND 491**

Definition at line 674 of file OSParseosrl.tab.hpp.

7.37.1.235 **#define BLOCKSSTART 492**

Definition at line 675 of file OSParseosrl.tab.hpp.

7.37.1.236 **#define BLOCKSEND 493**

Definition at line 676 of file OSParseosrl.tab.hpp.

7.37.1.237 **#define EMPTYSHAPEATT 494**

Definition at line 677 of file OSParseosrl.tab.hpp.

7.37.1.238 **#define SHAPEATT 495**

Definition at line 678 of file OSParseosrl.tab.hpp.

7.37.1.239 **#define EMPTYSYMMETRYATT 496**

Definition at line 679 of file OSParseosrl.tab.hpp.

7.37.1.240 **#define SYMMETRYATT 497**

Definition at line 680 of file OSParseosrl.tab.hpp.

7.37.1.241 **#define EMPTYNEGATIVEPATTERNATT 498**

Definition at line 681 of file OSParseosrl.tab.hpp.

7.37.1.242 **#define NEGATIVEPATTERNATT 499**

Definition at line 682 of file OSParseosrl.tab.hpp.

7.37.1.243 #define CONSTANTATT 500

Definition at line 683 of file OSParseosrl.tab.hpp.

7.37.1.244 #define NUMBEROFBLOCKSATT 501

Definition at line 684 of file OSParseosrl.tab.hpp.

7.37.1.245 #define NUMBEROFCOLUMNSATT 502

Definition at line 685 of file OSParseosrl.tab.hpp.

7.37.1.246 #define NUMBEROFROWSATT 503

Definition at line 686 of file OSParseosrl.tab.hpp.

7.37.1.247 #define NUMBEROFVALUESATT 504

Definition at line 687 of file OSParseosrl.tab.hpp.

7.37.1.248 #define COEFATT 505

Definition at line 688 of file OSParseosrl.tab.hpp.

7.37.1.249 #define BASEMATRIXIDXATT 506

Definition at line 689 of file OSParseosrl.tab.hpp.

7.37.1.250 #define TARGETMATRIXFIRSTROWATT 507

Definition at line 690 of file OSParseosrl.tab.hpp.

7.37.1.251 #define TARGETMATRIXFIRSTCOLATT 508

Definition at line 691 of file OSParseosrl.tab.hpp.

7.37.1.252 #define BASEMATRIXSTARTROWATT 509

Definition at line 692 of file OSParseosrl.tab.hpp.

7.37.1.253 #define BASEMATRIXSTARTCOLATT 510

Definition at line 693 of file OSParseosrl.tab.hpp.

7.37.1.254 #define BASEMATRIXENDROWATT 511

Definition at line 694 of file OSParseosrl.tab.hpp.

7.37.1.255 #define BASEMATRIXENDCOLATT 512

Definition at line 695 of file OSParseosrl.tab.hpp.

7.37.1.256 #define SCALARMULTIPLIERATT 513

Definition at line 696 of file OSParseosrl.tab.hpp.

7.37.1.257 **#define EMPTYBASETRANSPOSEATT 514**

Definition at line 697 of file OSParseosrl.tab.hpp.

7.37.1.258 **#define BASETRANSPOSEATT 515**

Definition at line 698 of file OSParseosrl.tab.hpp.

7.37.1.259 **#define ELEMENTSSTART 516**

Definition at line 699 of file OSParseosrl.tab.hpp.

7.37.1.260 **#define ELEMENTSEND 517**

Definition at line 700 of file OSParseosrl.tab.hpp.

7.37.1.261 **#define CONSTANTELEMENTSSTART 518**

Definition at line 701 of file OSParseosrl.tab.hpp.

7.37.1.262 **#define CONSTANTELEMENTSEND 519**

Definition at line 702 of file OSParseosrl.tab.hpp.

7.37.1.263 **#define STARTVECTORSTART 520**

Definition at line 703 of file OSParseosrl.tab.hpp.

7.37.1.264 **#define STARTVECTOREND 521**

Definition at line 704 of file OSParseosrl.tab.hpp.

7.37.1.265 **#define NONZEROSSTART 522**

Definition at line 705 of file OSParseosrl.tab.hpp.

7.37.1.266 **#define NONZEROSEND 523**

Definition at line 706 of file OSParseosrl.tab.hpp.

7.37.1.267 **#define INDEXESSTART 524**

Definition at line 707 of file OSParseosrl.tab.hpp.

7.37.1.268 **#define INDEXESEND 525**

Definition at line 708 of file OSParseosrl.tab.hpp.

7.37.1.269 **#define VALUESSTART 526**

Definition at line 709 of file OSParseosrl.tab.hpp.

7.37.1.270 **#define VALUESEND 527**

Definition at line 710 of file OSParseosrl.tab.hpp.

7.37.1.271 #define VARREFERENCEELEMENTSSTART 528

Definition at line 711 of file OSParseosrl.tab.hpp.

7.37.1.272 #define VARREFERENCEELEMENTSEND 529

Definition at line 712 of file OSParseosrl.tab.hpp.

7.37.1.273 #define LINEARELEMENTSSTART 530

Definition at line 713 of file OSParseosrl.tab.hpp.

7.37.1.274 #define LINEARELEMENTSEND 531

Definition at line 714 of file OSParseosrl.tab.hpp.

7.37.1.275 #define GENERALELEMENTSSTART 532

Definition at line 715 of file OSParseosrl.tab.hpp.

7.37.1.276 #define GENERALELEMENTSEND 533

Definition at line 716 of file OSParseosrl.tab.hpp.

7.37.1.277 #define CONREFERENCEELEMENTSSTART 534

Definition at line 717 of file OSParseosrl.tab.hpp.

7.37.1.278 #define CONREFERENCEELEMENTSEND 535

Definition at line 718 of file OSParseosrl.tab.hpp.

7.37.1.279 #define VALUETYPEATT 536

Definition at line 719 of file OSParseosrl.tab.hpp.

7.37.1.280 #define OBJREFERENCEELEMENTSSTART 537

Definition at line 720 of file OSParseosrl.tab.hpp.

7.37.1.281 #define OBJREFERENCEELEMENTSEND 538

Definition at line 721 of file OSParseosrl.tab.hpp.

7.37.1.282 #define PATTERNELEMENTSSTART 539

Definition at line 722 of file OSParseosrl.tab.hpp.

7.37.1.283 #define PATTERNELEMENTSEND 540

Definition at line 723 of file OSParseosrl.tab.hpp.

7.37.1.284 #define TRANSFORMATIONSTART 541

Definition at line 724 of file OSParseosrl.tab.hpp.

7.37.1.285 **#define TRANSFORMATIONEND** 542

Definition at line 725 of file OSParseosrl.tab.hpp.

7.37.1.286 **#define COLOFFSETSSTART** 543

Definition at line 726 of file OSParseosrl.tab.hpp.

7.37.1.287 **#define COLOFFSETSEND** 544

Definition at line 727 of file OSParseosrl.tab.hpp.

7.37.1.288 **#define ROWOFFSETSSTART** 545

Definition at line 728 of file OSParseosrl.tab.hpp.

7.37.1.289 **#define ROWOFFSETSEND** 546

Definition at line 729 of file OSParseosrl.tab.hpp.

7.37.1.290 **#define EMPTYROWMAJORATT** 547

Definition at line 730 of file OSParseosrl.tab.hpp.

7.37.1.291 **#define ROWMAJORATT** 548

Definition at line 731 of file OSParseosrl.tab.hpp.

7.37.1.292 **#define BLOCKROWIDXATT** 549

Definition at line 732 of file OSParseosrl.tab.hpp.

7.37.1.293 **#define BLOCKCOLIDXATT** 550

Definition at line 733 of file OSParseosrl.tab.hpp.

7.37.1.294 **#define DUMMY** 551

Definition at line 734 of file OSParseosrl.tab.hpp.

7.37.1.295 **#define NONLINEAREXPRESSIONSSTART** 552

Definition at line 735 of file OSParseosrl.tab.hpp.

7.37.1.296 **#define NONLINEAREXPRESSIONSEND** 553

Definition at line 736 of file OSParseosrl.tab.hpp.

7.37.1.297 **#define NUMBEROFNONLINEAREXPRESSIONS** 554

Definition at line 737 of file OSParseosrl.tab.hpp.

7.37.1.298 **#define NLSTART** 555

Definition at line 738 of file OSParseosrl.tab.hpp.

7.37.1.299 #define NLEND 556

Definition at line 739 of file OSParseosrl.tab.hpp.

7.37.1.300 #define MATRIXEXPRESSIONSSTART 557

Definition at line 740 of file OSParseosrl.tab.hpp.

7.37.1.301 #define MATRIXEXPRESSIONSEND 558

Definition at line 741 of file OSParseosrl.tab.hpp.

7.37.1.302 #define NUMBEROFEXPR 559

Definition at line 742 of file OSParseosrl.tab.hpp.

7.37.1.303 #define EXPRSTART 560

Definition at line 743 of file OSParseosrl.tab.hpp.

7.37.1.304 #define EXPREND 561

Definition at line 744 of file OSParseosrl.tab.hpp.

7.37.1.305 #define NUMBEROFMATRIXTERMSATT 562

Definition at line 745 of file OSParseosrl.tab.hpp.

7.37.1.306 #define MATRIXTERMSTART 563

Definition at line 746 of file OSParseosrl.tab.hpp.

7.37.1.307 #define MATRIXTERMEND 564

Definition at line 747 of file OSParseosrl.tab.hpp.

7.37.1.308 #define POWERSTART 565

Definition at line 748 of file OSParseosrl.tab.hpp.

7.37.1.309 #define POWEREND 566

Definition at line 749 of file OSParseosrl.tab.hpp.

7.37.1.310 #define PLUSSTART 567

Definition at line 750 of file OSParseosrl.tab.hpp.

7.37.1.311 #define PLUSEND 568

Definition at line 751 of file OSParseosrl.tab.hpp.

7.37.1.312 #define MINUSSTART 569

Definition at line 752 of file OSParseosrl.tab.hpp.

7.37.1.313 #define MINUSEND 570

Definition at line 753 of file OSParseosrl.tab.hpp.

7.37.1.314 #define DIVIDESTART 571

Definition at line 754 of file OSParseosrl.tab.hpp.

7.37.1.315 #define DIVIDEEND 572

Definition at line 755 of file OSParseosrl.tab.hpp.

7.37.1.316 #define LNSTART 573

Definition at line 756 of file OSParseosrl.tab.hpp.

7.37.1.317 #define LNEND 574

Definition at line 757 of file OSParseosrl.tab.hpp.

7.37.1.318 #define SQRTSTART 575

Definition at line 758 of file OSParseosrl.tab.hpp.

7.37.1.319 #define SQRTEND 576

Definition at line 759 of file OSParseosrl.tab.hpp.

7.37.1.320 #define SUMSTART 577

Definition at line 760 of file OSParseosrl.tab.hpp.

7.37.1.321 #define SUMEND 578

Definition at line 761 of file OSParseosrl.tab.hpp.

7.37.1.322 #define PRODUCTSTART 579

Definition at line 762 of file OSParseosrl.tab.hpp.

7.37.1.323 #define PRODUCTEND 580

Definition at line 763 of file OSParseosrl.tab.hpp.

7.37.1.324 #define EXPSTART 581

Definition at line 764 of file OSParseosrl.tab.hpp.

7.37.1.325 #define EXPEND 582

Definition at line 765 of file OSParseosrl.tab.hpp.

7.37.1.326 #define NEGATESTART 583

Definition at line 766 of file OSParseosrl.tab.hpp.

7.37.1.327 #define NEGATEEND 584

Definition at line 767 of file OSParseosrl.tab.hpp.

7.37.1.328 #define IFSTART 585

Definition at line 768 of file OSParseosrl.tab.hpp.

7.37.1.329 #define IFEND 586

Definition at line 769 of file OSParseosrl.tab.hpp.

7.37.1.330 #define SQUARESTART 587

Definition at line 770 of file OSParseosrl.tab.hpp.

7.37.1.331 #define SQUAREEND 588

Definition at line 771 of file OSParseosrl.tab.hpp.

7.37.1.332 #define COSSTART 589

Definition at line 772 of file OSParseosrl.tab.hpp.

7.37.1.333 #define COSEND 590

Definition at line 773 of file OSParseosrl.tab.hpp.

7.37.1.334 #define SINSTART 591

Definition at line 774 of file OSParseosrl.tab.hpp.

7.37.1.335 #define SINEND 592

Definition at line 775 of file OSParseosrl.tab.hpp.

7.37.1.336 #define VARIABLESTART 593

Definition at line 776 of file OSParseosrl.tab.hpp.

7.37.1.337 #define VARIABLEEND 594

Definition at line 777 of file OSParseosrl.tab.hpp.

7.37.1.338 #define ABSSTART 595

Definition at line 778 of file OSParseosrl.tab.hpp.

7.37.1.339 #define ABSEND 596

Definition at line 779 of file OSParseosrl.tab.hpp.

7.37.1.340 #define ERFSTART 597

Definition at line 780 of file OSParseosrl.tab.hpp.

7.37.1.341 **#define** ERFEND 598

Definition at line 781 of file OSParseosrl.tab.hpp.

7.37.1.342 **#define** MAXSTART 599

Definition at line 782 of file OSParseosrl.tab.hpp.

7.37.1.343 **#define** MAXEND 600

Definition at line 783 of file OSParseosrl.tab.hpp.

7.37.1.344 **#define** ALLDIFFSTART 601

Definition at line 784 of file OSParseosrl.tab.hpp.

7.37.1.345 **#define** ALLDIFFEND 602

Definition at line 785 of file OSParseosrl.tab.hpp.

7.37.1.346 **#define** MINSTART 603

Definition at line 786 of file OSParseosrl.tab.hpp.

7.37.1.347 **#define** MINEND 604

Definition at line 787 of file OSParseosrl.tab.hpp.

7.37.1.348 **#define** ESTART 605

Definition at line 788 of file OSParseosrl.tab.hpp.

7.37.1.349 **#define** EEND 606

Definition at line 789 of file OSParseosrl.tab.hpp.

7.37.1.350 **#define** PISTART 607

Definition at line 790 of file OSParseosrl.tab.hpp.

7.37.1.351 **#define** PIEND 608

Definition at line 791 of file OSParseosrl.tab.hpp.

7.37.1.352 **#define** TIMESSTART 609

Definition at line 792 of file OSParseosrl.tab.hpp.

7.37.1.353 **#define** TIMESEND 610

Definition at line 793 of file OSParseosrl.tab.hpp.

7.37.1.354 **#define** NUMBERSTART 611

Definition at line 794 of file OSParseosrl.tab.hpp.

7.37.1.355 **#define NUMBEREND 612**

Definition at line 795 of file OSParseosrl.tab.hpp.

7.37.1.356 **#define MATRIXDETERMINANTSTART 613**

Definition at line 796 of file OSParseosrl.tab.hpp.

7.37.1.357 **#define MATRIXDETERMINANTEND 614**

Definition at line 797 of file OSParseosrl.tab.hpp.

7.37.1.358 **#define MATRIXTRACESTART 615**

Definition at line 798 of file OSParseosrl.tab.hpp.

7.37.1.359 **#define MATRIXTRACEEND 616**

Definition at line 799 of file OSParseosrl.tab.hpp.

7.37.1.360 **#define MATRXTOSCALARSTART 617**

Definition at line 800 of file OSParseosrl.tab.hpp.

7.37.1.361 **#define MATRXTOSCALAREND 618**

Definition at line 801 of file OSParseosrl.tab.hpp.

7.37.1.362 **#define MATRIXDIAGONALSTART 619**

Definition at line 802 of file OSParseosrl.tab.hpp.

7.37.1.363 **#define MATRIXDIAGONALEND 620**

Definition at line 803 of file OSParseosrl.tab.hpp.

7.37.1.364 **#define MATRIXDOTTIMESSTART 621**

Definition at line 804 of file OSParseosrl.tab.hpp.

7.37.1.365 **#define MATRIXDOTTIMSEND 622**

Definition at line 805 of file OSParseosrl.tab.hpp.

7.37.1.366 **#define MATRIXLOWERTRIANGLESTART 623**

Definition at line 806 of file OSParseosrl.tab.hpp.

7.37.1.367 **#define MATRIXLOWERTRIANGLEEND 624**

Definition at line 807 of file OSParseosrl.tab.hpp.

7.37.1.368 **#define MATRIXUPPERTRIANGLESTART 625**

Definition at line 808 of file OSParseosrl.tab.hpp.

7.37.1.369 **#define MATRIXUPPERTRIANGLEEND 626**

Definition at line 809 of file OSParseosrl.tab.hpp.

7.37.1.370 **#define MATRIXMERGESTART 627**

Definition at line 810 of file OSParseosrl.tab.hpp.

7.37.1.371 **#define MATRIXMERGEEND 628**

Definition at line 811 of file OSParseosrl.tab.hpp.

7.37.1.372 **#define MATRIXMINUSSTART 629**

Definition at line 812 of file OSParseosrl.tab.hpp.

7.37.1.373 **#define MATRIXMINUSEND 630**

Definition at line 813 of file OSParseosrl.tab.hpp.

7.37.1.374 **#define MATRIXNEGATESTART 631**

Definition at line 814 of file OSParseosrl.tab.hpp.

7.37.1.375 **#define MATRIXNEGATEEND 632**

Definition at line 815 of file OSParseosrl.tab.hpp.

7.37.1.376 **#define MATRIXPLUSSTART 633**

Definition at line 816 of file OSParseosrl.tab.hpp.

7.37.1.377 **#define MATRIXPLUSEND 634**

Definition at line 817 of file OSParseosrl.tab.hpp.

7.37.1.378 **#define MATRIXTIMESSTART 635**

Definition at line 818 of file OSParseosrl.tab.hpp.

7.37.1.379 **#define MATRIXTIMSEEND 636**

Definition at line 819 of file OSParseosrl.tab.hpp.

7.37.1.380 **#define MATRIXPRODUCTSTART 637**

Definition at line 820 of file OSParseosrl.tab.hpp.

7.37.1.381 **#define MATRIXPRODUCTEND 638**

Definition at line 821 of file OSParseosrl.tab.hpp.

7.37.1.382 **#define MATRIXSCALARTIMESSTART 639**

Definition at line 822 of file OSParseosrl.tab.hpp.

7.37.1.383 **#define MATRIXSCALARTIMESEND 640**

Definition at line 823 of file OSParseosrl.tab.hpp.

7.37.1.384 **#define MATRIXSUBMATRIXATSTART 641**

Definition at line 824 of file OSParseosrl.tab.hpp.

7.37.1.385 **#define MATRIXSUBMATRIXATEND 642**

Definition at line 825 of file OSParseosrl.tab.hpp.

7.37.1.386 **#define MATRIXTRANPOSESTART 643**

Definition at line 826 of file OSParseosrl.tab.hpp.

7.37.1.387 **#define MATRIXTRANPOSEEND 644**

Definition at line 827 of file OSParseosrl.tab.hpp.

7.37.1.388 **#define MATRIXREFERENCESTART 645**

Definition at line 828 of file OSParseosrl.tab.hpp.

7.37.1.389 **#define MATRIXREFERENCEEND 646**

Definition at line 829 of file OSParseosrl.tab.hpp.

7.37.1.390 **#define IDENTITYMATRIXSTART 647**

Definition at line 830 of file OSParseosrl.tab.hpp.

7.37.1.391 **#define IDENTITYMATRIXEND 648**

Definition at line 831 of file OSParseosrl.tab.hpp.

7.37.1.392 **#define MATRIXINVERSESTART 649**

Definition at line 832 of file OSParseosrl.tab.hpp.

7.37.1.393 **#define MATRIXINVERSEEND 650**

Definition at line 833 of file OSParseosrl.tab.hpp.

7.37.1.394 **#define EMPTYINCLUDEDIAGONALATT 651**

Definition at line 834 of file OSParseosrl.tab.hpp.

7.37.1.395 **#define INCLUDEDIAGONALATT 652**

Definition at line 835 of file OSParseosrl.tab.hpp.

7.37.1.396 **#define IDATT 653**

Definition at line 836 of file OSParseosrl.tab.hpp.

7.37.1.397 `#define YYSTYPE_IS_TRIVIAL 1`

Definition at line 853 of file OSParseosrl.tab.hpp.

7.37.1.398 `#define YYSTYPE /* obsolescent; will be withdrawn */`

Definition at line 854 of file OSParseosrl.tab.hpp.

7.37.1.399 `#define YYSTYPE_IS_DECLARED 1`

Definition at line 855 of file OSParseosrl.tab.hpp.

7.37.1.400 `#define yyLtype YYLTYPE /* obsolescent; will be withdrawn */`

Definition at line 868 of file OSParseosrl.tab.hpp.

7.37.1.401 `#define YYLTYPE_IS_DECLARED 1`

Definition at line 869 of file OSParseosrl.tab.hpp.

7.37.1.402 `#define YYLTYPE_IS_TRIVIAL 1`

Definition at line 870 of file OSParseosrl.tab.hpp.

7.37.2 Typedef Documentation

7.37.2.1 `typedef union YYSTYPE YYSTYPE`

7.37.2.2 `typedef struct YYLTYPE YYLTYPE`

7.37.3 Enumeration Type Documentation

7.37.3.1 `enum yytokentype`

Enumerator:

QUOTE

ATTRIBUTETEXT

ELEMENTTEXT

ITEMTEXT

INTEGER

DOUBLE

TWOQUOTES

ENDOFELEMENT

GREATERTHAN

OSILEND

INSTANCEDATAEND

INSTANCEDATASTARTEND

EMPTYIDATT

IDXONEATT

IDXTWOATT

VALUEATT
QUADRATICCOEFFICIENTSSTART
QUADRATICCOEFFICIENTSEND
NUMBEROFQTERMSATT
QTERMSTART
QTERMEND
MATRICESSTART
MATRICESEND
NUMBEROFMATRICESATT
CONESSTART
CONESEND
NUMBEROFCONESATT
NONNEGATIVECONESTART
NONNEGATIVECONEEND
NONPOSITIVECONESTART
NONPOSITIVECONEEND
ORTHANTCONESTART
ORTHANTCONEEND
POLYHEDRALCONESTART
POLYHEDRALCONEEND
QUADRATICCONESTART
QUADRATICCONEEND
ROTATEDQUADRATICCONESTART
ROTATEDQUADRATICCONEEND
SEMIDEFINITECONESTART
SEMIDEFINITECONEEND
PRODUCTCONESTART
PRODUCTCONEEND
INTERSECTIONCONESTART
INTERSECTIONCONEEND
DUALCONESTART
DUALCONEEND
POLARCONESTART
POLARCONEEND
DIRECTIONSTART
DIRECTIONEND
FACTORSSTART
FACTORSEND
COMPONENTSSTART
COMPONENTSEND
NORMSCALEFACTORATT
DISTORTIONMATRIXIDXATT

AXISDIRECTIONATT
FIRSTAXISDIRECTIONATT
SECONDAXISDIRECTIONATT
EMPTYSEMIDEFINITENESSATT
SEMIDEFINITENESSATT
REFERENCEMATRIXIDXATT
MATRIXPROGRAMMINGSTART
MATRIXPROGRAMMINGEND
VARTYPEATT
MATRIXVARIABLESSTART
MATRIXVARIABLESEND
NUMBEROFMATRIXVARATT
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJECTIVESSTART
MATRIXOBJECTIVESEND
NUMBEROFMATRIXOBJATT
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTRAINTSSTART
MATRIXCONSTRAINTSEND
NUMBEROFMATRIXCONATT
MATRIXCONSTART
MATRIXCONEND
MATRIXIDXATT
LBMATRIXIDXATT
LBCONEIDXATT
UBMATRIXIDXATT
UBCONEIDXATT
TEMPLATEMATRIXIDXATT
VARREFERENCEMATRIXIDXATT
OBJREFERENCEMATRIXIDXATT
CONREFERENCEMATRIXIDXATT
ORDERCONEIDXATT
CONSTANTMATRIXIDXATT
TIMEDOMAINSTART
TIMEDOMAINEND
STAGESSTART
STAGESEND
STAGESTART
STAGEEND
NUMBEROFSTAGESATT

HORIZONATT
STARTATT
VARIABLESSTART
CONSTRAINTSSTART
OBJECTIVESSTART
VARIABLESEND
CONSTRAINTSEND
OBJECTIVESEND
NUMBEROFVARIABLESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOBJECTIVESATT
STARTIDXATT
VARSTART
VAREND
CONSTART
CONEND
OBJSTART
OBJEND
INTERVALSTART
INTERVALEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND

ENUMERATIONSTART
ENUMERATIONEND
NUMBEROFELATT
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYNAMEATT
NAMEATT
EMPTYTYPEATT
TYPEATT
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
NUMBEROFVARIDXATT
IDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT

TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPOSEATT
BASETRANSPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COOFFSETSTART
COOFFSETSEND
ROWOFFSETSTART
ROWOFFSETSEND
EMPTYROWMAJORATT

ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART

COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART

MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMSEEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMSEEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANPOSESTART
MATRIXTRANPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTETEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE
TWOQUOTES
GREATERTHAN
ENDOFELEMENT
OSOLSTART
OSOLSTARTEMPT
OSOLATTRIBUTETEXT
SOLEND
NUMBEROFOTHEROPTIONSATT
NUMBEROFENUMERATIONSATT
NUMBEROFJOBIDSATT
NUMBEROFPATHSATT
NUMBEROFPATHPAIRSATT
FROMATT
TOATT

MAKECOPYATT
CATEGORYATT
TYPEATT
GROUPWEIGHTATT
NUMBEROFPROCESSESATT
NUMBEROFSOLVEROPTIONSATT
NUMBEROFSOSATT
NUMBEROFVARIABLESATT
NUMBEROFOBJECTIVESATT
NUMBEROFCONSTRAINTSATT
NUMBEROFOTHERVARIABLEOPTIONSATT
NUMBEROFOTHEROBJECTIVEOPTIONSATT
NUMBEROFOTHERCONSTRAINTOPTIONSATT
NUMBEROFITEMSATT
NUMBEROFVARATT
NUMBEROFOBJATT
NUMBEROFCONATT
NUMBEROFELATT
NAMEATT
IDXATT
SOSIDXATT
VALUEATT
UNITATT
DESCRIPTIONATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT
OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
EMPTYTYPEATT
EMPTYNAMEATT
EMPTYCATEGORYATT
EMPTYDESCRIPTIONATT
EMPTYUNITATT
EMPTYVALUEATT
EMPTYLBVALUEATT
EMPTYUBVALUEATT
LBVALUEATT
UBVALUEATT

EMPTYLBDUALVALUEATT
EMPTYUBDUALVALUEATT
LBDUALVALUEATT
UBDUALVALUEATT
SOLVERATT
EMPTYSOLVERATT
WEIGHTATT
EMPTYWEIGHTATT
TRANSPORTTYPEATT
LOCATIONTYPEATT
GENERALSTART
GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND
JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
SERVICEURISTART
SERVICEURIEND
SERVICENAMESTART
SERVICENAMEEND
INSTANCENAMESTART
INSTANCENAMEEND
INSTANCELOCATIONSTART
INSTANCELOCATIONEND
JOBIDSTART
JOBIDEND
SOLVERTOINVOKESTART
SOLVERTOINVOKEEND
LICENSESTART
LICENSEEND
USERNAMESTART
USERNAMEEND
PASSWORDSTART
PASSWORDEND
CONTACTSTART
CONTACTEND
OTHEROPTIONSSTART
OTHEROPTIONSEND

OTHERSTART
OTHEREND
MINDISKSPACESTART
MINDISKSPACEEND
MINMEMORYSTART
MINMEMORYEND
MINCPUSPEEDSTART
MINCPUSPEEDEND
MINCPUNUMBERSTART
MINCPUNUMBEREND
SERVICETYPESTART
SERVICETYPEEND
MAXTIMESTART
MAXTIMEEND
REQUESTEDSTARTTIMESTART
REQUESTEDSTARTTIMEEND
DEPENDENCIESSTART
DEPENDENCIESEND
REQUIREDDIRECTORIESSTART
REQUIREDDIRECTORIESEND
REQUIREDFILESSTART
REQUIREDFILESEND
PATHSTART
PATHEND
PATHPAIRSTART
PATHPAIREND
DIRECTORIESTOMAKESTART
DIRECTORIESTOMAKEEND
FILESTOMAKESTART
FILESTOMAKEEND
DIRECTORIESTODELETESTART
DIRECTORIESTODELETEEND
FILESTODELETESTART
FILESTODELETEEND
INPUTDIRECTORIESTOMOVESTART
INPUTDIRECTORIESTOMOVEEND
INPUTFILESTOMOVESTART
INPUTFILESTOMOVEEND
OUTPUTDIRECTORIESTOMOVESTART
OUTPUTDIRECTORIESTOMOVEEND
OUTPUTFILESTOMOVESTART
OUTPUTFILESTOMOVEEND

PROCESSESTOKILLSTART
PROCESSESTOKILLEND
PROCESSSTART
PROCESSEND
VARIABLESSTART
VARIABLESEND
INITIALVARIABLEVALUESSTART
INITIALVARIABLEVALUESEND
VARSTART
VAREND
INITIALVARIABLEVALUESSTRINGSTART
INITIALVARIABLEVALUESSTRINGEND
INITIALBASISSTATUSSTART
INITIALBASISSTATUSEND
BASICSTART
BASICEND
ATUPPERSTART
ATUPPEREND
ATLOWERSTART
ATLOWEREND
ATEQUALITYSTART
ATEQUALITYEND
SUPERBASICSTART
SUPERBASICEND
ISFREESTART
ISFREEEND
UNKNOWNSTART
UNKNOWNEND
INTEGERVARIABLEBRANCHINGWEIGHTSSTART
INTEGERVARIABLEBRANCHINGWEIGHTSEND
SOSVARIABLEBRANCHINGWEIGHTSSTART
SOSVARIABLEBRANCHINGWEIGHTSEND
SOSSTART
SOSEND
OBJECTIVESSTART
OBJECTIVESEND
INITIALOBJECTIVEVALUESSTART
INITIALOBJECTIVEVALUESEND
OBJSTART
OBJEND
INITIALOBJECTIVEBOUNDSSTART
INITIALOBJECTIVEBOUNDSEND

CONSTRAINTSSTART
CONSTRAINTSEND
INITIALCONSTRAINTVALUESSTART
INITIALCONSTRAINTVALUESEND
CONSTART
CONEND
INITIALDUALVALUESSTART
INITIALDUALVALUESEND
SOLVEROPTIONSSTART
SOLVEROPTIONSEND
SOLVEROPTIONSTART
SOLVEROPTIONEND
ENUMERATIONSTART
ENUMERATIONEND
ITEMEMPTY
ITEMSTART
ITEMEND
ITEMSTARTANDEND
BASE64START
BASE64END
INCRATT
MULTATT
SIZEOFATT
ELSTART
ELEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
HEADERSTART
HEADEREND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART

FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT
EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
NUMBEROFVARIDXATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPPOSEATT
BASETRANSPPOSEATT

ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART
OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
VARIDXSTART
VARIDXEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART

NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPEND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART
PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART

ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMESEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART

MATRIXSCALARTIMESEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANPOSESTART
MATRIXTRANPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT
ATTRIBUTE TEXT
ELEMENTTEXT
ITEMTEXT
INTEGER
DOUBLE
QUOTE
TWOQUOTES
GREATER THAN
END OF ELEMENT
OSRLSTART
OSRLSTARTEMPT
OSRLATTRIBUTE TEXT
OSRLEND
NUMBER OF CON ATT
NUMBER OF CONSTRAINTS ATT
NUMBER OF EL ATT
NUMBER OF ENUMERATIONS ATT
NUMBER OF IDX ATT
NUMBER OF ITEMS ATT
NUMBER OF OBJ ATT
NUMBER OF OBJECTIVES ATT
NUMBER OF OTHER CONSTRAINT RESULTS ATT
NUMBER OF OTHER OBJECTIVE RESULTS ATT
NUMBER OF OTHER RESULTS ATT
NUMBER OF OTHER SOLUTION RESULTS ATT
NUMBER OF OTHER VARIABLE RESULTS ATT
NUMBER OF SOLUTIONS ATT
NUMBER OF SOLVER OUTPUTS ATT

NUMBEROFSUBSTATUSESATT
NUMBEROFTIMESATT
NUMBEROFVARATT
NUMBEROFVARIABLESATT
NUMBEROFVARIDXATT
TARGETOBJECTIVEIDXATT
IDXATT
INCRATT
MULTATT
SIZEOFATT
CATEGORYATT
EMPTYCATEGORYATT
DESCRIPTIONATT
EMPTYDESCRIPTIONATT
NAMEATT
EMPTYNAMEATT
TYPEATT
EMPTYTYPEATT
CONTYPEATT
EMPTYCONTYPEATT
ENUMTYPEATT
EMPTYENUMTYPEATT
OBJTYPEATT
EMPTYOBJTYPEATT
VARTYPEATT
EMPTYVARTYPEATT
UNITATT
EMPTYUNITATT
VALUEATT
EMPTYVALUEATT
WEIGHTEDOBJECTIVESATT
EMPTYWEIGHTEDOBJECTIVESATT
TARGETOBJECTIVENAMEATT
EMPTYTARGETOBJECTIVENAMEATT
HEADERSTART
HEADEREND
GENERALSTART
GENERALEND
SYSTEMSTART
SYSTEMEND
SERVICESTART
SERVICEEND

JOBSTART
JOBEND
OPTIMIZATIONSTART
OPTIMIZATIONEND
ITEMSTART
ITEMEND
ITEMSTARTANDEND
ITEMEMPTY
ACTUALSTARTTIMESTART
ACTUALSTARTTIMEEND
ATEQUALITYSTART
ATEQUALITYEND
ATLOWERSTART
ATLOWEREND
ATUPPERSTART
ATUPPEREND
AVAILABLECPUNUMBERSTART
AVAILABLECPUNUMBEREND
AVAILABLECPUSPEEDSTART
AVAILABLECPUSPEEDEND
AVAILABLEDISKSPACESTART
AVAILABLEDISKSPACEEND
AVAILABLEMEMORYSTART
AVAILABLEMEMORYEND
BASE64START
BASE64END
BASICSTART
BASICEND
BASISSTATUSSTART
BASISSTATUSEND
BASSTATUSSTART
BASSTATUSEND
CONSTART
CONEND
CONSTRAINTSSTART
CONSTRAINTSEND
CURRENTJOBCOUNTSTART
CURRENTJOBCOUNTEND
CURRENTSTATESTART
CURRENTSTATEEND
DUALVALUESSTART
DUALVALUESEND

ELSTART
ELEND
ENUMERATIONSTART
ENUMERATIONEND
ENDTIMESTART
ENDTIMEEND
GENERALSTATUSSTART
GENERALSTATUSEND
GENERALSUBSTATUSSTART
GENERALSUBSTATUSEND
IDXSTART
IDXEND
INSTANCENAMESTART
INSTANCENAMEEND
ISFREESTART
ISFREEEND
JOBIDSTART
JOBIDEND
MESSAGESTART
MESSAGEEND
OBJSTART
OBJEND
OBJECTIVESSTART
OBJECTIVESEND
OPTIMIZATIONSOLUTIONSTATUSSTART
OPTIMIZATIONSOLUTIONSTATUSEND
OPTIMIZATIONSOLUTIONSUBSTATUSSTART
OPTIMIZATIONSOLUTIONSUBSTATUSEND
OTHERSTART
OTHEREND
OTHERRESULTSSTART
OTHERRESULTSEND
OTHERSOLUTIONRESULTSTART
OTHERSOLUTIONRESULTEND
OTHERSOLUTIONRESULTSSTART
OTHERSOLUTIONRESULTSEND
OTHERSOLVEROUTPUTSTART
OTHERSOLVEROUTPUTEND
SCHEDULEDSTARTTIMESTART
SCHEDULEDSTARTTIMEEND
SERVICENAMESTART
SERVICENAMEEND

SERVICEURISTART
SERVICEURIEND
SERVICEUTILIZATIONSTART
SERVICEUTILIZATIONEND
SOLUTIONSTART
SOLUTIONEND
SOLVERINVOKEDSTART
SOLVERINVOKEDEND
SOLVEROUTPUTSTART
SOLVEROUTPUTEND
STATUSSTART
STATUSEND
SUBMITTIMESTART
SUBMITTIMEEND
SUBSTATUSSTART
SUBSTATUSEND
SUPERBASICSTART
SUPERBASICEND
SYSTEMINFORMATIONSTART
SYSTEMINFORMATIONEND
TIMESTART
TIMEEND
TIMESERVICESTARTEDSTART
TIMESERVICESTARTEDEND
TIMESTAMPSTART
TIMESTAMPEND
TIMINGINFORMATIONSTART
TIMINGINFORMATIONEND
TOTALJOBSSOFARSTART
TOTALJOBSSOFAREND
UNKNOWNSTART
UNKNOWNEND
USEDCPUNUMBERSTART
USEDCPUNUMBEREND
USEDCPUSPEEDSTART
USEDCPUSPEEDEND
USEDDISKSPACESTART
USEDDISKSPACEEND
USEDMEMORYSTART
USEDMEMORYEND
VALUESSTRINGSTART
VALUESSTRINGEND

VARSTART
VAREND
VARIABLESSTART
VARIABLESEND
VARIDXSTART
VARIDXEND
MATRIXVARSTART
MATRIXVAREND
MATRIXOBJSTART
MATRIXOBJEND
MATRIXCONSTART
MATRIXCONEND
FILENAMESTART
FILENAMEEND
FILENAMEEMPTY
FILENAMESTARTANDEND
FILESOURCESTART
FILESOURCEEND
FILESOURCEEMPTY
FILESOURCESTARTANDEND
FILEDESCRIPTIONSTART
FILEDESCRIPTIONEND
FILEDESCRIPTIONEMPTY
FILEDESCRIPTIONSTARTANDEND
FILECREATORSTART
FILECREATOREND
FILECREATOREMPTY
FILECREATORSTARTANDEND
FILELICENCESTART
FILELICENCEEND
FILELICENCEEMPTY
FILELICENCESTARTANDEND
MATRIXSTART
MATRIXEND
BASEMATRIXEND
BASEMATRIXSTART
BLOCKSTART
BLOCKEND
BLOCKSSTART
BLOCKSEND
EMPTYSHAPEATT
SHAPEATT

EMPTYSYMMETRYATT
SYMMETRYATT
EMPTYNEGATIVEPATTERNATT
NEGATIVEPATTERNATT
CONSTANTATT
NUMBEROFBLOCKSATT
NUMBEROFCOLUMNSATT
NUMBEROFROWSATT
NUMBEROFVALUESATT
COEFATT
BASEMATRIXIDXATT
TARGETMATRIXFIRSTROWATT
TARGETMATRIXFIRSTCOLATT
BASEMATRIXSTARTROWATT
BASEMATRIXSTARTCOLATT
BASEMATRIXENDROWATT
BASEMATRIXENDCOLATT
SCALARMULTIPLIERATT
EMPTYBASETRANSPOSEATT
BASETRANSPOSEATT
ELEMENTSSTART
ELEMENTSEND
CONSTANTELEMENTSSTART
CONSTANTELEMENTSEND
STARTVECTORSTART
STARTVECTOREND
NONZEROSSTART
NONZEROSSEND
INDEXESSTART
INDEXESEND
VALUESSTART
VALUESEND
VARREFERENCEELEMENTSSTART
VARREFERENCEELEMENTSEND
LINEARELEMENTSSTART
LINEARELEMENTSEND
GENERALELEMENTSSTART
GENERALELEMENTSEND
CONREFERENCEELEMENTSSTART
CONREFERENCEELEMENTSEND
VALUETYPEATT
OBJREFERENCEELEMENTSSTART

OBJREFERENCEELEMENTSEND
PATTERNELEMENTSSTART
PATTERNELEMENTSEND
TRANSFORMATIONSTART
TRANSFORMATIONEND
COLOFFSETSSTART
COLOFFSETSEND
ROWOFFSETSSTART
ROWOFFSETSEND
EMPTYROWMAJORATT
ROWMAJORATT
BLOCKROWIDXATT
BLOCKCOLIDXATT
DUMMY
NONLINEAREXPRESSIONSSTART
NONLINEAREXPRESSIONSEND
NUMBEROFNONLINEAREXPRESSIONS
NLSTART
NLEND
MATRIXEXPRESSIONSSTART
MATRIXEXPRESSIONSEND
NUMBEROFEXPR
EXPRSTART
EXPREND
NUMBEROFMATRIXTERMSATT
MATRIXTERMSTART
MATRIXTERMEND
POWERSTART
POWEREND
PLUSSTART
PLUSEND
MINUSSTART
MINUSEND
DIVIDESTART
DIVIDEEND
LNSTART
LNEND
SQRTSTART
SQRTEND
SUMSTART
SUMEND
PRODUCTSTART

PRODUCTEND
EXPSTART
EXPEND
NEGATESTART
NEGATEEND
IFSTART
IFEND
SQUARESTART
SQUAREEND
COSSTART
COSEND
SINSTART
SINEND
VARIABLESTART
VARIABLEEND
ABSSTART
ABSEND
ERFSTART
ERFEND
MAXSTART
MAXEND
ALLDIFFSTART
ALLDIFFEND
MINSTART
MINEND
ESTART
EEND
PISTART
PIEND
TIMESSTART
TIMESEND
NUMBERSTART
NUMBEREND
MATRIXDETERMINANTSTART
MATRIXDETERMINANTEND
MATRIXTRACESTART
MATRIXTRACEEND
MATRIXTOSCALARSTART
MATRIXTOSCALAREND
MATRIXDIAGONALSTART
MATRIXDIAGONALEND
MATRIXDOTTIMESSTART

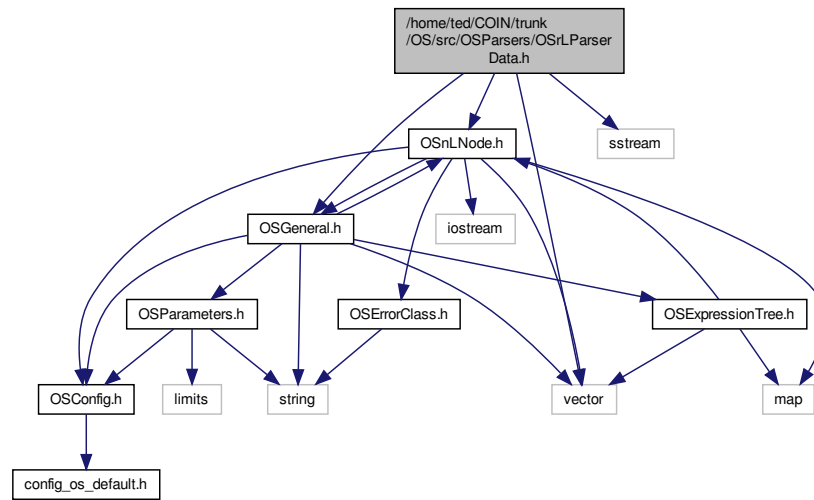
MATRIXDOTTIMESEND
MATRIXLOWERTRIANGLESTART
MATRIXLOWERTRIANGLEEND
MATRIXUPPERTRIANGLESTART
MATRIXUPPERTRIANGLEEND
MATRIXMERGESTART
MATRIXMERGEEND
MATRIXMINUSSTART
MATRIXMINUSEND
MATRIXNEGATESTART
MATRIXNEGATEEND
MATRIXPLUSSTART
MATRIXPLUSEND
MATRIXTIMESSTART
MATRIXTIMSEEND
MATRIXPRODUCTSTART
MATRIXPRODUCTEND
MATRIXSCALARTIMESSTART
MATRIXSCALARTIMSEEND
MATRIXSUBMATRIXATSTART
MATRIXSUBMATRIXATEND
MATRIXTRANSPOSESTART
MATRIXTRANSPOSEEND
MATRIXREFERENCESTART
MATRIXREFERENCEEND
IDENTITYMATRIXSTART
IDENTITYMATRIXEND
MATRIXINVERSESTART
MATRIXINVERSEEND
EMPTYINCLUDEDIAGONALATT
INCLUDEDIAGONALATT
IDATT

Definition at line 41 of file OSParseosrl.tab.hpp.

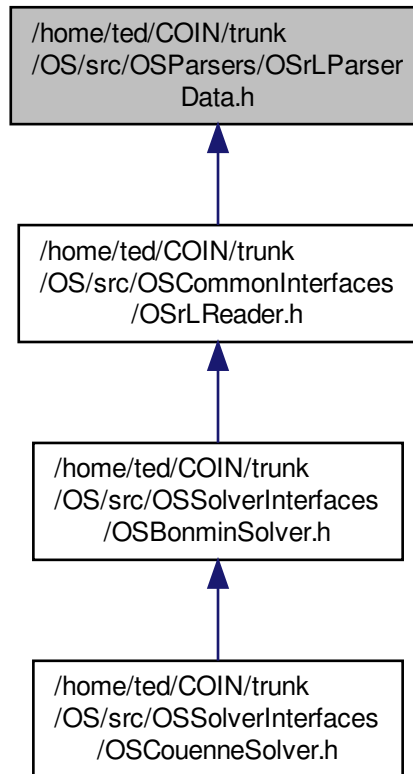
7.38 /home/ted/COIN/trunk/OS/src/OSParsers/OSrLParserData.h File Reference

```
#include "OSnLNode.h"  
#include "OSGeneral.h"  
#include <vector>  
#include <sstream>
```

Include dependency graph for OSrLParserData.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [OtherVariableResultStruct](#)
A structure to information about an [OtherVariableResult](#) element.
- class [OSrLParserData](#)
The [OSrLParserData](#) Class.

7.38.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

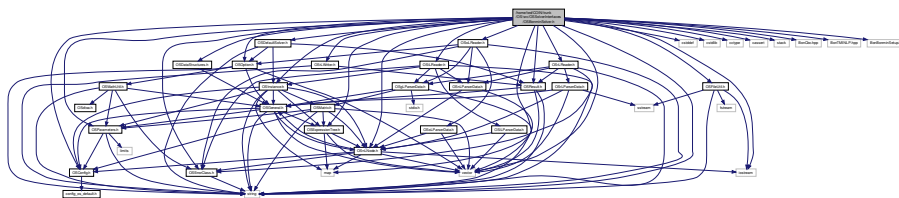
Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSrLParserData.h](#).

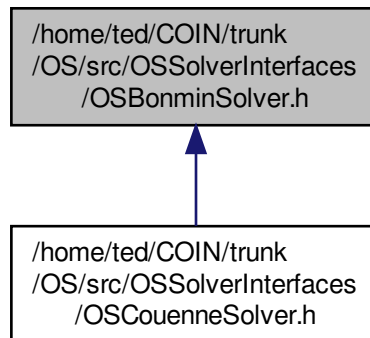
7.39 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSBonminSolver.h File Reference

```
#include "OSConfig.h"
#include "OSDefaultSolver.h"
#include "OSrLWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSiLReader.h"
#include "OSrLReader.h"
#include "OSoLReader.h"
#include "OSExpressionTree.h"
#include "OSDataStructures.h"
#include "OSFileUtil.h"
#include "OSErrorClass.h"
#include "OSResult.h"
#include "OSOption.h"
#include <cstdint>
#include <cstdlib>
#include <cctype>
#include <cassert>
#include <stack>
#include <string>
#include <iostream>
#include <vector>
#include <map>
#include "BonCbc.hpp"
#include "BonTMINLP.hpp"
#include "BonBonminSetup.hpp"
```

Include dependency graph for OSBonminSolver.h:



This graph shows which files directly or indirectly include this file:



Classes

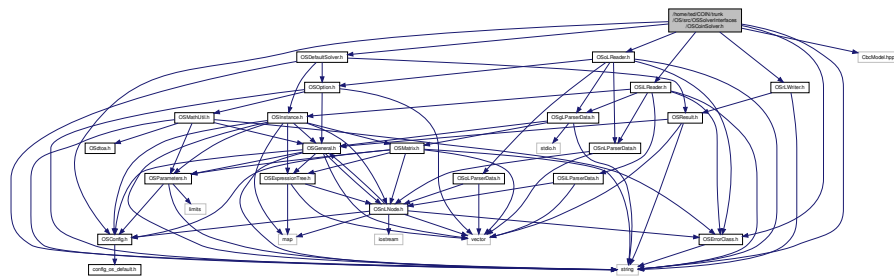
- class [BonminProblem](#)
- class [BonminSolver](#)

The [BonminSolver](#) class solves problems using *Ipopt*.

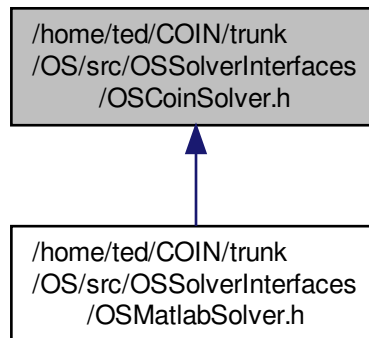
7.40 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCoinSolver.h File Reference

```
#include "OSConfig.h"
#include "OSDefaultSolver.h"
#include "OSrLWriter.h"
#include "OSErrorClass.h"
#include "OSiLReader.h"
#include "OSoLReader.h"
#include "CbcModel.hpp"
#include <string>
```

Include dependency graph for OSCoinSolver.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CoinSolver](#)

Implements a solve method for the [Coin](#) solvers.

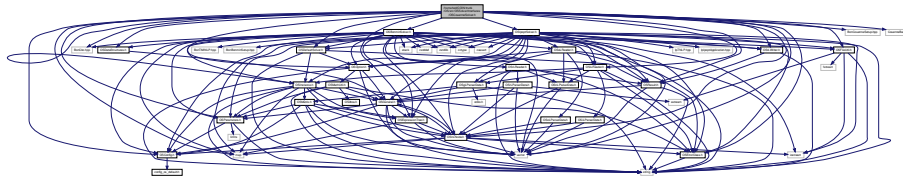
7.41 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCouenneSolver.h File Reference

```

#include "OSConfig.h"
#include "OSDefaultSolver.h"
#include "OSBonminSolver.h"
#include "OSIpoptSolver.h"
#include "OSrLWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSiLReader.h"
#include "OSExpressionTree.h"
#include "OSnLNode.h"
#include "OSDataStructures.h"
#include "OSFileUtil.h"
#include "OSErrorClass.h"
#include "OSResult.h"
#include "OSOption.h"
#include "BonCbc.hpp"
#include "BonCouenneSetup.hpp"
#include "CouenneBab.hpp"
#include <vector>
#include <map>

```

Include dependency graph for OSCouenneSolver.h:



Classes

- class [CouenneSolver](#)
The [CouenneSolver](#) class solves problems using *Ipopt*.

Namespaces

- namespace [Couenne](#)

7.41.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2011, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSCouenneSolver.h](#).

7.42 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSCsdpSolver.h File Reference

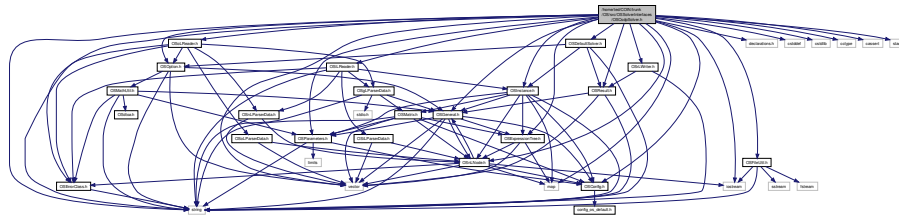
```
#include "OSConfig.h"
```

```

#include "OSDefaultSolver.h"
#include "OSrLWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSiLReader.h"
#include "OSoLReader.h"
#include "OSExpressionTree.h"
#include "OSGeneral.h"
#include "OSFileUtil.h"
#include "OSErrorClass.h"
#include "OSResult.h"
#include "OSOption.h"
#include "declarations.h"
#include <cstdlib>
#include <cstdliblib>
#include <cctype>
#include <cassert>
#include <stack>
#include <string>
#include <iostream>
#include <vector>
#include <map>

```

Include dependency graph for OSCsdpSolver.h:



Classes

- class [CsdpSolver](#)

The [CsdpSolver](#) class solves problems using Csdp.

7.42.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2014, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSCsdpSolver.h](#).

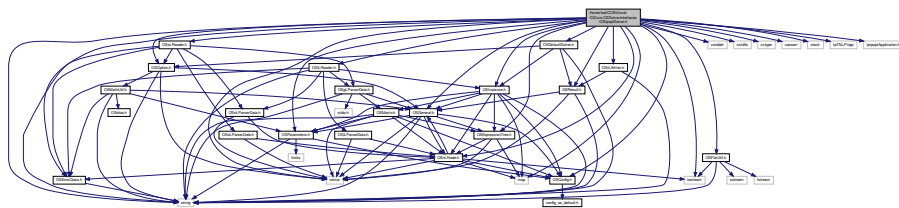
7.43 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSIpoptSolver.h File Reference

```

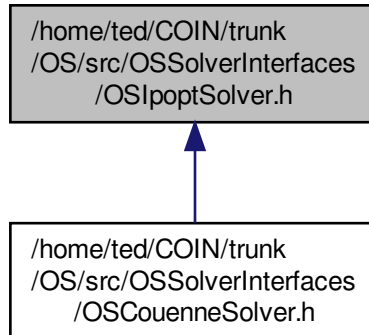
#include "OSConfig.h"
#include "OSDefaultSolver.h"
#include "OSrLWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSiLReader.h"
#include "OSoLReader.h"
#include "OSExpressionTree.h"
#include "OSGeneral.h"
#include "OSFileUtil.h"
#include "OSErrorClass.h"
#include "OSResult.h"
#include "OSOption.h"
#include <cstdlib>
#include <cstdliblib>
#include <cctype>
#include <cassert>
#include <stack>
#include <string>
#include <iostream>
#include <vector>
#include <map>
#include "IpTNLP.hpp"
#include "IpIpoptApplication.hpp"

```

Include dependency graph for OSIpoptSolver.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IpoptProblem](#)
- class [IpoptSolver](#)

The [IpoptSolver](#) class solves problems using *Ipopt*.

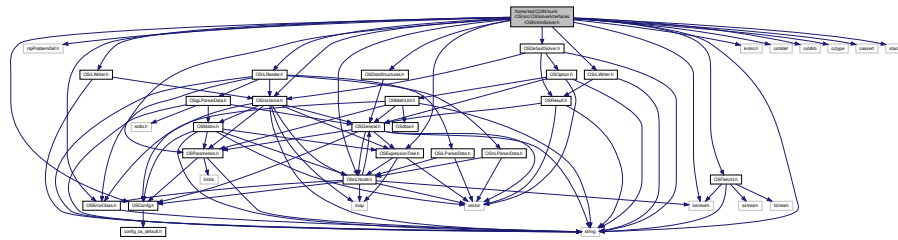
7.44 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSKnitroSolver.h File Reference

```

#include "OSConfig.h"
#include "nlpProblemDef.h"
#include "OSDefaultSolver.h"
#include "OSrLWriter.h"
#include "OSiLWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSiLReader.h"
#include "OSExpressionTree.h"
#include "OSDataStructures.h"
#include "OSFileUtil.h"
#include "OSErrorClass.h"
#include "knitro.h"
#include <cstddef>
#include <cstdlib>
#include <cctype>
#include <cassert>
#include <stack>
#include <string>
#include <iostream>

```

Include dependency graph for OSKnitroSolver.h:



Classes

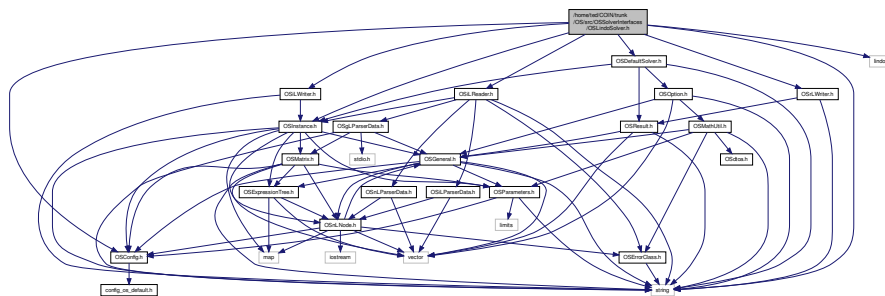
- class [KnitroProblem](#)
- class [KnitroSolver](#)

the [KnitroSolver](#) class solves problems using Knitro.

7.45 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSLindoSolver.h File Reference

```
#include "OSDefaultSolver.h"
#include "OSInstance.h"
#include "lindo.h"
#include "OSrLWriter.h"
#include "OSiLWriter.h"
#include "OSiLReader.h"
#include "OSConfig.h"
#include <string>
```

Include dependency graph for OSLindoSolver.h:



Classes

- class [LindoSolver](#)

the [LindoSolver](#) class solves problems using Lindo.

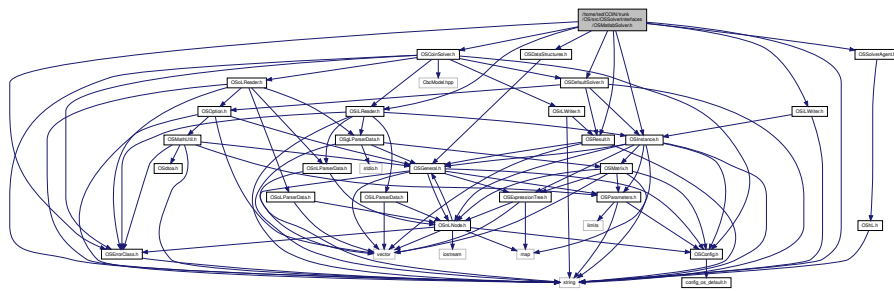
7.46 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSMatlabSolver.h File Reference

```

#include "OSCoinSolver.h"
#include "OSResult.h"
#include "OSILReader.h"
#include "OSILWriter.h"
#include "OSInstance.h"
#include "OSDefaultSolver.h"
#include "OSDataStructures.h"
#include "OSSolverAgent.h"
#include "OSErrorClass.h"
#include <string>

```

Include dependency graph for OSMatlabSolver.h:



Classes

- class [OSMatlab](#)
The *OSMatlab* Class.

7.47 /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OSRunSolver.h File Reference

```

#include "OSDefaultSolver.h"

```

- `std::string runSolver` (std::string solverName, std::string osol, `OSInstance` *osinstance)
This class is used to invoke a solver locally.
- `std::string runSolver` (std::string solverName, `OSOption` *osoption, std::string osil)
Alternate signature for this method.
- `std::string runSolver` (std::string solverName, std::string osol, std::string osil)
Alternate signature for this method.
- `std::string runSolver` (std::string solverName, `OSOption` *osoption, `OSInstance` *osinstance)
Alternate signature for this method.
- `DefaultSolver` * `selectSolver` (std::string solverName, `OSInstance` *osinstance)
A method to select the solver.

Author

Remarks

Definition in file [OSRunSolver.h](#).

7.47.2 Function Documentation

7.47.2.1 `std::string runSolver (std::string solverName, std::string osol, OSInstance * osinstance)`

This class is used to invoke a solver locally.

A wrapper around the solve() method

Parameters

<i>solverName</i> ,:	The name of the solver selected by the user If empty, a default solver is selected
<i>osol</i> ,:	A string containing the user options in osol format
<i>osinstance</i> ,:	A pointer to an OSInstance object containing the instance to be optimized

Returns

the solution (or error message) in OSrL format

7.47.2.2 `std::string runSolver (std::string solverName, OSOption * osoption, std::string osil)`

Alternate signature for this method.

Parameters

<i>solverName</i> ,:	The name of the solver selected by the user If empty, a default solver is selected
<i>osoption</i> ,:	A pointer to an OSOption object containing the options to be passed to the solver
<i>osil</i> ,:	A string containing the instance to be optimized

Returns

the solution (or error message) in OSrL format

7.47.2.3 `std::string runSolver (std::string solverName, std::string osol, std::string osil)`

Alternate signature for this method.

Parameters

<i>solverName</i> ,:	The name of the solver selected by the user If empty, a default solver is selected
<i>osol</i> ,:	A string containing the user options in osol format
<i>osil</i> ,:	A string containing the instance to be optimized

Returns

the solution (or error message) in OSrL format

7.47.2.4 `std::string runSolver (std::string solverName, OSOption * osoption, OSInstance * osinstance)`

Alternate signature for this method.

Parameters

<i>solverName</i> ,:	The name of the solver selected by the user If empty, a default solver is selected
<i>osoption</i> ,:	A pointer to an OSOption object containing the options to be passed to the solver
<i>osinstance</i> ,:	A pointer to an OSInstance object containing the instance to be optimized

Returns

the solution (or error message) in OSrL format

7.47.2.5 DefaultSolver* selectSolver (std::string solverName, OSInstance * osinstance)

A method to select the solver.

Parameters

<i>solverName</i> ,:	The name of the solver selected by the user. If empty, a default solver is selected based on the characteristics of the problem.
<i>osinstance</i> ,:	A pointer to an OSInstance object containing the instance to be optimized.

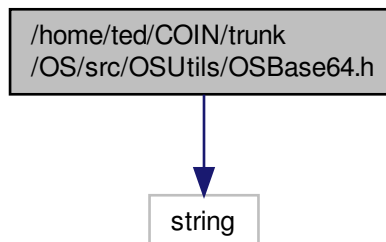
Returns

a pointer to the selected solver or NULL if no such solver exists on the system

7.48 /home/ted/COIN/trunk/OS/src/OSUtils/OSBase64.h File Reference

```
#include <string>
```

Include dependency graph for OSBase64.h:



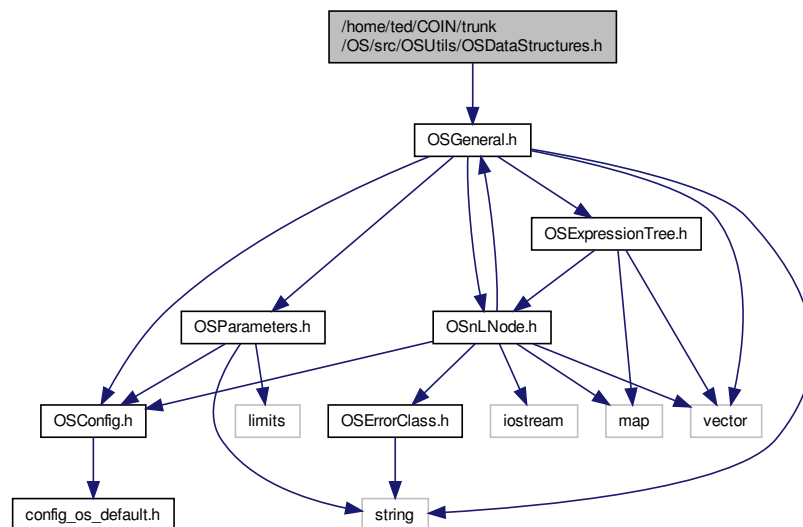
```
graph BT; A["/home/ted/COIN/trunk  
/OS/src/OSCommonInterfaces  
/OSgLWriter.h"] --> B["/home/ted/COIN/trunk  
/OS/src/OSUtils/OSBase64.h"]
```

/home/ted/COIN/trunk
/OS/src/OSCommonInterfaces
/OSgLWriter.h

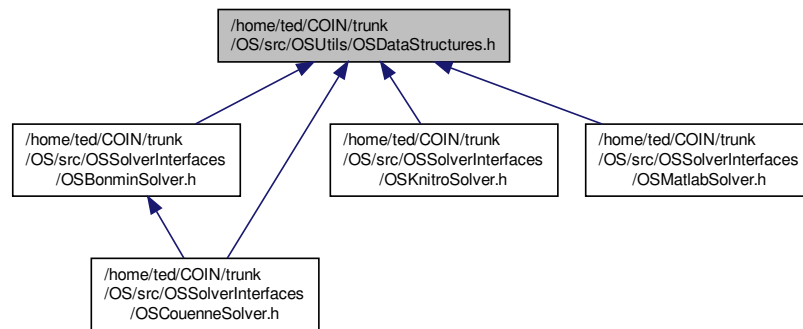
/home/ted/COIN/trunk
/OS/src/OSUtils/OSBase64.h

- class `Base64`
use this class to read and write data in base64.

```
#include "OSGeneral.h"
Include dependency graph for OSDataStructures.h:
```

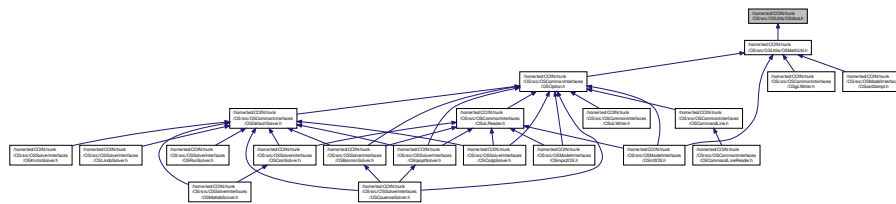


This graph shows which files directly or indirectly include this file:



7.50 /home/ted/COIN/trunk/OS/src/OSUtils/OSdtoa.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- double [os_strtod](#) (const char *str, char **strEnd)
- char * [os_dtoa](#) (double d, int mode, int ndigits, int *decpt, int *sign, char **rve)
- void [os_freedtoa](#) (char *s)

7.50.1 Function Documentation

7.50.1.1 double [os_strtod](#) (const char * *str*, char ** *strEnd*)

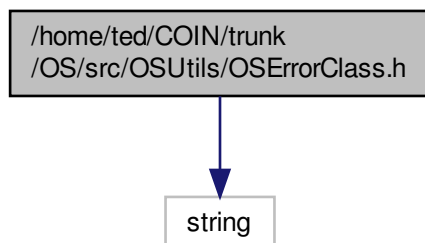
7.50.1.2 char* [os_dtoa](#) (double *d*, int *mode*, int *ndigits*, int * *decpt*, int * *sign*, char ** *rve*)

7.50.1.3 void [os_freedtoa](#) (char * *s*)

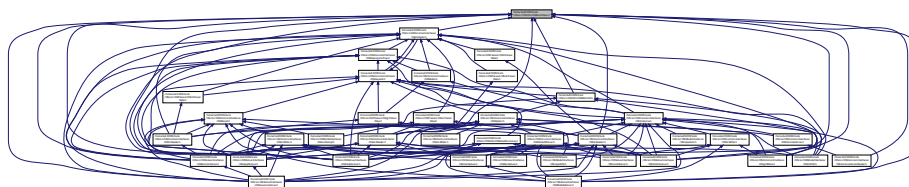
7.51 /home/ted/COIN/trunk/OS/src/OSUtils/OSErrorClass.h File Reference

```
#include <string>
```

Include dependency graph for OSErrorClass.h:



This graph shows which files directly or indirectly include this file:



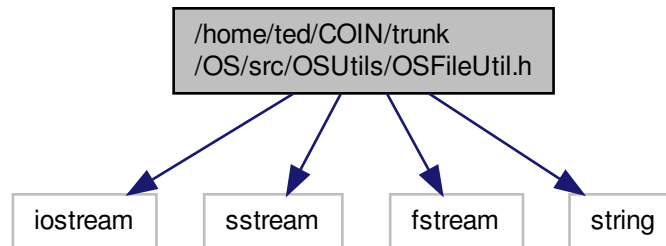
Classes

- class [ErrorClass](#)
used for throwing exceptions.

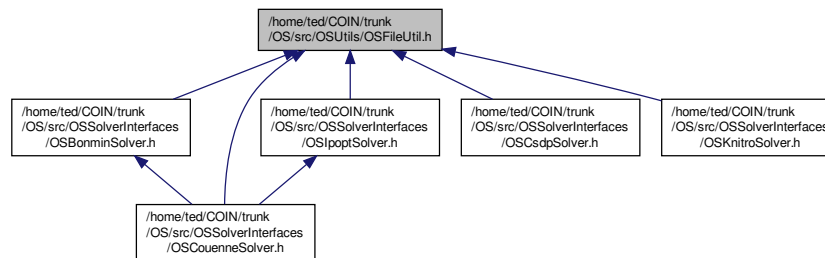
7.52 /home/ted/COIN/trunk/OS/src/OSUtils/OSFileUtil.h File Reference

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
```

Include dependency graph for OSFileUtil.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [FileUtil](#)
class used to make it easy to read and write files.

7.53 /home/ted/COIN/trunk/OS/src/OSUtils/OSMathUtil.h File Reference

```

#include "OSParameters.h"
#include "OSdtoa.h"
#include "OSErrorClass.h"
#include "OSGeneral.h"
#include <string>

```

[illegible]

- class **MathUtil**
this class has routines for linear algebra.

- double [os_strtod_wrap](#) (const char *str, char **strEnd)
- void [getMultIncr](#) (int *i, int *mult, int *incr, int size, int defaultIncr)
getMultIncr
- void [getMultIncr](#) (double *a, int *mult, double *incr, int size)
getMultIncr
- int [getMult](#) (int *i, int size)
getMult
- int [getMult](#) (double *a, int size)
getMult
- double [OSRand](#) ()
OSRand()
- double [OSiRand](#) (int iMin, int iMax)
OSiRand(int iMin, int iMax)

7.53.1 Function Documentation

7.53.1.1 `double os_strtod_wrap (const char * str, char ** strEnd)`7.53.1.2 `void getMultIncr (int * i, int * mult, int * incr, int size, int defaultIncr)` `[inline]`

getMultIncr

Identify the next run in an integer array

Parameters

<i>i</i>	holds a pointer to the array to be processed.
<i>mult</i>	holds the length of the run. This parameter is passed by reference
<i>incr</i>	holds the increment. This parameter is also passed by reference
<i>size</i>	holds the number of elements in the array. This parameter is passed by value
<i>defaultIncr</i>	holds the default value for incr from the schema file. Using just <el mult="..." saves space whenever a run of two or more elements has been encountered, whereas <el mult="..." incr="..." saves space only for runs of three or more elements. Thus the defaultIncr must be treated specially (and it might change from one schema element to the next).

Definition at line 166 of file OSMathUtil.h.

7.53.1.3 `void getMultIncr (double * a, int * mult, double * incr, int size)` `[inline]`

getMultIncr

Identify the next run in an array of type double.

Parameters

<i>i</i>	holds a pointer to the array to be processed.
<i>mult</i>	holds the length of the run. This parameter is passed by reference
<i>incr</i>	holds the increment. This parameter is also passed by reference
<i>size</i>	holds the number of elements in the array. This parameter is passed by value

Definition at line 204 of file OSMathUtil.h.

7.53.1.4 `int getMult (int * i, int size)` `[inline]`

getMult

Identify the number of duplicates at the start of an integer array

Parameters

<i>i</i>	holds a pointer to the array to be processed.
<i>size</i>	holds the number of elements in the array.

Returns

the length of the run.

Definition at line 244 of file OSMathUtil.h.

7.53.1.5 `int getMult (double * a, int size)` `[inline]`

getMult

Identify the number of duplicates at the start of an array of type double

Parameters

<i>i</i>	holds a pointer to the array to be processed.
<i>size</i>	holds the number of elements in the array.

Returns

the length of the run.

Definition at line 272 of file OSMathUtil.h.

7.53.1.6 double OSRand ()

[OSRand\(\)](#)

Returns

a uniformly distributed random number between 0 and 1 (inclusive) The random number generator used, rand(), is not very good and should be replaced by a serious random number generator for serious work.

7.53.1.7 double OSiRand (int iMin, int iMax)

[OSiRand\(int iMin, int iMax\)](#)

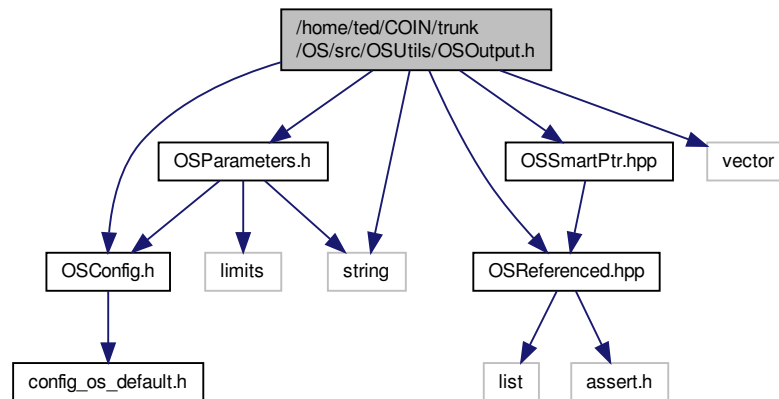
Returns

a uniformly distributed random integer between iMin and iMax (inclusive) The random number generator used, rand(), is not very good and should be replaced by a serious random number generator for serious work.

7.54 /home/ted/COIN/trunk/OS/src/OSUtils/OSOutput.h File Reference

```
#include "OSConfig.h"
#include "OSParameters.h"
#include "OSReferenced.hpp"
#include "OSSmartPtr.hpp"
#include <string>
#include <vector>
```

Include dependency graph for OSOutput.h:



Classes

- class [OSOutputChannel](#)
a class that holds information about one output channel (file, device, stream, peripheral, etc.)
- class [OSOutput](#)
This class handles all the output from `OSSolverService`, `OSAmplClient` and other executables derived from them.

Variables

- const [OSSmartPtr< OSOutput >](#) `osoutput`

7.54.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2012-2013, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSOutput.h](#).

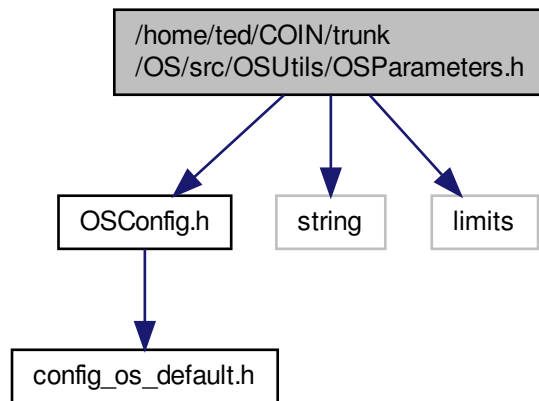
7.54.2 Variable Documentation

7.54.2.1 const [OSSmartPtr<OSOutput>](#) `osoutput`

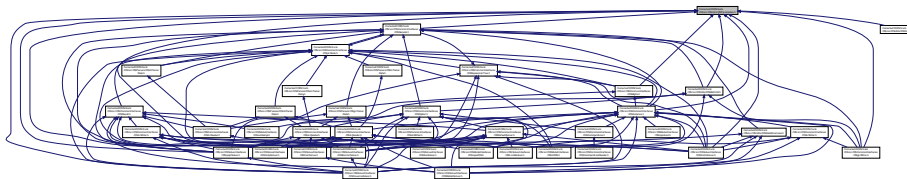
7.55 /home/ted/COIN/trunk/OS/src/OSUtils/OSParameters.h File Reference

```
#include "OSConfig.h"
#include <string>
#include <limits>
```

Include dependency graph for OSParameters.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define OS_PLUS 1001`
- `#define OS_SUM 1002`
- `#define OS_MINUS 1003`
- `#define OS_NEGATE 1004`
- `#define OS_TIMES 1005`
- `#define OS_DIVIDE 1006`
- `#define OS_POWER 1009`
- `#define OS_PRODUCT 1010`
- `#define OS_ABS 2001`
- `#define OS_SQUARE 2005`
- `#define OS_SQRT 2006`
- `#define OS_LN 2007`

- #define `OS_EXP` 2010
 - #define `OS_ERF` 2023
 - #define `OS_SIN` 3001
 - #define `OS_COS` 3002
 - #define `OS_MIN` 4010
 - #define `OS_MAX` 4011
 - #define `OS_NUMBER` 5001
 - #define `OS_PI` 5003
 - #define `OS_E` 5004
 - #define `OS_VARIABLE` 6001
 - #define `OS_IF` 7001
 - #define `OS_ALLDIFF` 7016
 - #define `OS_MATRIX_DETERMINANT` 8001
 - #define `OS_MATRIX_TRACE` 8002
 - #define `OS_MATRIX_TO_SCALAR` 8003
 - #define `OS_MATRIX_PLUS` 8501
 - #define `OS_MATRIX_SUM` 8502
 - #define `OS_MATRIX_MINUS` 8503
 - #define `OS_MATRIX_NEGATE` 8504
 - #define `OS_MATRIX_TIMES` 8505
 - #define `OS_MATRIX_PRODUCT` 8506
 - #define `OS_MATRIX_INVERSE` 8510
 - #define `OS_MATRIX_TRANSPOSE` 8515
 - #define `OS_MATRIX_SCALARTIMES` 8518
 - #define `OS_MATRIX_DOTTIMES` 8520
 - #define `OS_IDENTITY_MATRIX` 8536
 - #define `OS_MATRIX_LOWERTRIANGLE` 8537
 - #define `OS_MATRIX_UPPERTRIANGLE` 8538
 - #define `OS_MATRIX_DIAGONAL` 8539
 - #define `OS_DIAGONAL_MATRIX_FROM_VECTOR` 8540
 - #define `OS_MATRIX_REFERENCE` 8541
 - #define `OS_MATRIX_SUBMATRIX_AT` 8544
 - #define `OS_MATRIX_VAR` 8601
 - #define `OS_MATRIX_OBJ` 8602
 - #define `OS_MATRIX_CON` 8603
 - #define `OS_E_VALUE` $\exp(1.0)$
 - #define `OS_PI_VALUE` $2*\text{asin}(1.0)$
 - #define `OS_SCHEMA_VERSION` "2.0"
 - #define `OS_NEAR_EQUAL` 1e-2
- we use `OS_NEAR_EQUAL` in `unitTest` to see if we are close to the optimal obj value*
- #define `OS_EPS` 1e-12
 - #define `DEFAULT_OUTPUT_LEVEL` `ENUM_OUTPUT_LEVEL_error`

Enumerations

- enum `ENUM_OUTPUT_LEVEL` {
`ENUM_OUTPUT_LEVEL_always` = 0, `ENUM_OUTPUT_LEVEL_error`, `ENUM_OUTPUT_LEVEL_summary`, `ENUM_OUTPUT_LEVEL_warning`,
`ENUM_OUTPUT_LEVEL_info`, `ENUM_OUTPUT_LEVEL_debug`, `ENUM_OUTPUT_LEVEL_trace`, `ENUM_OUTPUT_LEVEL_detailed_trace`,
`ENUM_OUTPUT_LEVEL_NUMBER_OF_LEVELS` }

Enumeration for the different verbosity levels that can be used in producing output.

- enum `ENUM_OUTPUT_AREA` {
`ENUM_OUTPUT_AREA_main` = 1, `ENUM_OUTPUT_AREA_OSAgent`, `ENUM_OUTPUT_AREA_Command_line_parser`, `ENUM_OUTPUT_AREA_OSiL_parser`,
`ENUM_OUTPUT_AREA_OSoL_parser`, `ENUM_OUTPUT_AREA_OSrL_parser`, `ENUM_OUTPUT_AREA_OSGeneral`, `ENUM_OUTPUT_AREA_OSInstance`,
`ENUM_OUTPUT_AREA_OSOOption`, `ENUM_OUTPUT_AREA_OSResult`, `ENUM_OUTPUT_AREA_OSMatrix`,
`ENUM_OUTPUT_AREA_OSiLwriter`,
`ENUM_OUTPUT_AREA_OSoLwriter`, `ENUM_OUTPUT_AREA_OSrLwriter`, `ENUM_OUTPUT_AREA_OSMODELInterfaces`, `ENUM_OUTPUT_AREA_OSSolverInterfaces`,
`ENUM_OUTPUT_AREA_OSUtils`, `ENUM_OUTPUT_AREA_NUMBER_OF_AREAS` }

Enumeration for the different areas that can produce output.

- enum `ENUM_CPUSPEEDUNIT` {
`ENUM_CPUSPEEDUNIT_hertz` = 1, `ENUM_CPUSPEEDUNIT_kilohertz`, `ENUM_CPUSPEEDUNIT_megahertz`,
`ENUM_CPUSPEEDUNIT_gigahertz`,
`ENUM_CPUSPEEDUNIT_terahertz`, `ENUM_CPUSPEEDUNIT_flops`, `ENUM_CPUSPEEDUNIT_kiloflops`, `ENUM_CPUSPEEDUNIT_megaflops`,
`ENUM_CPUSPEEDUNIT_gigaflops`, `ENUM_CPUSPEEDUNIT_teraflops`, `ENUM_CPUSPEEDUNIT_petaflops` }
- enum `ENUM_STORAGEUNIT` {
`ENUM_STORAGEUNIT_byte` = 1, `ENUM_STORAGEUNIT_kilobyte`, `ENUM_STORAGEUNIT_megabyte`, `ENUM_STORAGEUNIT_gigabyte`,
`ENUM_STORAGEUNIT_terabyte`, `ENUM_STORAGEUNIT_petabyte`, `ENUM_STORAGEUNIT_exabyte`, `ENUM_STORAGEUNIT_zettabyte`,
`ENUM_STORAGEUNIT_yottabyte` }
- enum `ENUM_TIMEUNIT` {
`ENUM_TIMEUNIT_tick` = 1, `ENUM_TIMEUNIT_millisecond`, `ENUM_TIMEUNIT_second`, `ENUM_TIMEUNIT_minute`,
`ENUM_TIMEUNIT_hour`, `ENUM_TIMEUNIT_day`, `ENUM_TIMEUNIT_week`, `ENUM_TIMEUNIT_month`,
`ENUM_TIMEUNIT_year` }
- enum `ENUM_TIMETYPE` { `ENUM_TIMETYPE_cpuTime` = 1, `ENUM_TIMETYPE_elapsedTime`, `ENUM_TIMETYPE_other` }
- enum `ENUM_TIMECATEGORY` {
`ENUM_TIMECATEGORY_total` = 1, `ENUM_TIMECATEGORY_input`, `ENUM_TIMECATEGORY_preprocessing`,
`ENUM_TIMECATEGORY_optimization`,
`ENUM_TIMECATEGORY_postprocessing`, `ENUM_TIMECATEGORY_output`, `ENUM_TIMECATEGORY_other` }
- enum `ENUM_LOCATIONTYPE` { `ENUM_LOCATIONTYPE_local` = 1, `ENUM_LOCATIONTYPE_http`, `ENUM_LOCATIONTYPE_ftp` }
- enum `ENUM_TRANSPORT_TYPE` {
`ENUM_TRANSPORT_TYPE_osp` = 1, `ENUM_TRANSPORT_TYPE_http`, `ENUM_TRANSPORT_TYPE_smtp`,
`ENUM_TRANSPORT_TYPE_ftp`,
`ENUM_TRANSPORT_TYPE_other` }
- enum `ENUM_SERVICE_TYPE` {
`ENUM_SERVICE_TYPE_analyzer` = 1, `ENUM_SERVICE_TYPE_solver`, `ENUM_SERVICE_TYPE_scheduler`, `ENUM_SERVICE_TYPE_modeler`,
`ENUM_SERVICE_TYPE_registry`, `ENUM_SERVICE_TYPE_agent`, `ENUM_SERVICE_TYPE_simulations` }

- enum `ENUM_GENERAL_RESULT_STATUS` { `ENUM_GENERAL_RESULT_STATUS_error` = 1, `ENUM_GENERAL_RESULT_STATUS_warning`, `ENUM_GENERAL_RESULT_STATUS_normal` }
- enum `ENUM_SYSTEM_CURRENT_STATE` { `ENUM_SYSTEM_CURRENT_STATE_busy` = 1, `ENUM_SYSTEM_CURRENT_STATE_busyButAccepting`, `ENUM_SYSTEM_CURRENT_STATE_idle`, `ENUM_SYSTEM_CURRENT_STATE_idleButNotAccepting`, `ENUM_SYSTEM_CURRENT_STATE_noResponse` }
- enum `ENUM_JOB_STATUS` { `ENUM_JOB_STATUS_waiting` = 1, `ENUM_JOB_STATUS_running`, `ENUM_JOB_STATUS_killed`, `ENUM_JOB_STATUS_finished`, `ENUM_JOB_STATUS_unknown` }
- enum `ENUM_BASIS_STATUS` { `ENUM_BASIS_STATUS_basic` = 0, `ENUM_BASIS_STATUS_atLower`, `ENUM_BASIS_STATUS_atUpper`, `ENUM_BASIS_STATUS_atEquality`, `ENUM_BASIS_STATUS_isFree`, `ENUM_BASIS_STATUS_superbasic`, `ENUM_BASIS_STATUS_unknown`, `ENUM_BASIS_STATUS_NUMBER_OF_STATES` }

Enumeration for the different states that can be used in representing a basis The last state, `ENUM_BASIS_STATUS_NUMBER_OF_STATES`, is used only to record the number of states, which makes it easier to convert between different representations.

- enum `ENUM_SOLUTION_STATUS` { `ENUM_SOLUTION_STATUS_unbounded` = 1, `ENUM_SOLUTION_STATUS_globallyOptimal`, `ENUM_SOLUTION_STATUS_locallyOptimal`, `ENUM_SOLUTION_STATUS_optimal`, `ENUM_SOLUTION_STATUS_bestSoFar`, `ENUM_SOLUTION_STATUS_feasible`, `ENUM_SOLUTION_STATUS_infeasible`, `ENUM_SOLUTION_STATUS_unsure`, `ENUM_SOLUTION_STATUS_error`, `ENUM_SOLUTION_STATUS_other` }
- enum `ENUM_SOLUTION_SUBSTATUSTYPE` { `ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByLimit` = 1, `ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByBounds`, `ENUM_SOLUTION_SUBSTATUSTYPE_other` }
- enum `ENUM_PROBLEM_COMPONENT` { `ENUM_PROBLEM_COMPONENT_variables` = 1, `ENUM_PROBLEM_COMPONENT_objectives`, `ENUM_PROBLEM_COMPONENT_constraints` }
- enum `ENUM_VARTYPE` { `ENUM_VARTYPE_continuous` = 1, `ENUM_VARTYPE_binary`, `ENUM_VARTYPE_integer`, `ENUM_VARTYPE_string`, `ENUM_VARTYPE_semicontinuous`, `ENUM_VARTYPE_semiinteger` }
- enum `ENUM_PATHPAIR` { `ENUM_PATHPAIR_input_dir` = 1, `ENUM_PATHPAIR_input_file`, `ENUM_PATHPAIR_output_file`, `ENUM_PATHPAIR_output_dir` }
- enum `ENUM_MATRIX_TYPE` { `ENUM_MATRIX_TYPE_zero` = 1, `ENUM_MATRIX_TYPE_constant` = 10, `ENUM_MATRIX_TYPE_varref`, `ENUM_MATRIX_TYPE_linear`, `ENUM_MATRIX_TYPE_quadratic`, `ENUM_MATRIX_TYPE_general`, `ENUM_MATRIX_TYPE_conref` = 20, `ENUM_MATRIX_TYPE_objref`, `ENUM_MATRIX_TYPE_mixedref`, `ENUM_MATRIX_TYPE_jumbled` = 30, `ENUM_MATRIX_TYPE_unknown` = 99 }

An enum to track the many different types of values that a matrix can contain Note that these types are partially ordered, which makes it easier to infer a matrix's type from the types of its constructors.

- enum `ENUM_CONREFERENCE_VALUETYPE` { `ENUM_CONREFERENCE_VALUETYPE_value` = 1, `ENUM_CONREFERENCE_VALUETYPE_status`, `ENUM_CONREFERENCE_VALUETYPE_surplus`, `ENUM_CONREFERENCE_VALUETYPE_shortage` }
- An enum to track the type of value contained in a reference to a constraint.*
- enum `ENUM_MATRIX_SYMMETRY` { `ENUM_MATRIX_SYMMETRY_none` = 1, `ENUM_MATRIX_SYMMETRY_upper`, `ENUM_MATRIX_SYMMETRY_lower`, `ENUM_MATRIX_SYMMETRY_skewUpper`, `ENUM_MATRIX_SYMMETRY_skewLower`, `ENUM_MATRIX_SYMMETRY_HermitianLower`, `ENUM_MATRIX_SYMMETRY_HermitianUpper` }

- enum `ENUM_MATRIX_CONSTRUCTOR_TYPE` {
`ENUM_MATRIX_CONSTRUCTOR_TYPE_unknown` = 0, `ENUM_MATRIX_CONSTRUCTOR_TYPE_baseMatrix`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_constantElements`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_varRefElements`,
`ENUM_MATRIX_CONSTRUCTOR_TYPE_linearElements`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_generalElements`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_objRefElements`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_conRefElements`,
`ENUM_MATRIX_CONSTRUCTOR_TYPE_rowRefElements`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_transformation`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_blocks`, `ENUM_MATRIX_CONSTRUCTOR_TYPE_block`,
`ENUM_MATRIX_CONSTRUCTOR_TYPE_matrix` }
- enum `ENUM_COMBINE_ARRAYS` { `ENUM_COMBINE_ARRAYS_replace`, `ENUM_COMBINE_ARRAYS_merge`, `ENUM_COMBINE_ARRAYS_ignore`, `ENUM_COMBINE_ARRAYS_throw` }
An enum to streamline set() methods of vectors.
- enum `ENUM_NL_EXPR_SHAPE` {
`ENUM_NL_EXPR_SHAPE_general` = 1, `ENUM_NL_EXPR_SHAPE_convex`, `ENUM_NL_EXPR_SHAPE_quadratic`, `ENUM_NL_EXPR_SHAPE_linear`,
`ENUM_NL_EXPR_SHAPE_constant` }
- enum `ENUM_CONE_TYPE` {
`ENUM_CONE_TYPE_nonnegative` = 1, `ENUM_CONE_TYPE_nonpositive`, `ENUM_CONE_TYPE_orthant`, `ENUM_CONE_TYPE_polyhedral`,
`ENUM_CONE_TYPE_quadratic`, `ENUM_CONE_TYPE_rotatedQuadratic`, `ENUM_CONE_TYPE_normed`, `ENUM_CONE_TYPE_rotatedNormed`,
`ENUM_CONE_TYPE_semidefinite`, `ENUM_CONE_TYPE_copositiveMatrices`, `ENUM_CONE_TYPE_completelyPositiveMatrices`, `ENUM_CONE_TYPE_hyperbolicity`,
`ENUM_CONE_TYPE_sumOfSquaresPolynomials`, `ENUM_CONE_TYPE_nonnegativePolynomials`, `ENUM_CONE_TYPE_moments`, `ENUM_CONE_TYPE_product`,
`ENUM_CONE_TYPE_intersection`, `ENUM_CONE_TYPE_dual`, `ENUM_CONE_TYPE_polar`, `ENUM_CONE_TYPE_unknown` }

Functions

- bool `OSIsnan` (double x)
checks whether a given double is NaN
- double `OSNaN` ()
returns the value for NaN used in OS
- std::string `OSgetVersionInfo` ()
- int `returnCPUSpeedUnit` (std::string unit)
- bool `verifyCPUSpeedUnit` (std::string unit)
- int `returnStorageUnit` (std::string unit)
- bool `verifyStorageUnit` (std::string unit)
- int `returnTimeUnit` (std::string unit)
- bool `verifyTimeUnit` (std::string unit)
- int `returnTimeType` (std::string type)
- bool `verifyTimeType` (std::string type)
- int `returnTimeCategory` (std::string category)
- bool `verifyTimeCategory` (std::string category)
- int `returnLocationType` (std::string type)
- bool `verifyLocationType` (std::string type)
- int `returnTransportType` (std::string type)
- bool `verifyTransportType` (std::string type)

- int [returnServiceType](#) (std::string type)
- bool [verifyServiceType](#) (std::string type)
- int [returnGeneralResultStatus](#) (std::string status)
- bool [verifyGeneralResultStatus](#) (std::string status)
- int [returnSystemCurrentState](#) (std::string status)
- bool [verifySystemCurrentState](#) (std::string status)
- int [returnJobStatus](#) (std::string status)
- bool [verifyJobStatus](#) (std::string status)
- int [returnBasisStatus](#) (std::string status)
- bool [verifyBasisStatus](#) (std::string status)
- std::string [returnBasisStatusString](#) (ENUM_BASIS_STATUS status)
- int [returnSolutionStatus](#) (std::string status)
- bool [verifySolutionStatus](#) (std::string status)
- int [returnSolutionSubstatusType](#) (std::string type)
- bool [verifySolutionSubstatusType](#) (std::string type)
- int [returnVarType](#) (char vt)
- bool [verifyVarType](#) (char vt)
- int [returnMatrixType](#) (std::string type)
- std::string [returnMatrixTypeString](#) (ENUM_MATRIX_TYPE type)
- bool [verifyMatrixType](#) (std::string type)
- [ENUM_MATRIX_TYPE](#) [mergeMatrixType](#) (ENUM_MATRIX_TYPE type1, ENUM_MATRIX_TYPE type2)
A function to merge two matrix types so we can infer the type of a matrix recursively.
- std::string [returnConReferenceValueTypeString](#) (ENUM_CONREFERENCE_VALUETYPE valueType)
- int [returnConReferenceValueType](#) (std::string valueType)
- bool [verifyConReferenceValueType](#) (std::string valueType)
- std::string [returnMatrixSymmetryString](#) (ENUM_MATRIX_SYMMETRY symmetry)
- int [returnMatrixSymmetry](#) (std::string symmetry)
- bool [verifyMatrixSymmetry](#) (std::string symmetry)
- int [returnMatrixConstructorType](#) (std::string cType)
- bool [verifyMatrixConstructorType](#) (std::string type)
- int [returnNIEExprShape](#) (std::string shape)
- std::string [returnExprShapeString](#) (ENUM_NL_EXPR_SHAPE shape)
- bool [verifyNIEExprShape](#) (std::string shape)
- int [returnConeType](#) (std::string type)
- bool [verifyConeType](#) (std::string type)

Variables

- const double [OSDBL_MAX](#) = std::numeric_limits<double>::max()
- const int [OSINT_MAX](#) = std::numeric_limits<int>::max()

7.55.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2015, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSParameters.h](#).

7.55.2 Macro Definition Documentation

7.55.2.1 `#define OS_PLUS 1001`

Definition at line 27 of file OSParameters.h.

7.55.2.2 `#define OS_SUM 1002`

Definition at line 28 of file OSParameters.h.

7.55.2.3 `#define OS_MINUS 1003`

Definition at line 29 of file OSParameters.h.

7.55.2.4 `#define OS_NEGATE 1004`

Definition at line 30 of file OSParameters.h.

7.55.2.5 `#define OS_TIMES 1005`

Definition at line 31 of file OSParameters.h.

7.55.2.6 `#define OS_DIVIDE 1006`

Definition at line 32 of file OSParameters.h.

7.55.2.7 `#define OS_POWER 1009`

Definition at line 33 of file OSParameters.h.

7.55.2.8 `#define OS_PRODUCT 1010`

Definition at line 34 of file OSParameters.h.

7.55.2.9 `#define OS_ABS 2001`

Definition at line 35 of file OSParameters.h.

7.55.2.10 `#define OS_SQUARE 2005`

Definition at line 36 of file OSParameters.h.

7.55.2.11 `#define OS_SQRT 2006`

Definition at line 37 of file OSParameters.h.

7.55.2.12 `#define OS_LN 2007`

Definition at line 38 of file OSParameters.h.

7.55.2.13 `#define OS_EXP 2010`

Definition at line 39 of file OSParameters.h.

7.55.2.14 `#define OS_ERF 2023`

Definition at line 40 of file OSParameters.h.

7.55.2.15 #define OS_SIN 3001

Definition at line 41 of file OSParameters.h.

7.55.2.16 #define OS_COS 3002

Definition at line 42 of file OSParameters.h.

7.55.2.17 #define OS_MIN 4010

Definition at line 43 of file OSParameters.h.

7.55.2.18 #define OS_MAX 4011

Definition at line 44 of file OSParameters.h.

7.55.2.19 #define OS_NUMBER 5001

Definition at line 45 of file OSParameters.h.

7.55.2.20 #define OS_PI 5003

Definition at line 46 of file OSParameters.h.

7.55.2.21 #define OS_E 5004

Definition at line 47 of file OSParameters.h.

7.55.2.22 #define OS_VARIABLE 6001

Definition at line 48 of file OSParameters.h.

7.55.2.23 #define OS_IF 7001

Definition at line 49 of file OSParameters.h.

7.55.2.24 #define OS_ALLDIFF 7016

Definition at line 50 of file OSParameters.h.

7.55.2.25 #define OS_MATRIX_DETERMINANT 8001

Definition at line 52 of file OSParameters.h.

7.55.2.26 #define OS_MATRIX_TRACE 8002

Definition at line 53 of file OSParameters.h.

7.55.2.27 #define OS_MATRIX_TO_SCALAR 8003

Definition at line 54 of file OSParameters.h.

7.55.2.28 #define OS_MATRIX_PLUS 8501

Definition at line 57 of file OSParameters.h.

7.55.2.29 **#define OS_MATRIX_SUM 8502**

Definition at line 58 of file OSParameters.h.

7.55.2.30 **#define OS_MATRIX_MINUS 8503**

Definition at line 59 of file OSParameters.h.

7.55.2.31 **#define OS_MATRIX_NEGATE 8504**

Definition at line 60 of file OSParameters.h.

7.55.2.32 **#define OS_MATRIX_TIMES 8505**

Definition at line 61 of file OSParameters.h.

7.55.2.33 **#define OS_MATRIX_PRODUCT 8506**

Definition at line 62 of file OSParameters.h.

7.55.2.34 **#define OS_MATRIX_INVERSE 8510**

Definition at line 63 of file OSParameters.h.

7.55.2.35 **#define OS_MATRIX_TRANSPOSE 8515**

Definition at line 64 of file OSParameters.h.

7.55.2.36 **#define OS_MATRIX_SCALARTIMES 8518**

Definition at line 65 of file OSParameters.h.

7.55.2.37 **#define OS_MATRIX_DOTTIMES 8520**

Definition at line 66 of file OSParameters.h.

7.55.2.38 **#define OS_IDENTITY_MATRIX 8536**

Definition at line 67 of file OSParameters.h.

7.55.2.39 **#define OS_MATRIX_LOWERTRIANGLE 8537**

Definition at line 68 of file OSParameters.h.

7.55.2.40 **#define OS_MATRIX_UPPERTRIANGLE 8538**

Definition at line 69 of file OSParameters.h.

7.55.2.41 **#define OS_MATRIX_DIAGONAL 8539**

Definition at line 70 of file OSParameters.h.

7.55.2.42 **#define OS_DIAGONAL_MATRIX_FROM_VECTOR 8540**

Definition at line 71 of file OSParameters.h.

7.55.2.43 #define OS_MATRIX_REFERENCE 8541

Definition at line 72 of file OSParameters.h.

7.55.2.44 #define OS_MATRIX_SUBMATRIX_AT 8544

Definition at line 73 of file OSParameters.h.

7.55.2.45 #define OS_MATRIX_VAR 8601

Definition at line 74 of file OSParameters.h.

7.55.2.46 #define OS_MATRIX_OBJ 8602

Definition at line 75 of file OSParameters.h.

7.55.2.47 #define OS_MATRIX_CON 8603

Definition at line 76 of file OSParameters.h.

7.55.2.48 #define OS_E_VALUE exp(1.0)

Definition at line 80 of file OSParameters.h.

7.55.2.49 #define OS_PI_VALUE 2*asin(1.0)

Definition at line 81 of file OSParameters.h.

7.55.2.50 #define OS_SCHEMA_VERSION "2.0"

Definition at line 83 of file OSParameters.h.

7.55.2.51 #define OS_NEAR_EQUAL 1e-2

we use OS_NEAR_EQUAL in unitTest to see if we are close to the optimal obj value

Definition at line 89 of file OSParameters.h.

7.55.2.52 #define OS_EPS 1e-12

Definition at line 91 of file OSParameters.h.

7.55.2.53 #define DEFAULT_OUTPUT_LEVEL ENUM_OUTPUT_LEVEL_error

Definition at line 121 of file OSParameters.h.

7.55.3 Enumeration Type Documentation**7.55.3.1 enum ENUM_OUTPUT_LEVEL**

Enumeration for the different verbosity levels that can be used in producing output.

The last three levels are used only in debug mode.

Enumerator:

ENUM_OUTPUT_LEVEL_always

ENUM_OUTPUT_LEVEL_error

ENUM_OUTPUT_LEVEL_summary
ENUM_OUTPUT_LEVEL_warning
ENUM_OUTPUT_LEVEL_info
ENUM_OUTPUT_LEVEL_debug
ENUM_OUTPUT_LEVEL_trace
ENUM_OUTPUT_LEVEL_detailed_trace
ENUM_OUTPUT_LEVEL_NUMBER_OF_LEVELS

Definition at line 107 of file OSParameters.h.

7.55.3.2 enum ENUM_OUTPUT_AREA

Enumeration for the different areas that can produce output.

The last entry ENUM_OUTPUT_AREA_NUMBER_OF_AREAS gives a convenient way to count them and to allocate space

Enumerator:

ENUM_OUTPUT_AREA_main
ENUM_OUTPUT_AREA_OSAgent
ENUM_OUTPUT_AREA_Command_line_parser
ENUM_OUTPUT_AREA_OSiL_parser
ENUM_OUTPUT_AREA_OSoL_parser
ENUM_OUTPUT_AREA_OSrL_parser
ENUM_OUTPUT_AREA_OSGeneral
ENUM_OUTPUT_AREA_OSInstance
ENUM_OUTPUT_AREA_OSOption
ENUM_OUTPUT_AREA_OSResult
ENUM_OUTPUT_AREA_OSMatrix
ENUM_OUTPUT_AREA_OSiLwriter
ENUM_OUTPUT_AREA_OSoLwriter
ENUM_OUTPUT_AREA_OSrLwriter
ENUM_OUTPUT_AREA_OSMoDelInterfaces
ENUM_OUTPUT_AREA_OSSolverInterfaces
ENUM_OUTPUT_AREA_OSUtils
ENUM_OUTPUT_AREA_NUMBER_OF_AREAS

Definition at line 128 of file OSParameters.h.

7.55.3.3 enum ENUM_CPUSPEEDUNIT

Enumerator:

ENUM_CPUSPEEDUNIT_hertz
ENUM_CPUSPEEDUNIT_kilohertz
ENUM_CPUSPEEDUNIT_megahertz
ENUM_CPUSPEEDUNIT_gigahertz

ENUM_CPUSPEEDUNIT_terahertz
ENUM_CPUSPEEDUNIT_flops
ENUM_CPUSPEEDUNIT_kiloflops
ENUM_CPUSPEEDUNIT_megaflops
ENUM_CPUSPEEDUNIT_gigaflops
ENUM_CPUSPEEDUNIT_teraflops
ENUM_CPUSPEEDUNIT_petaflops

Definition at line 161 of file OSParameters.h.

7.55.3.4 enum ENUM_STORAGEUNIT

Enumerator:

ENUM_STORAGEUNIT_byte
ENUM_STORAGEUNIT_kilobyte
ENUM_STORAGEUNIT_megabyte
ENUM_STORAGEUNIT_gigabyte
ENUM_STORAGEUNIT_terabyte
ENUM_STORAGEUNIT_petabyte
ENUM_STORAGEUNIT_exabyte
ENUM_STORAGEUNIT_zettabyte
ENUM_STORAGEUNIT_yottabyte

Definition at line 197 of file OSParameters.h.

7.55.3.5 enum ENUM_TIMEUNIT

Enumerator:

ENUM_TIMEUNIT_tick
ENUM_TIMEUNIT_millisecond
ENUM_TIMEUNIT_second
ENUM_TIMEUNIT_minute
ENUM_TIMEUNIT_hour
ENUM_TIMEUNIT_day
ENUM_TIMEUNIT_week
ENUM_TIMEUNIT_month
ENUM_TIMEUNIT_year

Definition at line 229 of file OSParameters.h.

7.55.3.6 enum ENUM_TIMETYPE

Enumerator:

ENUM_TIMETYPE_cpuTime
ENUM_TIMETYPE_elapsedTime
ENUM_TIMETYPE_other

Definition at line 261 of file OSParameters.h.

7.55.3.7 enum ENUM_TIMECATEGORY

Enumerator:

ENUM_TIMECATEGORY_total
ENUM_TIMECATEGORY_input
ENUM_TIMECATEGORY_preprocessing
ENUM_TIMECATEGORY_optimization
ENUM_TIMECATEGORY_postprocessing
ENUM_TIMECATEGORY_output
ENUM_TIMECATEGORY_other

Definition at line 281 of file OSParameters.h.

7.55.3.8 enum ENUM_LOCATIONTYPE

Enumerator:

ENUM_LOCATIONTYPE_local
ENUM_LOCATIONTYPE_http
ENUM_LOCATIONTYPE_ftp

Definition at line 309 of file OSParameters.h.

7.55.3.9 enum ENUM_TRANSPORT_TYPE

Enumerator:

ENUM_TRANSPORT_TYPE_osp
ENUM_TRANSPORT_TYPE_http
ENUM_TRANSPORT_TYPE_smtp
ENUM_TRANSPORT_TYPE_ftp
ENUM_TRANSPORT_TYPE_other

Definition at line 329 of file OSParameters.h.

7.55.3.10 enum ENUM_SERVICE_TYPE

Enumerator:

ENUM_SERVICE_TYPE_analyzer
ENUM_SERVICE_TYPE_solver
ENUM_SERVICE_TYPE_scheduler
ENUM_SERVICE_TYPE_modeler
ENUM_SERVICE_TYPE_registry
ENUM_SERVICE_TYPE_agent
ENUM_SERVICE_TYPE_simulations

Definition at line 353 of file OSParameters.h.

7.55.3.11 enum ENUM_GENERAL_RESULT_STATUS

Enumerator:

ENUM_GENERAL_RESULT_STATUS_error
ENUM_GENERAL_RESULT_STATUS_warning
ENUM_GENERAL_RESULT_STATUS_normal

Definition at line 381 of file OSParameters.h.

7.55.3.12 enum ENUM_SYSTEM_CURRENT_STATE

Enumerator:

ENUM_SYSTEM_CURRENT_STATE_busy
ENUM_SYSTEM_CURRENT_STATE_busyButAccepting
ENUM_SYSTEM_CURRENT_STATE_idle
ENUM_SYSTEM_CURRENT_STATE_idleButNotAccepting
ENUM_SYSTEM_CURRENT_STATE_noResponse

Definition at line 401 of file OSParameters.h.

7.55.3.13 enum ENUM_JOB_STATUS

Enumerator:

ENUM_JOB_STATUS_waiting
ENUM_JOB_STATUS_running
ENUM_JOB_STATUS_killed
ENUM_JOB_STATUS_finished
ENUM_JOB_STATUS_unknown

Definition at line 425 of file OSParameters.h.

7.55.3.14 enum ENUM_BASIS_STATUS

Enumeration for the different states that can be used in representating a basis The last state, ENUM_BASIS_STATUS_NUMBER_OF_STATES, is used *only* to record the number of states, which makes it easier to convert between different representations.

(For instance, AMPL uses a different order, so there may be a need to recode values. See OSosrl2ampl.cpp for an application.)

Enumerator:

ENUM_BASIS_STATUS_basic
ENUM_BASIS_STATUS_atLower
ENUM_BASIS_STATUS_atUpper
ENUM_BASIS_STATUS_atEquality
ENUM_BASIS_STATUS_isFree
ENUM_BASIS_STATUS_superbasic
ENUM_BASIS_STATUS_unknown
ENUM_BASIS_STATUS_NUMBER_OF_STATES

Definition at line 456 of file OSParameters.h.

7.55.3.15 enum ENUM_SOLUTION_STATUS

Enumerator:

ENUM_SOLUTION_STATUS_unbounded
ENUM_SOLUTION_STATUS_globallyOptimal
ENUM_SOLUTION_STATUS_locallyOptimal
ENUM_SOLUTION_STATUS_optimal
ENUM_SOLUTION_STATUS_bestSoFar
ENUM_SOLUTION_STATUS_feasible
ENUM_SOLUTION_STATUS_infeasible
ENUM_SOLUTION_STATUS_unsure
ENUM_SOLUTION_STATUS_error
ENUM_SOLUTION_STATUS_other

Definition at line 498 of file OSParameters.h.

7.55.3.16 enum ENUM_SOLUTION_SUBSTATUSTYPE

Enumerator:

ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByLimit
ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByBounds
ENUM_SOLUTION_SUBSTATUSTYPE_other

Definition at line 532 of file OSParameters.h.

7.55.3.17 enum ENUM_PROBLEM_COMPONENT

Enumerator:

ENUM_PROBLEM_COMPONENT_variables
ENUM_PROBLEM_COMPONENT_objectives
ENUM_PROBLEM_COMPONENT_constraints

Definition at line 552 of file OSParameters.h.

7.55.3.18 enum ENUM_VARTYPE

Enumerator:

ENUM_VARTYPE_continuous
ENUM_VARTYPE_binary
ENUM_VARTYPE_integer
ENUM_VARTYPE_string
ENUM_VARTYPE_semicontinuous
ENUM_VARTYPE_semiinteger

Definition at line 559 of file OSParameters.h.

7.55.3.19 enum ENUM_PATHPAIR

Enumerator:

ENUM_PATHPAIR_input_dir
ENUM_PATHPAIR_input_file
ENUM_PATHPAIR_output_file
ENUM_PATHPAIR_output_dir

Definition at line 586 of file OSParameters.h.

7.55.3.20 enum ENUM_MATRIX_TYPE

An enum to track the many different types of values that a matrix can contain Note that these types are partially ordered, which makes it easier to infer a matrix's type from the types of its constructors.

Enumerator:

ENUM_MATRIX_TYPE_zero
ENUM_MATRIX_TYPE_constant
ENUM_MATRIX_TYPE_varref
ENUM_MATRIX_TYPE_linear
ENUM_MATRIX_TYPE_quadratic
ENUM_MATRIX_TYPE_general
ENUM_MATRIX_TYPE_conref
ENUM_MATRIX_TYPE_objref
ENUM_MATRIX_TYPE_mixedref
ENUM_MATRIX_TYPE_jumbled
ENUM_MATRIX_TYPE_unknown

Definition at line 599 of file OSParameters.h.

7.55.3.21 enum ENUM_CONREFERENCE_VALUETYPE

An enum to track the type of value contained in a reference to a constraint.

Enumerator:

ENUM_CONREFERENCE_VALUETYPE_value
ENUM_CONREFERENCE_VALUETYPE_status
ENUM_CONREFERENCE_VALUETYPE_surplus
ENUM_CONREFERENCE_VALUETYPE_shortage

Definition at line 704 of file OSParameters.h.

7.55.3.22 enum ENUM_MATRIX_SYMMETRY

Enumerator:

ENUM_MATRIX_SYMMETRY_none
ENUM_MATRIX_SYMMETRY_upper

ENUM_MATRIX_SYMMETRY_lower
ENUM_MATRIX_SYMMETRY_skewUpper
ENUM_MATRIX_SYMMETRY_skewLower
ENUM_MATRIX_SYMMETRY_HermitianLower
ENUM_MATRIX_SYMMETRY_HermitianUpper

Definition at line 736 of file OSParameters.h.

7.55.3.23 enum ENUM_MATRIX_CONSTRUCTOR_TYPE

Enumerator:

ENUM_MATRIX_CONSTRUCTOR_TYPE_unknown
ENUM_MATRIX_CONSTRUCTOR_TYPE_baseMatrix
ENUM_MATRIX_CONSTRUCTOR_TYPE_constantElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_varRefElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_linearElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_generalElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_objRefElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_conRefElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_rowRefElements
ENUM_MATRIX_CONSTRUCTOR_TYPE_transformation
ENUM_MATRIX_CONSTRUCTOR_TYPE_blocks
ENUM_MATRIX_CONSTRUCTOR_TYPE_block
ENUM_MATRIX_CONSTRUCTOR_TYPE_matrix

Definition at line 777 of file OSParameters.h.

7.55.3.24 enum ENUM_COMBINE_ARRAYS

An enum to streamline set() methods of vectors.

Enumerator:

ENUM_COMBINE_ARRAYS_replace
ENUM_COMBINE_ARRAYS_merge
ENUM_COMBINE_ARRAYS_ignore
ENUM_COMBINE_ARRAYS_throw

Definition at line 819 of file OSParameters.h.

7.55.3.25 enum ENUM_NL_EXPR_SHAPE

Enumerator:

ENUM_NL_EXPR_SHAPE_general
ENUM_NL_EXPR_SHAPE_convex
ENUM_NL_EXPR_SHAPE_quadratic
ENUM_NL_EXPR_SHAPE_linear
ENUM_NL_EXPR_SHAPE_constant

Definition at line 829 of file OSParameters.h.

7.55.3.26 enum ENUM_CONE_TYPE

Enumerator:

ENUM_CONE_TYPE_nonnegative
ENUM_CONE_TYPE_nonpositive
ENUM_CONE_TYPE_orthant
ENUM_CONE_TYPE_polyhedral
ENUM_CONE_TYPE_quadratic
ENUM_CONE_TYPE_rotatedQuadratic
ENUM_CONE_TYPE_normed
ENUM_CONE_TYPE_rotatedNormed
ENUM_CONE_TYPE_semidefinite
ENUM_CONE_TYPE_copositiveMatrices
ENUM_CONE_TYPE_completelyPositiveMatrices
ENUM_CONE_TYPE_hyperbolicity
ENUM_CONE_TYPE_sumOfSquaresPolynomials
ENUM_CONE_TYPE_nonnegativePolynomials
ENUM_CONE_TYPE_moments
ENUM_CONE_TYPE_product
ENUM_CONE_TYPE_intersection
ENUM_CONE_TYPE_dual
ENUM_CONE_TYPE_polar
ENUM_CONE_TYPE_unknown

Definition at line 864 of file OSParameters.h.

7.55.4 Function Documentation

7.55.4.1 bool OSIsnan (double x)

checks whether a given double is NaN

7.55.4.2 double OSNaN ()

returns the value for NaN used in OS

7.55.4.3 std::string OSgetVersionInfo ()

7.55.4.4 int returnCPUSpeedUnit (std::string unit) [inline]

Definition at line 176 of file OSParameters.h.

7.55.4.5 bool verifyCPUSpeedUnit (std::string unit) [inline]

Definition at line 192 of file OSParameters.h.

7.55.4.6 int returnStorageUnit (std::string unit) [inline]

Definition at line 210 of file OSParameters.h.

7.55.4.7 `bool verifyStorageUnit (std::string unit)` `[inline]`

Definition at line 224 of file OSParameters.h.

7.55.4.8 `int returnTimeUnit (std::string unit)` `[inline]`

Definition at line 242 of file OSParameters.h.

7.55.4.9 `bool verifyTimeUnit (std::string unit)` `[inline]`

Definition at line 256 of file OSParameters.h.

7.55.4.10 `int returnTimeType (std::string type)` `[inline]`

Definition at line 268 of file OSParameters.h.

7.55.4.11 `bool verifyTimeType (std::string type)` `[inline]`

Definition at line 276 of file OSParameters.h.

7.55.4.12 `int returnTimeCategory (std::string category)` `[inline]`

Definition at line 292 of file OSParameters.h.

7.55.4.13 `bool verifyTimeCategory (std::string category)` `[inline]`

Definition at line 304 of file OSParameters.h.

7.55.4.14 `int returnLocationType (std::string type)` `[inline]`

Definition at line 316 of file OSParameters.h.

7.55.4.15 `bool verifyLocationType (std::string type)` `[inline]`

Definition at line 324 of file OSParameters.h.

7.55.4.16 `int returnTransportType (std::string type)` `[inline]`

Definition at line 338 of file OSParameters.h.

7.55.4.17 `bool verifyTransportType (std::string type)` `[inline]`

Definition at line 348 of file OSParameters.h.

7.55.4.18 `int returnServiceType (std::string type)` `[inline]`

Definition at line 364 of file OSParameters.h.

7.55.4.19 `bool verifyServiceType (std::string type)` `[inline]`

Definition at line 376 of file OSParameters.h.

7.55.4.20 `int returnGeneralResultStatus (std::string status)` `[inline]`

Definition at line 388 of file OSParameters.h.

7.55.4.21 `bool verifyGeneralResultStatus (std::string status)` `[inline]`

Definition at line 396 of file OSParameters.h.

7.55.4.22 `int returnSystemCurrentState (std::string status)` `[inline]`

Definition at line 410 of file OSParameters.h.

7.55.4.23 `bool verifySystemCurrentState (std::string status)` `[inline]`

Definition at line 420 of file OSParameters.h.

7.55.4.24 `int returnJobStatus (std::string status)` `[inline]`

Definition at line 434 of file OSParameters.h.

7.55.4.25 `bool verifyJobStatus (std::string status)` `[inline]`

Definition at line 444 of file OSParameters.h.

7.55.4.26 `int returnBasisStatus (std::string status)` `[inline]`

Definition at line 468 of file OSParameters.h.

7.55.4.27 `bool verifyBasisStatus (std::string status)` `[inline]`

Definition at line 480 of file OSParameters.h.

7.55.4.28 `std::string returnBasisStatusString (ENUM_BASIS_STATUS status)` `[inline]`

Definition at line 485 of file OSParameters.h.

7.55.4.29 `int returnSolutionStatus (std::string status)` `[inline]`

Definition at line 512 of file OSParameters.h.

7.55.4.30 `bool verifySolutionStatus (std::string status)` `[inline]`

Definition at line 527 of file OSParameters.h.

7.55.4.31 `int returnSolutionSubstatusType (std::string type)` `[inline]`

Definition at line 539 of file OSParameters.h.

7.55.4.32 `bool verifySolutionSubstatusType (std::string type)` `[inline]`

Definition at line 547 of file OSParameters.h.

7.55.4.33 `int returnVarType (char vt)` `[inline]`

Definition at line 569 of file OSParameters.h.

7.55.4.34 `bool verifyVarType (char vt)` `[inline]`

Definition at line 580 of file OSParameters.h.

7.55.4.35 `int returnMatrixType (std::string type) [inline]`

Definition at line 619 of file OSParameters.h.

7.55.4.36 `std::string returnMatrixTypeString (ENUM_MATRIX_TYPE type) [inline]`

Definition at line 637 of file OSParameters.h.

7.55.4.37 `bool verifyMatrixType (std::string type) [inline]`

Definition at line 653 of file OSParameters.h.

7.55.4.38 `ENUM_MATRIX_TYPE mergeMatrixType (ENUM_MATRIX_TYPE type1, ENUM_MATRIX_TYPE type2) [inline]`

A function to merge two matrix types so we can infer the type of a matrix recursively.

Definition at line 661 of file OSParameters.h.

7.55.4.39 `std::string returnConReferenceValueTypeString (ENUM_CONREFERENCE_VALUETYPE valueType) [inline]`

Definition at line 712 of file OSParameters.h.

7.55.4.40 `int returnConReferenceValueType (std::string valueType) [inline]`

Definition at line 721 of file OSParameters.h.

7.55.4.41 `bool verifyConReferenceValueType (std::string valueType) [inline]`

Definition at line 730 of file OSParameters.h.

7.55.4.42 `std::string returnMatrixSymmetryString (ENUM_MATRIX_SYMMETRY symmetry) [inline]`

Definition at line 747 of file OSParameters.h.

7.55.4.43 `int returnMatrixSymmetry (std::string symmetry) [inline]`

Definition at line 759 of file OSParameters.h.

7.55.4.44 `bool verifyMatrixSymmetry (std::string symmetry) [inline]`

Definition at line 771 of file OSParameters.h.

7.55.4.45 `int returnMatrixConstructorType (std::string cType) [inline]`

Definition at line 794 of file OSParameters.h.

7.55.4.46 `bool verifyMatrixConstructorType (std::string type) [inline]`

Definition at line 810 of file OSParameters.h.

7.55.4.47 `int returnNExprShape (std::string shape) [inline]`

Definition at line 838 of file OSParameters.h.

7.55.4.48 `std::string returnExprShapeString (ENUM_NL_EXPR_SHAPE shape)` `[inline]`

Definition at line 848 of file `OSParameters.h`.

7.55.4.49 `bool verifyNExprShape (std::string shape)` `[inline]`

Definition at line 858 of file `OSParameters.h`.

7.55.4.50 `int returnConeType (std::string type)` `[inline]`

Definition at line 888 of file `OSParameters.h`.

7.55.4.51 `bool verifyConeType (std::string type)` `[inline]`

Definition at line 914 of file `OSParameters.h`.

7.55.5 Variable Documentation

7.55.5.1 `const double OSDBL_MAX = std::numeric_limits<double>::max()`

Definition at line 93 of file `OSParameters.h`.

7.55.5.2 `const int OSINT_MAX = std::numeric_limits<int>::max()`

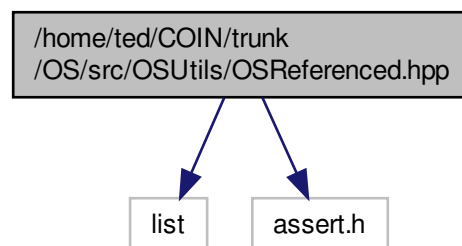
Definition at line 94 of file `OSParameters.h`.

7.56 /home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.hpp File Reference

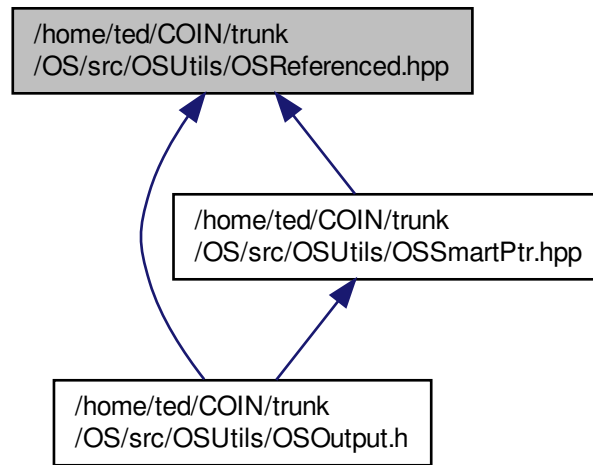
```
#include <list>
```

```
#include <assert.h>
```

Include dependency graph for `OSReferenced.hpp`:



This graph shows which files directly or indirectly include this file:



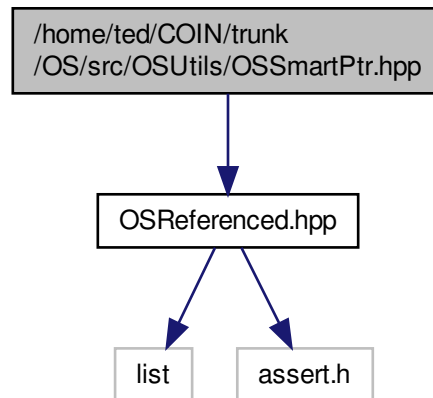
Classes

- class [OSReferencer](#)
Pseudo-class, from which everything has to inherit that wants to use be registered as a Referencer for a ReferencedObject.
- class [OSReferencedObject](#)
ReferencedObject class.

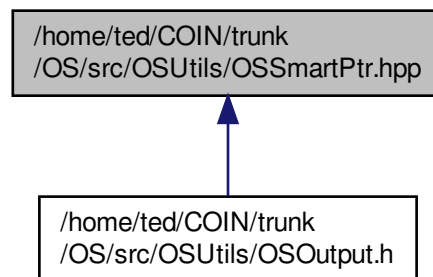
7.57 /home/ted/COIN/trunk/OS/src/OSUtils/OSSmartPtr.hpp File Reference

```
#include "OSReferenced.hpp"
```

Include dependency graph for OSSmartPtr.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class `OSSmartPtr< T >`
Template class for Smart Pointers.

Functions

- template<class U1 , class U2 >
bool `ComparePointers` (const U1 *lhs, const U2 *rhs)

OSSmartPtr friend function declarations.

- `template<class U >`
`U * GetRawPtr (const OSSmartPtr< U > &smart_ptr)`
- `template<class U >`
`OSSmartPtr< const U > ConstPtr (const OSSmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool IsNull (const OSSmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool IsValid (const OSSmartPtr< U > &smart_ptr)`
- `template<class U1 , class U2 >`
`bool operator== (const OSSmartPtr< U1 > &lhs, const OSSmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool operator== (const OSSmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator== (U1 *lhs, const OSSmartPtr< U2 > &raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (const OSSmartPtr< U1 > &lhs, const OSSmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (const OSSmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (U1 *lhs, const OSSmartPtr< U2 > &raw_rhs)`

7.57.1 Function Documentation**7.57.1.1 `template<class U > U * GetRawPtr (const OSSmartPtr< U > & smart_ptr)`**

Use to get the value of the raw ptr (i.e. to pass to other methods/functions, etc.) Note: This method does NOT copy, therefore, modifications using this value modify the underlying object contained by the `OSSmartPtr`, NEVER delete this returned value.

Definition at line 452 of file `OSSmartPtr.hpp`.

7.57.1.2 `template<class U > OSSmartPtr< const U > ConstPtr (const OSSmartPtr< U > & smart_ptr)`

Definition at line 458 of file `OSSmartPtr.hpp`.

7.57.1.3 `template<class U > bool IsNull (const OSSmartPtr< U > & smart_ptr)`

Use this to check if the `OSSmartPtr` IsNull. This is preferred to `if(GetRawPtr(sp) == NULL)`

Definition at line 471 of file `OSSmartPtr.hpp`.

7.57.1.4 `template<class U > bool IsValid (const OSSmartPtr< U > & smart_ptr)`

Use this to check if the `OSSmartPtr` is not null This is preferred to `if(GetRawPtr(sp) != NULL)`

Definition at line 465 of file `OSSmartPtr.hpp`.

7.57.1.5 `template<class U1 , class U2 > bool operator== (const OSSmartPtr< U1 > & lhs, const OSSmartPtr< U2 > & rhs)`

Definition at line 499 of file `OSSmartPtr.hpp`.

7.57.1.6 `template<class U1 , class U2 > bool operator== (const OSSmartPtr< U1 > & lhs, U2 * raw_rhs)`

Definition at line 507 of file `OSSmartPtr.hpp`.

7.57.1.7 `template<class U1 , class U2 > bool operator==(U1 * lhs, const OSSmartPtr< U2 > & raw_rhs)`

Definition at line 514 of file OSSmartPtr.hpp.

7.57.1.8 `template<class U1 , class U2 > bool operator!=(const OSSmartPtr< U1 > & lhs, const OSSmartPtr< U2 > & rhs)`

Definition at line 521 of file OSSmartPtr.hpp.

7.57.1.9 `template<class U1 , class U2 > bool operator!=(const OSSmartPtr< U1 > & lhs, U2 * raw_rhs)`

Definition at line 528 of file OSSmartPtr.hpp.

7.57.1.10 `template<class U1 , class U2 > bool operator!=(U1 * lhs, const OSSmartPtr< U2 > & raw_rhs)`

Definition at line 535 of file OSSmartPtr.hpp.

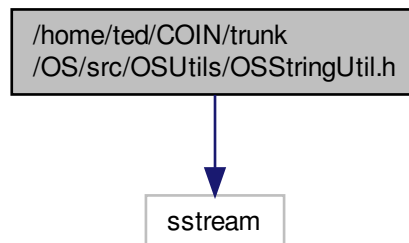
7.57.1.11 `template<class U1 , class U2 > bool ComparePointers (const U1 * lhs, const U2 * rhs)`

Definition at line 478 of file OSSmartPtr.hpp.

7.58 /home/ted/COIN/trunk/OS/src/OSUtils/OSStringUtil.h File Reference

```
#include <sstream>
```

Include dependency graph for OSStringUtil.h:



Functions

- `std::string writeStringData (std::string str)`
writeStringData
- `std::string makeStringFromInt (std::string theString, int theInt)`

7.58.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2010, Horand Gassmann, Jun Ma, Kipp Martin, Northwestern University, and the University of Chicago. All Rights Reserved. This software is licensed under the Eclipse Public License. Please see the accompanying LICENSE file in root directory for terms.

Definition in file [OSStringUtil.h](#).

7.58.2 Function Documentation

7.58.2.1 `std::string writeStringData (std::string str)`

`writeStringData`

Prepare and output a string that may contain special characters (single or double quotes)

Parameters

<i>str</i>	holds the string to be output. If the string does not contain double quotes, it is output surrounded by double quotes, if the string contains double quotes, it is output surrounded by single quotes,
------------	--

Returns

the prepared string, ready to be printed

7.58.2.2 `std::string makeStringFromInt (std::string theString, int theInt)`

Index

- ~Base64
 - Base64, [31](#)
- ~BaseMatrix
 - BaseMatrix, [34](#)
- ~BasisStatus
 - BasisStatus, [38](#)
- ~BonminProblem
 - BonminProblem, [44](#)
- ~BonminSolver
 - BonminSolver, [48](#)
- ~BranchingWeight
 - BranchingWeight, [51](#)
- ~CPUNumber
 - CPUNumber, [102](#)
- ~CPUSpeed
 - CPUSpeed, [104](#)
- ~CoinSolver
 - CoinSolver, [55](#)
- ~CompletelyPositiveMatricesCone
 - CompletelyPositiveMatricesCone, [59](#)
- ~ConReferenceMatrixElement
 - ConReferenceMatrixElement, [68](#)
- ~ConReferenceMatrixElements
 - ConReferenceMatrixElements, [71](#)
- ~ConReferenceMatrixValues
 - ConReferenceMatrixValues, [74](#)
- ~Cone
 - Cone, [63](#)
- ~Cones
 - Cones, [66](#)
- ~ConstantMatrixElements
 - ConstantMatrixElements, [77](#)
- ~ConstantMatrixValues
 - ConstantMatrixValues, [80](#)
- ~Constraint
 - Constraint, [82](#)
- ~ConstraintOption
 - ConstraintOption, [84](#)
- ~ConstraintSolution
 - ConstraintSolution, [89](#)
- ~Constraints
 - Constraints, [87](#)
- ~ContactOption
 - ContactOption, [92](#)
- ~CopositiveMatricesCone
 - CopositiveMatricesCone, [95](#)
- ~CouenneSolver
 - CouenneSolver, [99](#)
- ~CsdpSolver
 - CsdpSolver, [108](#)
- ~DefaultSolver
 - DefaultSolver, [113](#)
- ~DirectoriesAndFiles
 - DirectoriesAndFiles, [116](#)
- ~DoubleVector
 - DoubleVector, [118](#)
- ~DualCone
 - DualCone, [120](#)
- ~DualVarValue
 - DualVarValue, [126](#)
- ~DualVariableValues
 - DualVariableValues, [123](#)
- ~ExpandedMatrixBlocks
 - ExpandedMatrixBlocks, [130](#)
- ~ExprNode
 - ExprNode, [134](#)
- ~FileUtil
 - FileUtil, [138](#)
- ~GeneralFileHeader
 - GeneralFileHeader, [141](#)
- ~GeneralMatrixElements
 - GeneralMatrixElements, [145](#)
- ~GeneralMatrixValues
 - GeneralMatrixValues, [148](#)
- ~GeneralOption
 - GeneralOption, [152](#)
- ~GeneralResult
 - GeneralResult, [155](#)
- ~GeneralSparseMatrix
 - GeneralSparseMatrix, [158](#)
- ~GeneralStatus
 - GeneralStatus, [161](#)
- ~GeneralSubstatus
 - GeneralSubstatus, [164](#)
- ~InitBasStatus
 - InitBasStatus, [168](#)
- ~InitConValue
 - InitConValue, [174](#)
- ~InitConstraintValues
 - InitConstraintValues, [171](#)
- ~InitDualVarValue
 - InitDualVarValue, [181](#)
- ~InitDualVariableValues
 - InitDualVariableValues, [177](#)
- ~InitObjBound
 - InitObjBound, [187](#)
- ~InitObjValue
 - InitObjValue, [198](#)
- ~InitObjectiveBounds
 - InitObjectiveBounds, [190](#)
- ~InitObjectiveValues
 - InitObjectiveValues, [194](#)

- ~InitVarValue
 - InitVarValue, [209](#)
- ~InitVarValueString
 - InitVarValueString, [212](#)
- ~InitVariableValues
 - InitVariableValues, [201](#)
- ~InitVariableValuesString
 - InitVariableValuesString, [205](#)
- ~InitialBasisStatus
 - InitialBasisStatus, [184](#)
- ~InstanceData
 - InstanceData, [215](#)
- ~InstanceLocationOption
 - InstanceLocationOption, [217](#)
- ~IntVector
 - IntVector, [228](#)
- ~IntegerVariableBranchingWeights
 - IntegerVariableBranchingWeights, [220](#)
- ~IntersectionCone
 - IntersectionCone, [224](#)
- ~IpoptProblem
 - IpoptProblem, [232](#)
- ~IpoptSolver
 - IpoptSolver, [236](#)
- ~JobDependencies
 - JobDependencies, [239](#)
- ~JobOption
 - JobOption, [242](#)
- ~JobResult
 - JobResult, [246](#)
- ~KnitroProblem
 - KnitroProblem, [250](#)
- ~KnitroSolver
 - KnitroSolver, [253](#)
- ~LindoSolver
 - LindoSolver, [257](#)
- ~LinearConstraintCoefficients
 - LinearConstraintCoefficients, [260](#)
- ~LinearMatrixElement
 - LinearMatrixElement, [262](#)
- ~LinearMatrixElementTerm
 - LinearMatrixElementTerm, [268](#)
- ~LinearMatrixElements
 - LinearMatrixElements, [265](#)
- ~LinearMatrixValues
 - LinearMatrixValues, [270](#)
- ~MathUtil
 - MathUtil, [272](#)
- ~Matrices
 - Matrices, [274](#)
- ~MatrixBlock
 - MatrixBlock, [277](#)
- ~MatrixBlocks
 - MatrixBlocks, [281](#)
- ~MatrixCon
 - MatrixCon, [285](#)
- ~MatrixConstraints
 - MatrixConstraints, [288](#)
- ~MatrixConstructor
 - MatrixConstructor, [290](#)
- ~MatrixElementValues
 - MatrixElementValues, [293](#)
- ~MatrixElements
 - MatrixElements, [292](#)
- ~MatrixExpression
 - MatrixExpression, [295](#)
- ~MatrixExpressionTree
 - MatrixExpressionTree, [301](#)
- ~MatrixExpressions
 - MatrixExpressions, [298](#)
- ~MatrixNode
 - MatrixNode, [303](#)
- ~MatrixObj
 - MatrixObj, [308](#)
- ~MatrixObjectives
 - MatrixObjectives, [311](#)
- ~MatrixProgramming
 - MatrixProgramming, [313](#)
- ~MatrixTransformation
 - MatrixTransformation, [316](#)
- ~MatrixType
 - MatrixType, [320](#)
- ~MatrixVar
 - MatrixVar, [325](#)
- ~MatrixVariables
 - MatrixVariables, [328](#)
- ~MaxTime
 - MaxTime, [330](#)
- ~MinCPUNumber
 - MinCPUNumber, [332](#)
- ~MinCPUSpeed
 - MinCPUSpeed, [334](#)
- ~MinDiskSpace
 - MinDiskSpace, [336](#)
- ~MinMemorySize
 - MinMemorySize, [338](#)
- ~NI
 - NI, [341](#)
- ~NonlinearExpressions
 - NonlinearExpressions, [344](#)
- ~NonnegativeCone
 - NonnegativeCone, [346](#)
- ~NonpositiveCone
 - NonpositiveCone, [349](#)
- ~OSCommandLine
 - OSCommandLine, [395](#)
- ~OSCommandLineReader
 - OSCommandLineReader, [400](#)

- ~OSExpressionTree
 - OSExpressionTree, [403](#)
- ~OSInstance
 - OSInstance, [443](#)
- ~OSMatlab
 - OSMatlab, [486](#)
- ~OSMatrix
 - OSMatrix, [491](#)
- ~OSOption
 - OSOption, [686](#)
- ~OSOutput
 - OSOutput, [723](#)
- ~OSOutputChannel
 - OSOutputChannel, [726](#)
- ~OSReferencedObject
 - OSReferencedObject, [729](#)
- ~OSResult
 - OSResult, [744](#)
- ~OSSmartPtr
 - OSSmartPtr, [834](#)
- ~OSSolverAgent
 - OSSolverAgent, [838](#)
- ~OSgLParseData
 - OSgLParseData, [409](#)
- ~OSgams2osil
 - OSgams2osil, [405](#)
- ~OShL
 - OShL, [417](#)
- ~OSiLParserData
 - OSiLParserData, [423](#)
- ~OSiLReader
 - OSiLReader, [431](#)
- ~OSiLWriter
 - OSiLWriter, [433](#)
- ~OSmps2OS
 - OSmps2OS, [496](#)
- ~OSmps2osil
 - OSmps2osil, [499](#)
- ~OSnLMNode
 - OSnLMNode, [507](#)
- ~OSnLMNodeDiagonalMatrixFromVector
 - OSnLMNodeDiagonalMatrixFromVector, [510](#)
- ~OSnLMNodeIdentityMatrix
 - OSnLMNodeIdentityMatrix, [512](#)
- ~OSnLMNodeMatrixCon
 - OSnLMNodeMatrixCon, [515](#)
- ~OSnLMNodeMatrixDiagonal
 - OSnLMNodeMatrixDiagonal, [517](#)
- ~OSnLMNodeMatrixDotTimes
 - OSnLMNodeMatrixDotTimes, [519](#)
- ~OSnLMNodeMatrixInverse
 - OSnLMNodeMatrixInverse, [521](#)
- ~OSnLMNodeMatrixLowerTriangle
 - OSnLMNodeMatrixLowerTriangle, [523](#)
- ~OSnLMNodeMatrixMinus
 - OSnLMNodeMatrixMinus, [525](#)
- ~OSnLMNodeMatrixNegate
 - OSnLMNodeMatrixNegate, [527](#)
- ~OSnLMNodeMatrixObj
 - OSnLMNodeMatrixObj, [530](#)
- ~OSnLMNodeMatrixPlus
 - OSnLMNodeMatrixPlus, [532](#)
- ~OSnLMNodeMatrixProduct
 - OSnLMNodeMatrixProduct, [534](#)
- ~OSnLMNodeMatrixReference
 - OSnLMNodeMatrixReference, [537](#)
- ~OSnLMNodeMatrixScalarTimes
 - OSnLMNodeMatrixScalarTimes, [539](#)
- ~OSnLMNodeMatrixSubmatrixAt
 - OSnLMNodeMatrixSubmatrixAt, [541](#)
- ~OSnLMNodeMatrixSum
 - OSnLMNodeMatrixSum, [543](#)
- ~OSnLMNodeMatrixTimes
 - OSnLMNodeMatrixTimes, [545](#)
- ~OSnLMNodeMatrixTranspose
 - OSnLMNodeMatrixTranspose, [547](#)
- ~OSnLMNodeMatrixUpperTriangle
 - OSnLMNodeMatrixUpperTriangle, [549](#)
- ~OSnLMNodeMatrixVar
 - OSnLMNodeMatrixVar, [552](#)
- ~OSnLNode
 - OSnLNode, [556](#)
- ~OSnLNodeAbs
 - OSnLNodeAbs, [561](#)
- ~OSnLNodeAllDiff
 - OSnLNodeAllDiff, [564](#)
- ~OSnLNodeCos
 - OSnLNodeCos, [567](#)
- ~OSnLNodeDivide
 - OSnLNodeDivide, [570](#)
- ~OSnLNodeE
 - OSnLNodeE, [573](#)
- ~OSnLNodeErf
 - OSnLNodeErf, [576](#)
- ~OSnLNodeExp
 - OSnLNodeExp, [579](#)
- ~OSnLNodeIf
 - OSnLNodeIf, [582](#)
- ~OSnLNodeLn
 - OSnLNodeLn, [585](#)
- ~OSnLNodeMatrixDeterminant
 - OSnLNodeMatrixDeterminant, [588](#)
- ~OSnLNodeMatrixToScalar
 - OSnLNodeMatrixToScalar, [591](#)
- ~OSnLNodeMatrixTrace
 - OSnLNodeMatrixTrace, [594](#)
- ~OSnLNodeMax
 - OSnLNodeMax, [597](#)

- ~OSnLNodeMin
 - OSnLNodeMin, [600](#)
- ~OSnLNodeMinus
 - OSnLNodeMinus, [603](#)
- ~OSnLNodeNegate
 - OSnLNodeNegate, [606](#)
- ~OSnLNodeNumber
 - OSnLNodeNumber, [610](#)
- ~OSnLNodePI
 - OSnLNodePI, [613](#)
- ~OSnLNodePlus
 - OSnLNodePlus, [616](#)
- ~OSnLNodePower
 - OSnLNodePower, [619](#)
- ~OSnLNodeProduct
 - OSnLNodeProduct, [622](#)
- ~OSnLNodeSin
 - OSnLNodeSin, [625](#)
- ~OSnLNodeSqrt
 - OSnLNodeSqrt, [628](#)
- ~OSnLNodeSquare
 - OSnLNodeSquare, [631](#)
- ~OSnLNodeSum
 - OSnLNodeSum, [634](#)
- ~OSnLNodeTimes
 - OSnLNodeTimes, [637](#)
- ~OSnLNodeVariable
 - OSnLNodeVariable, [640](#)
- ~OSnLParserData
 - OSnLParserData, [646](#)
- ~OSnl2OS
 - OSnl2OS, [502](#)
- ~OSoLParserData
 - OSoLParserData, [657](#)
- ~OSoLReader
 - OSoLReader, [672](#)
- ~OSoLWriter
 - OSoLWriter, [674](#)
- ~OSosl2ampl
 - OSosl2ampl, [721](#)
- ~OSrL2Gams
 - OSrL2Gams, [812](#)
- ~OSrLParserData
 - OSrLParserData, [817](#)
- ~OSrLReader
 - OSrLReader, [828](#)
- ~OSrLWriter
 - OSrLWriter, [830](#)
- ~ObjCoef
 - ObjCoef, [351](#)
- ~ObjReferenceMatrixElements
 - ObjReferenceMatrixElements, [367](#)
- ~ObjReferenceMatrixValues
 - ObjReferenceMatrixValues, [370](#)
- ~ObjValue
 - ObjValue, [372](#)
- ~Objective
 - Objective, [353](#)
- ~ObjectiveOption
 - ObjectiveOption, [355](#)
- ~ObjectiveSolution
 - ObjectiveSolution, [361](#)
- ~ObjectiveValues
 - ObjectiveValues, [364](#)
- ~Objectives
 - Objectives, [359](#)
- ~OptimizationOption
 - OptimizationOption, [375](#)
- ~OptimizationResult
 - OptimizationResult, [378](#)
- ~OptimizationSolution
 - OptimizationSolution, [381](#)
- ~OptimizationSolutionStatus
 - OptimizationSolutionStatus, [384](#)
- ~OptimizationSolutionSubstatus
 - OptimizationSolutionSubstatus, [387](#)
- ~OrthantCone
 - OrthantCone, [389](#)
- ~OtherConOption
 - OtherConOption, [841](#)
- ~OtherConResult
 - OtherConResult, [844](#)
- ~OtherConstraintOption
 - OtherConstraintOption, [848](#)
- ~OtherConstraintResult
 - OtherConstraintResult, [852](#)
- ~OtherObjOption
 - OtherObjOption, [863](#)
- ~OtherObjResult
 - OtherObjResult, [866](#)
- ~OtherObjectiveOption
 - OtherObjectiveOption, [856](#)
- ~OtherObjectiveResult
 - OtherObjectiveResult, [860](#)
- ~OtherOption
 - OtherOption, [868](#)
- ~OtherOptionEnumeration
 - OtherOptionEnumeration, [871](#)
- ~OtherOptions
 - OtherOptions, [874](#)
- ~OtherResult
 - OtherResult, [877](#)
- ~OtherResults
 - OtherResults, [880](#)
- ~OtherSolutionResult
 - OtherSolutionResult, [882](#)
- ~OtherSolutionResults
 - OtherSolutionResults, [885](#)

- ~OtherSolverOutput
 - OtherSolverOutput, [887](#)
- ~OtherVarOption
 - OtherVarOption, [899](#)
- ~OtherVarResult
 - OtherVarResult, [902](#)
- ~OtherVariableOption
 - OtherVariableOption, [890](#)
- ~OtherVariableResult
 - OtherVariableResult, [894](#)
- ~PathPair
 - PathPair, [904](#)
- ~PathPairs
 - PathPairs, [907](#)
- ~PolarCone
 - PolarCone, [911](#)
- ~PolyhedralCone
 - PolyhedralCone, [914](#)
- ~Processes
 - Processes, [918](#)
- ~ProductCone
 - ProductCone, [921](#)
- ~QuadraticCoefficients
 - QuadraticCoefficients, [924](#)
- ~QuadraticCone
 - QuadraticCone, [926](#)
- ~QuadraticTerm
 - QuadraticTerm, [929](#)
- ~QuadraticTerms
 - QuadraticTerms, [931](#)
- ~RotatedQuadraticCone
 - RotatedQuadraticCone, [933](#)
- ~RowReferenceMatrixElements
 - RowReferenceMatrixElements, [938](#)
- ~SOSVariableBranchingWeights
 - SOSVariableBranchingWeights, [965](#)
- ~SOSWeights
 - SOSWeights, [968](#)
- ~ScalarExpressionTree
 - ScalarExpressionTree, [942](#)
- ~SemidefiniteCone
 - SemidefiniteCone, [945](#)
- ~ServiceOption
 - ServiceOption, [949](#)
- ~ServiceResult
 - ServiceResult, [952](#)
- ~SolverOption
 - SolverOption, [955](#)
- ~SolverOptions
 - SolverOptions, [959](#)
- ~SolverOutput
 - SolverOutput, [962](#)
- ~SparseHessianMatrix
 - SparseHessianMatrix, [971](#)
- ~SparseIntVector
 - SparseIntVector, [973](#)
- ~SparseJacobianMatrix
 - SparseJacobianMatrix, [974](#)
- ~SparseMatrix
 - SparseMatrix, [976](#)
- ~SparseVector
 - SparseVector, [978](#)
- ~StorageCapacity
 - StorageCapacity, [980](#)
- ~SystemOption
 - SystemOption, [983](#)
- ~SystemResult
 - SystemResult, [986](#)
- ~TimeDomain
 - TimeDomain, [988](#)
- ~TimeDomainInterval
 - TimeDomainInterval, [989](#)
- ~TimeDomainStage
 - TimeDomainStage, [990](#)
- ~TimeDomainStageCon
 - TimeDomainStageCon, [992](#)
- ~TimeDomainStageConstraints
 - TimeDomainStageConstraints, [993](#)
- ~TimeDomainStageObj
 - TimeDomainStageObj, [994](#)
- ~TimeDomainStageObjectives
 - TimeDomainStageObjectives, [995](#)
- ~TimeDomainStageVar
 - TimeDomainStageVar, [998](#)
- ~TimeDomainStageVariables
 - TimeDomainStageVariables, [999](#)
- ~TimeDomainStages
 - TimeDomainStages, [997](#)
- ~TimeMeasurement
 - TimeMeasurement, [1001](#)
- ~TimeSpan
 - TimeSpan, [1004](#)
- ~TimingInformation
 - TimingInformation, [1007](#)
- ~VarReferenceMatrixElements
 - VarReferenceMatrixElements, [1025](#)
- ~VarReferenceMatrixValues
 - VarReferenceMatrixValues, [1029](#)
- ~VarValue
 - VarValue, [1031](#)
- ~VarValueString
 - VarValueString, [1033](#)
- ~Variable
 - Variable, [1009](#)
- ~VariableOption
 - VariableOption, [1011](#)
- ~VariableSolution
 - VariableSolution, [1017](#)

- ~VariableValues
 - VariableValues, [1020](#)
- ~VariableValuesString
 - VariableValuesString, [1022](#)
- ~Variables
 - Variables, [1015](#)
- ~WSUtil
 - WSUtil, [1035](#)
- /home/ted/COIN/trunk/OS/src/OSAgent/OSSolverAgent.h, [1043](#)
- /home/ted/COIN/trunk/OS/src/OSAgent/OSWSUtil.h, [1044](#)
- /home/ted/COIN/trunk/OS/src/OSAgent/OShL.h, [1042](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-CommandLine.h, [1046](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-CommandLineReader.h, [1047](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-DefaultSolver.h, [1048](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-ExpressionTree.h, [1048](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-General.h, [1050](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-Instance.h, [1057](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-Matrix.h, [1060](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-Option.h, [1069](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OS-Result.h, [1072](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/O-SgLWriter.h, [1052](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSi-LReader.h, [1055](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSi-LWriter.h, [1056](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/O-SnLNode.h, [1063](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/O-SoLReader.h, [1066](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/O-SoLWriter.h, [1068](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSr-LReader.h, [1075](#)
- /home/ted/COIN/trunk/OS/src/OSCommonInterfaces/OSr-LWriter.h, [1076](#)
- /home/ted/COIN/trunk/OS/src/OSConfig.h, [1077](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/O-Sgams2osil.hpp, [1078](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/O-Smps2OS.h, [1078](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/O-Smps2osil.h, [1079](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/OSnl2-OS.h, [1080](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/O-Sosl2ampl.h, [1082](#)
- /home/ted/COIN/trunk/OS/src/OSModelInterfaces/O-Sosl2gams.hpp, [1083](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSOptions-Struc.h, [1089](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSParseosil.-tab.hpp, [1090](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSParseosol.-tab.hpp, [1158](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSParseosrl.-tab.hpp, [1233](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSgLPParser-Data.h, [1083](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSiLPParser-Data.h, [1085](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSnLPParser-Data.h, [1086](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSoLPParser-Data.h, [1088](#)
- /home/ted/COIN/trunk/OS/src/OSarsers/OSrLPParser-Data.h, [1305](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-BonminSolver.h, [1308](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-CoinSolver.h, [1309](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-CouenneSolver.h, [1310](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-CsdpSolver.h, [1311](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-lpoptSolver.h, [1313](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-KnitroSolver.h, [1314](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-LindoSolver.h, [1315](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-MatlabSolver.h, [1316](#)
- /home/ted/COIN/trunk/OS/src/OSSolverInterfaces/OS-RunSolver.h, [1316](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSBase64.h, [1319](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSDataStructures.-h, [1320](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSErrorClass.h, [1322](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSFileUtil.h, [1322](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSMathUtil.h, [1323](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSOutput.h, [1326](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSParameters.h, [1328](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSReferenced.-

- hpp, [1349](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSSmartPtr.hpp, [1350](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSStringUtil.h, [1353](#)
- /home/ted/COIN/trunk/OS/src/OSUtils/OSdtoa.h, [1321](#)
- /home/ted/COIN/trunk/OS/src/config_default.h, [1039](#)
- /home/ted/COIN/trunk/OS/src/config_os_default.h, [1040](#)
- ABSEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- ABSSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- ACTUALSTARTTIMEEND
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- ACTUALSTARTTIMESTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- ALLDIFFEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- ALLDIFFSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- ATEQUALITYEND
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- ATEQUALITYSTART
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- ATLOWEREND
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- ATLOWERSTART
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- ATTRIBUTETEXT
 - OSParseosil.tab.hpp, [1130](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- ATUPPEREND
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- ATUPPERSTART
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- AVAILABLECPUNUMBEREND
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLECPUNUMBERSTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLECPUSPEEDEND
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLECPUSPEEDSTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLEDISKSPACEEND
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLEDISKSPACESTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLEMEMORYEND
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AVAILABLEMEMORYSTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- AXISDIRECTIONATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1279](#)
- ABSEND
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- ABSSTART
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- ACTUALSTARTTIMEEND
 - OSParseosrl.tab.hpp, [1255](#)
- ACTUALSTARTTIMESTART

- OSParseosrl.tab.hpp, 1255
- ADdouble
 - OSnLNode.h, 1066
- ADvector
 - OSnLNode.h, 1066
- ALLDIFFEND
 - OSParseosil.tab.hpp, 1126
 - OSParseosol.tab.hpp, 1201
 - OSParseosrl.tab.hpp, 1274
- ALLDIFFSTART
 - OSParseosil.tab.hpp, 1126
 - OSParseosol.tab.hpp, 1201
 - OSParseosrl.tab.hpp, 1274
- ATEQUALITYEND
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- ATEQUALITYSTART
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- ATLOWEREND
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- ATLOWERSTART
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- ATTRIBUTETEXT
 - OSParseosil.tab.hpp, 1106
 - OSParseosol.tab.hpp, 1175
 - OSParseosrl.tab.hpp, 1249
- ATUPPEREND
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- ATUPPERSTART
 - OSParseosol.tab.hpp, 1187
 - OSParseosrl.tab.hpp, 1255
- AVAILABLECPUSPEEDEND
 - OSParseosrl.tab.hpp, 1256
- AVAILABLEMEMORYEND
 - OSParseosrl.tab.hpp, 1256
- AVAILABLEMEMORYSTART
 - OSParseosrl.tab.hpp, 1256
- AXISDIRECTIONATT
 - OSParseosil.tab.hpp, 1110
- actualStartTime
 - JobResult, 247
- actualStartTimePresent
 - OSrLParserData, 825
- AddChannel
 - OSOutput, 724
- addCon
 - InitConstraintValues, 172
 - InitDualVariableValues, 179
 - OtherConstraintOption, 848
- addCone
 - OSInstance, 468–473
- addConstraint
 - OSInstance, 464
- addIdx
 - BasisStatus, 39
- addJobID
 - JobDependencies, 240
- addMatrix
 - OSInstance, 468
- addObj
 - InitObjectiveBounds, 191
 - InitObjectiveValues, 195
 - OtherObjectiveOption, 857
- addObjective
 - OSInstance, 463
- addOther
 - ConstraintOption, 85
 - ObjectiveOption, 356
 - OtherOptions, 875
 - VariableOption, 1012
- addPath
 - DirectoriesAndFiles, 117
- addPathPair
 - PathPairs, 908
- addProcess
 - Processes, 919
- addQTermsToExpressionTree
 - OSInstance, 479
- addQTermsToExressionTree
 - OSInstance, 478
- AddRef
 - OSReferencedObject, 730
- addSOS
 - SOSVariableBranchingWeights, 966
- addSlackVars
 - LindoSolver, 258
- addSolverOption
 - SolverOptions, 960
- addTimingInformation
 - OSResult, 771
- addVar
 - InitialBasisStatus, 185
 - InitVariableValues, 202
 - InitVariableValuesString, 206, 207
 - IntegerVariableBranchingWeights, 221, 222
 - OtherVariableOption, 891
 - SOSWeights, 969
- addVariable
 - OSInstance, 462
- alignsOnBlockBoundary
 - BaseMatrix, 35
 - ConReferenceMatrixElements, 71
 - ConstantMatrixElements, 77
 - GeneralMatrixElements, 145

- LinearMatrixElements, [266](#)
- MatrixBlock, [277](#)
- MatrixBlocks, [282](#)
- MatrixNode, [305](#)
- MatrixTransformation, [316](#)
- MatrixType, [320](#)
- ObjReferenceMatrixElements, [367](#)
- OSMatrix, [492](#)
- RowReferenceMatrixElements, [938](#)
- VarReferenceMatrixElements, [1026](#)
- allDiffVec
 - OSnLParserData, [650](#)
- app
 - lpoptSolver, [237](#)
- app_
 - CouenneSolver, [100](#)
- areDerivativesImplemented
 - KnitroProblem, [250](#)
- atEquality
 - BasisStatus, [40](#)
- atLower
 - BasisStatus, [40](#)
- atUpper
 - BasisStatus, [40](#)
- availableCPUNumber
 - SystemResult, [987](#)
- availableCPUSpeed
 - SystemResult, [987](#)
- availableDiskSpace
 - SystemResult, [987](#)
- availableMemory
 - SystemResult, [987](#)
- axisDirection
 - OSiLParserData, [427](#)
 - QuadraticCone, [928](#)
- axisDirectionPresent
 - OSiLParserData, [427](#)
- BASE64END
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- BASE64START
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- BASEMATRIXEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- BASEMATRIXENDCOLATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1292](#), [1302](#)
- BASEMATRIXENDROWATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1292](#), [1302](#)
- BASEMATRIXIDXATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- BASEMATRIXSTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- BASEMATRIXSTARTCOLATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1292](#), [1302](#)
- BASEMATRIXSTARTROWATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1292](#), [1302](#)
- BASETRANSDPOSEATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1292](#), [1302](#)
- BASICEND
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- BASICSTART
 - OSParseosil.tab.hpp, [1142](#), [1150](#)
 - OSParseosol.tab.hpp, [1217](#), [1225](#)
 - OSParseosrl.tab.hpp, [1290](#), [1298](#)
- BASISSTATUSEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- BASISSTATUSSTART
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- BASSTATUSEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- BASSTATUSSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- BLOCKCOLIDXATT
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- BLOCKEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)

- OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
- OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- BLOCKROWIDXATT
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- BLOCKSEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- BLOCKSSTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- BLOCKSTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- bADMustReTape
 - OSExpressionTree, [403](#)
- bAMatrixModified
 - OSInstance, [483](#)
- BASE64END
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1256](#)
- BASE64START
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1256](#)
- BASEMATRIXEND
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- BASEMATRIXENDCOLATT
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- BASEMATRIXENDROWATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- BASEMATRIXIDXATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- BASEMATRIXSTART
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- BASETRANSPPOSEATT
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- BASICEND
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1256](#)
- BASICSTART
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1256](#)
- BASISSTATUSEND
 - OSParseosrl.tab.hpp, [1256](#)
- BASISSTATUSSTART
 - OSParseosrl.tab.hpp, [1256](#)
- BASSTATUSEND
 - OSParseosrl.tab.hpp, [1256](#)
- BASSTATUSSTART
 - OSParseosrl.tab.hpp, [1256](#)
- bCallbuildSolverInstance
 - DefaultSolver, [114](#)
- bConstraintsModified
 - OSInstance, [483](#)
- bDeleteArrays
 - DoubleVector, [118](#)
 - ExpandedMatrixBlocks, [130](#)
 - GeneralSparseMatrix, [159](#)
 - IntVector, [229](#)
 - SparseHessianMatrix, [971](#)
 - SparseIntVector, [973](#)
 - SparseJacobianMatrix, [974](#)
 - SparseMatrix, [977](#)
 - SparseVector, [978](#)
- bDestroyNINodes
 - OSExpressionTree, [403](#)
- BLOCKCOLIDXATT
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- BLOCKEND
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- BLOCKROWIDXATT
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- BLOCKSEND
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- BLOCKSSTART
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- BLOCKSTART
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)

- bObjectivesModified
 - OSInstance, [483](#)
- bSetSolverOptions
 - DefaultSolver, [114](#)
- bUseExpTreeForFunEval
 - OSInstance, [483](#)
- bVariablesModified
 - OSInstance, [483](#)
- Base64, [30](#)
 - ~Base64, [31](#)
 - Base64, [31](#)
 - decodeb64, [32](#)
 - encodeb64, [31](#)
- BaseMatrix, [32](#)
 - ~BaseMatrix, [34](#)
 - alignsOnBlockBoundary, [35](#)
 - BaseMatrix, [34](#)
 - baseMatrix, [35](#)
 - baseMatrixEndCol, [36](#)
 - baseMatrixEndRow, [36](#)
 - baseMatrixIdx, [35](#)
 - baseMatrixStartCol, [36](#)
 - baseMatrixStartRow, [36](#)
 - baseTranspose, [36](#)
 - BaseMatrix, [34](#)
 - cloneMatrixNode, [35](#)
 - getMatrixNodeInXML, [35](#)
 - getMatrixType, [34](#)
 - getNodeName, [34](#)
 - getNodeTypes, [34](#)
 - IsEqual, [35](#)
 - scalarMultiplier, [36](#)
 - targetMatrixFirstCol, [36](#)
 - targetMatrixFirstRow, [36](#)
- baseMatrix
 - BaseMatrix, [35](#)
- baseMatrixEndCol
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- baseMatrixEndColPresent
 - OSgLPParserData, [413](#)
- baseMatrixEndRow
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- baseMatrixEndRowPresent
 - OSgLPParserData, [413](#)
- baseMatrixIdx
 - BaseMatrix, [35](#)
 - OSgLPParserData, [412](#)
- baseMatrixIdxPresent
 - OSgLPParserData, [413](#)
- baseMatrixStartCol
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- baseMatrixStartColPresent
 - OSgLPParserData, [413](#)
- baseMatrixStartRow
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- baseMatrixStartRowPresent
 - OSgLPParserData, [413](#)
- baseTranspose
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- baseTransposePresent
 - OSgLPParserData, [413](#)
- basic
 - BasisStatus, [40](#)
- BasisStatus, [36](#)
 - ~BasisStatus, [38](#)
 - addIdx, [39](#)
 - atEquality, [40](#)
 - atLower, [40](#)
 - atUpper, [40](#)
 - basic, [40](#)
 - BasisStatus, [38](#)
 - BasisStatus, [38](#)
 - deepCopyFrom, [38](#)
 - getBasisDense, [40](#)
 - getEI, [39](#)
 - getIntVector, [40](#)
 - getNumberOfEI, [39](#)
 - IsEqual, [38](#)
 - isFree, [41](#)
 - setIntVector, [39](#)
 - setRandom, [38](#)
 - superbasic, [41](#)
 - unknown, [41](#)
- basisStatus
 - ConstraintSolution, [90](#)
 - ObjectiveSolution, [362](#)
 - VariableSolution, [1018](#)
- bb
 - BonminSolver, [49](#)
 - CouenneSolver, [100](#)
- bl
 - OSMatlab, [486](#)
- block
 - MatrixBlocks, [283](#)
- blockColIdx
 - MatrixBlock, [278](#)
 - OSgLPParserData, [414](#)
- blockColIdxPresent
 - OSgLPParserData, [414](#)
- blockColumns
 - ExpandedMatrixBlocks, [131](#)
- blockNumber
 - ExpandedMatrixBlocks, [131](#)

- blockRowIdx
 - MatrixBlock, [278](#)
 - OSgLPParserData, [414](#)
- blockRowIdxPresent
 - OSgLPParserData, [414](#)
- blockRows
 - ExpandedMatrixBlocks, [131](#)
- blocks
 - ExpandedMatrixBlocks, [131](#)
- BonminProblem, [41](#)
 - ~BonminProblem, [44](#)
 - BonminProblem, [44](#)
 - BonminProblem, [44](#)
 - branchingInfo, [45](#)
 - eval_f, [45](#)
 - eval_g, [45](#)
 - eval_grad_f, [45](#)
 - eval_h, [45](#)
 - eval_jac_g, [45](#)
 - finalize_solution, [45](#)
 - get_bounds_info, [45](#)
 - get_constraints_linearity, [44](#)
 - get_nlp_info, [44](#)
 - get_scaling_parameters, [45](#)
 - get_starting_point, [45](#)
 - get_variables_linearity, [44](#)
 - get_variables_types, [44](#)
 - osinstance, [46](#)
 - osoption, [46](#)
 - printSolutionAtEndOfAlgorithm, [46](#)
 - sosConstraints, [45](#)
 - status, [46](#)
- BonminSolver, [46](#)
 - ~BonminSolver, [48](#)
 - bb, [49](#)
 - BonminSolver, [48](#)
 - BonminSolver, [48](#)
 - buildSolverInstance, [49](#)
 - dataEchoCheck, [49](#)
 - m_osilreader, [49](#)
 - m_osolreader, [49](#)
 - setSolverOptions, [49](#)
 - solve, [49](#)
 - status, [49](#)
 - tminlp, [49](#)
 - writeResult, [49](#)
- branchingInfo
 - BonminProblem, [45](#)
- BranchingWeight, [50](#)
 - ~BranchingWeight, [51](#)
 - BranchingWeight, [51](#)
 - BranchingWeight, [51](#)
 - deepCopyFrom, [52](#)
 - idx, [52](#)
 - IsEqual, [51](#)
 - name, [52](#)
 - setRandom, [51](#)
 - value, [52](#)
- browser
 - OSCommandLine, [398](#)
 - osOptionsStruc, [719](#)
- bu
 - OSMatlab, [487](#)
- buildSolverInstance
 - BonminSolver, [49](#)
 - CoinSolver, [56](#)
 - CouenneSolver, [99](#)
 - CsdpSolver, [109](#)
 - DefaultSolver, [113](#)
 - IpoptSolver, [237](#)
 - KnitroSolver, [253](#)
 - LindoSolver, [257](#)
- CATEGORYATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- COEFATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- COLOFFSETSEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- COLOFFSETSSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- COMPONENTSEND
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1279](#)
- COMPONENTSSTART
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1279](#)
- CONEND
 - OSParseosil.tab.hpp, [1133](#), [1143](#), [1151](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1298](#)
- CONESSEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- CONESSTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)

- OSParseosrl.tab.hpp, [1279](#)
- CONREFERENCEELEMENTSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- CONREFERENCEELEMENTSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- CONREFERENCEMATRIIXATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- CONSTANTATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- CONSTANTELEMENTSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- CONSTANTELEMENTSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- CONSTANTMATRIIXATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- CONSTART
 - OSParseosil.tab.hpp, [1133](#), [1143](#), [1151](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1298](#)
- CONSTRAINTSEND
 - OSParseosil.tab.hpp, [1133](#), [1143](#), [1151](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1298](#)
- CONSTRAINTSSTART
 - OSParseosil.tab.hpp, [1133](#), [1143](#), [1151](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1281](#), [1290](#), [1298](#)
- CONTACTEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1288](#)
- CONTACTSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1288](#)
- CONTYPEATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- COSEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- COSSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- CURRENTJOBCEUNTEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- CURRENTJOBCEUNTSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1225](#)
 - OSParseosrl.tab.hpp, [1298](#)
- CURRENTSTATEEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1298](#)
- CURRENTSTATESTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1298](#)
- CATEGORYATT
 - OSParseosol.tab.hpp, [1177](#)
 - OSParseosrl.tab.hpp, [1252](#)
- COEFATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- COIN_HAS_CBC
 - config_default.h, [1040](#)
- COIN_HAS_CGL
 - config_default.h, [1040](#)
- COIN_HAS_CLP
 - config_default.h, [1040](#)
- COIN_HAS_COINUTILS
 - config_default.h, [1040](#)
- COIN_HAS_OSI
 - config_default.h, [1040](#)
- COIN_HAS_SYMPHONY
 - config_default.h, [1040](#)
- COIN_HAS_VOL
 - config_default.h, [1040](#)
- COIN_OS_CHECKLEVEL
 - config_default.h, [1039](#)
- COIN_OS_VERBOSITY
 - config_default.h, [1039](#)
- COLOFFSETSEND
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- COLOFFSETSSTART
 - OSParseosil.tab.hpp, [1122](#)

- OSParseosol.tab.hpp, 1197
- OSParseosrl.tab.hpp, 1270
- COMPONENTSEND
 - OSParseosil.tab.hpp, 1110
- COMPONENTSSTART
 - OSParseosil.tab.hpp, 1110
- CONEND
 - OSParseosil.tab.hpp, 1114
 - OSParseosol.tab.hpp, 1189
 - OSParseosrl.tab.hpp, 1257
- CONESEND
 - OSParseosil.tab.hpp, 1108
- CONESSTART
 - OSParseosil.tab.hpp, 1108
- CONSTANTATT
 - OSParseosil.tab.hpp, 1119
 - OSParseosol.tab.hpp, 1193
 - OSParseosrl.tab.hpp, 1266
- CONSTANTELEMENTSEND
 - OSParseosil.tab.hpp, 1120
 - OSParseosol.tab.hpp, 1195
 - OSParseosrl.tab.hpp, 1268
- CONSTANTMATRIXIDXATT
 - OSParseosil.tab.hpp, 1113
- CONSTART
 - OSParseosil.tab.hpp, 1114
 - OSParseosol.tab.hpp, 1189
 - OSParseosrl.tab.hpp, 1256
- CONSTRAINTSEND
 - OSParseosil.tab.hpp, 1114
 - OSParseosol.tab.hpp, 1189
 - OSParseosrl.tab.hpp, 1257
- CONSTRAINTSSTART
 - OSParseosil.tab.hpp, 1113
 - OSParseosol.tab.hpp, 1189
 - OSParseosrl.tab.hpp, 1257
- CONTACTEND
 - OSParseosol.tab.hpp, 1182
- CONTACTSTART
 - OSParseosol.tab.hpp, 1182
- CONTYPEATT
 - OSParseosol.tab.hpp, 1178
 - OSParseosrl.tab.hpp, 1252
- COSEND
 - OSParseosil.tab.hpp, 1125
 - OSParseosol.tab.hpp, 1200
 - OSParseosrl.tab.hpp, 1273
- COSSTART
 - OSParseosil.tab.hpp, 1125
 - OSParseosol.tab.hpp, 1200
 - OSParseosrl.tab.hpp, 1273
- CPUNumber, 100
 - ~CPUNumber, 102
- CPUNumber, 102
- CPUNumber, 102
 - deepCopyFrom, 102
 - description, 103
 - IsEqual, 102
 - setRandom, 102
 - value, 103
- CPUSpeed, 103
 - ~CPUSpeed, 104
- CPUSpeed, 104
- CPUSpeed, 104
 - deepCopyFrom, 105
 - description, 105
 - IsEqual, 105
 - setRandom, 105
 - unit, 105
 - value, 105
- CURRENTJOBCEUNTEND
 - OSParseosrl.tab.hpp, 1257
- CURRENTJOBCEUNTSTART
 - OSParseosrl.tab.hpp, 1257
- CURRENTSTATEEND
 - OSParseosrl.tab.hpp, 1257
- CURRENTSTATESTART
 - OSParseosrl.tab.hpp, 1257
- calculateAllConstraintFunctionGradients
 - OSInstance, 476
- calculateAllConstraintFunctionValues
 - OSInstance, 474, 475
- calculateAllObjectiveFunctionGradients
 - OSInstance, 476
- calculateAllObjectiveFunctionValues
 - OSInstance, 475
- calculateConstraintFunctionGradient
 - OSInstance, 476
- calculateFunction
 - OSnLNode, 557
 - OSnLNodeAbs, 561
 - OSnLNodeAllDiff, 564
 - OSnLNodeCos, 567
 - OSnLNodeDivide, 570
 - OSnLNodeE, 574
 - OSnLNodeErf, 577
 - OSnLNodeExp, 579
 - OSnLNodeIf, 582
 - OSnLNodeLn, 585
 - OSnLNodeMatrixDeterminant, 588
 - OSnLNodeMatrixToScalar, 591
 - OSnLNodeMatrixTrace, 594
 - OSnLNodeMax, 597
 - OSnLNodeMin, 600
 - OSnLNodeMinus, 603
 - OSnLNodeNegate, 606
 - OSnLNodeNumber, 610
 - OSnLNodePI, 614

- OSnLNodePlus, [617](#)
- OSnLNodePower, [619](#)
- OSnLNodeProduct, [622](#)
- OSnLNodeSin, [625](#)
- OSnLNodeSqrt, [628](#)
- OSnLNodeSquare, [631](#)
- OSnLNodeSum, [634](#)
- OSnLNodeTimes, [637](#)
- OSnLNodeVariable, [641](#)
- ScalarExpressionTree, [943](#)
- calculateFunctionValue
 - OSInstance, [474](#)
- calculateHessian
 - OSInstance, [478](#)
- calculateLagrangianHessian
 - OSInstance, [477](#)
- calculateObjectiveFunctionGradient
 - OSInstance, [477](#)
- category
 - OtherConstraintOption, [849](#)
 - OtherObjectiveOption, [858](#)
 - OtherSolutionResult, [883](#)
 - OtherVariableOption, [892](#)
 - SolverOption, [956](#)
 - SolverOutput, [963](#)
 - TimeMeasurement, [1002](#)
- categoryAttribute
 - OSnLParserData, [646](#)
 - OSoLParserData, [667](#)
 - OSrLParserData, [822](#)
- categoryAttributePresent
 - OSnLParserData, [646](#)
 - OSoLParserData, [667](#)
 - OSrLParserData, [821](#)
- cloneExprNode
 - ExprNode, [136](#)
 - OSnLMNodeDiagonalMatrixFromVector, [511](#)
 - OSnLMNodeIdentityMatrix, [513](#)
 - OSnLMNodeMatrixCon, [515](#)
 - OSnLMNodeMatrixDiagonal, [517](#)
 - OSnLMNodeMatrixDotTimes, [519](#)
 - OSnLMNodeMatrixInverse, [521](#)
 - OSnLMNodeMatrixLowerTriangle, [523](#)
 - OSnLMNodeMatrixMinus, [526](#)
 - OSnLMNodeMatrixNegate, [528](#)
 - OSnLMNodeMatrixObj, [530](#)
 - OSnLMNodeMatrixPlus, [532](#)
 - OSnLMNodeMatrixProduct, [534](#)
 - OSnLMNodeMatrixReference, [537](#)
 - OSnLMNodeMatrixScalarTimes, [539](#)
 - OSnLMNodeMatrixSubmatrixAt, [541](#)
 - OSnLMNodeMatrixSum, [543](#)
 - OSnLMNodeMatrixTimes, [545](#)
 - OSnLMNodeMatrixTranspose, [547](#)
 - OSnLMNodeMatrixUpperTriangle, [549](#)
 - OSnLMNodeMatrixVar, [552](#)
 - OSnLNodeAbs, [562](#)
 - OSnLNodeAllDiff, [565](#)
 - OSnLNodeCos, [568](#)
 - OSnLNodeDivide, [571](#)
 - OSnLNodeE, [574](#)
 - OSnLNodeErf, [577](#)
 - OSnLNodeExp, [580](#)
 - OSnLNodeIf, [583](#)
 - OSnLNodeLn, [586](#)
 - OSnLNodeMatrixDeterminant, [589](#)
 - OSnLNodeMatrixToScalar, [592](#)
 - OSnLNodeMatrixTrace, [595](#)
 - OSnLNodeMax, [598](#)
 - OSnLNodeMin, [601](#)
 - OSnLNodeMinus, [604](#)
 - OSnLNodeNegate, [607](#)
 - OSnLNodeNumber, [610](#)
 - OSnLNodePI, [614](#)
 - OSnLNodePlus, [617](#)
 - OSnLNodePower, [620](#)
 - OSnLNodeProduct, [623](#)
 - OSnLNodeSin, [626](#)
 - OSnLNodeSqrt, [629](#)
 - OSnLNodeSquare, [632](#)
 - OSnLNodeSum, [635](#)
 - OSnLNodeTimes, [638](#)
 - OSnLNodeVariable, [642](#)
- cloneMatrixNode
 - BaseMatrix, [35](#)
 - ConReferenceMatrixElements, [72](#)
 - ConstantMatrixElements, [78](#)
 - GeneralMatrixElements, [146](#)
 - LinearMatrixElements, [266](#)
 - MatrixBlock, [277](#)
 - MatrixBlocks, [282](#)
 - MatrixNode, [305](#)
 - MatrixTransformation, [317](#)
 - ObjReferenceMatrixElements, [368](#)
 - OSMatrix, [493](#)
 - RowReferenceMatrixElements, [939](#)
 - VarReferenceMatrixElements, [1026](#)
- coef
 - LinearMatrixElementTerm, [268](#)
 - Objective, [353](#)
 - OSnLNodeVariable, [642](#)
 - QuadraticTerm, [930](#)
- coefficients
 - QuadraticTerms, [931](#)
- Coin, [30](#)
- CoinSolver, [52](#)
 - ~CoinSolver, [55](#)
 - buildSolverInstance, [56](#)

- CoinSolver, 55
- CoinSolver, 55
- dataEchoCheck, 56
- getCoinSolverType, 56
- m_osilreader, 57
- m_osolreader, 57
- osiSolver, 57
- setCoinPackedMatrix, 56
- setSolverOptions, 56
- solve, 55
- writeResult, 56, 57
- colIdx
 - LinearConstraintCoefficients, 261
- colOffsets
 - ExpandedMatrixBlocks, 131
 - MatrixBlocks, 283
 - OSgLPParserData, 411
- ComparePointers
 - OSSmartPtr.hpp, 1353
- CompletelyPositiveMatricesCone, 57
 - ~CompletelyPositiveMatricesCone, 59
 - CompletelyPositiveMatricesCone, 59
 - CompletelyPositiveMatricesCone, 59
 - deepCopyFrom, 59
 - getConeInXML, 59
 - getConeName, 59
 - IsEqual, 59
 - setRandom, 59
- components
 - IntersectionCone, 226
- con
 - Constraints, 87
 - DualVariableValues, 124
 - InitConstraintValues, 173
 - InitDualVariableValues, 179
 - OtherConstraintOption, 849
 - OtherConstraintResult, 853
 - TimeDomainStageConstraints, 993
- con_body
 - CouenneSolver, 100
- conReference
 - ConReferenceMatrixElement, 68
- ConReferenceMatrixElement, 67
 - ~ConReferenceMatrixElement, 68
 - conReference, 68
 - ConReferenceMatrixElement, 68
 - ConReferenceMatrixElement, 68
 - deepCopyFrom, 68
 - IsEqual, 68
 - setRandom, 68
 - value, 69
 - valueType, 69
- ConReferenceMatrixElements, 69
 - ~ConReferenceMatrixElements, 71
- alignsOnBlockBoundary, 71
- cloneMatrixNode, 72
- ConReferenceMatrixElements, 71
- ConReferenceMatrixElements, 71
- deepCopyFrom, 72
- getMatrixNodeInXML, 71
- getMatrixType, 71
- getNodeName, 71
- getNodeName, 71
- IsEqual, 72
- setRandom, 72
- values, 72
- conReferenceMatrixIdx
 - MatrixCon, 285
 - OSiLPParserData, 430
- conReferenceMatrixIdxPresent
 - OSiLPParserData, 429
- ConReferenceMatrixValues, 73
 - ~ConReferenceMatrixValues, 74
 - ConReferenceMatrixValues, 74
 - ConReferenceMatrixValues, 74
 - deepCopyFrom, 74
 - el, 74
 - IsEqual, 74
 - setRandom, 74
- conType
 - OtherConstraintOption, 850
 - OtherConstraintResult, 853
- conTypeAttribute
 - OSnLPParserData, 647
 - OSoLPParserData, 668
 - OSrLPParserData, 823
- conTypeAttributePresent
 - OSnLPParserData, 646
 - OSoLPParserData, 668
 - OSrLPParserData, 822
- conVals
 - SparseJacobianMatrix, 975
- Cone, 60
 - ~Cone, 63
 - Cone, 63
 - coneType, 64
 - deepCopyFrom, 64
 - getConeInXML, 63
 - getConeName, 63
 - idx, 65
 - IsEqual, 63
 - name, 64
 - numberOfColumns, 64
 - numberOfOtherIndexes, 64
 - numberOfRows, 64
 - otherIndexes, 64
 - setRandom, 63
- cone

- Cones, [67](#)
- coneCounter
 - OSILParserData, [426](#)
- coneType
 - Cone, [64](#)
 - DualCone, [121](#)
 - IntersectionCone, [226](#)
 - PolarCone, [912](#)
 - PolyhedralCone, [916](#)
 - ProductCone, [923](#)
 - QuadraticCone, [928](#)
 - RotatedQuadraticCone, [935](#)
 - SemidefiniteCone, [947](#)
- Cones, [65](#)
 - ~Cones, [66](#)
 - cone, [67](#)
 - Cones, [66](#)
 - deepCopyFrom, [66](#)
 - IsEqual, [66](#)
 - numberOfCones, [67](#)
 - setRandom, [66](#)
- cones
 - InstanceData, [216](#)
- config_default.h
 - COIN_HAS_CBC, [1040](#)
 - COIN_HAS_CGL, [1040](#)
 - COIN_HAS_CLP, [1040](#)
 - COIN_HAS_OSI, [1040](#)
 - COIN_HAS_SYMPHONY, [1040](#)
 - COIN_HAS_VOL, [1040](#)
 - COIN_OS_VERBOSITY, [1039](#)
- config_os_default.h
 - OS_HAS_AS_L, [1041](#)
 - OS_HAS_BONMIN, [1041](#)
 - OS_HAS_COUENNE, [1041](#)
 - OS_HAS_CPPAD, [1041](#)
 - OS_HAS_DYLP, [1041](#)
 - OS_HAS_GLPK, [1041](#)
 - OS_HAS_IPOPT, [1041](#)
 - OS_HAS_SYMPHONY, [1041](#)
 - OS_HAS_VOL, [1041](#)
 - OS_VERSION, [1041](#)
- configFile
 - OSCommandLine, [396](#)
 - osOptionsStruc, [717](#)
- ConstPtr
 - OSSmartPtr, [836](#)
 - OSSmartPtr.hpp, [1352](#)
- constant
 - Constraint, [82](#)
 - LinearMatrixElement, [263](#)
 - Objective, [353](#)
- ConstantMatrixElements, [75](#)
 - ~ConstantMatrixElements, [77](#)
- alignsOnBlockBoundary, [77](#)
- cloneMatrixNode, [78](#)
- ConstantMatrixElements, [77](#)
- ConstantMatrixElements, [77](#)
- deepCopyFrom, [78](#)
- getMatrixNodeInXML, [77](#)
- getMatrixType, [77](#)
- getNodeName, [77](#)
- getNodeName, [77](#)
- IsEqual, [78](#)
- setRandom, [78](#)
- values, [78](#)
- constantMatrixIdx
 - MatrixObj, [309](#)
 - OSILParserData, [430](#)
- constantMatrixIdxPresent
 - OSILParserData, [429](#)
- ConstantMatrixValues, [79](#)
 - ~ConstantMatrixValues, [80](#)
 - ConstantMatrixValues, [80](#)
 - ConstantMatrixValues, [80](#)
 - deepCopyFrom, [80](#)
 - el, [81](#)
 - IsEqual, [80](#)
 - setRandom, [80](#)
- Constraint, [81](#)
 - ~Constraint, [82](#)
 - constant, [82](#)
 - Constraint, [82](#)
 - IsEqual, [82](#)
 - lb, [82](#)
 - name, [82](#)
 - ub, [82](#)
- ConstraintOption, [83](#)
 - ~ConstraintOption, [84](#)
 - addOther, [85](#)
 - ConstraintOption, [84](#)
 - ConstraintOption, [84](#)
 - deepCopyFrom, [85](#)
 - initialBasisStatus, [86](#)
 - initialConstraintValues, [85](#)
 - initialDualValues, [85](#)
 - IsEqual, [84](#)
 - numberOfOtherConstraintOptions, [85](#)
 - other, [86](#)
 - setOther, [85](#)
 - setRandom, [84](#)
- ConstraintSolution, [88](#)
 - ~ConstraintSolution, [89](#)
 - basisStatus, [90](#)
 - ConstraintSolution, [89](#)
 - ConstraintSolution, [89](#)
 - dualValues, [90](#)
 - IsEqual, [89](#)

- numberOfOtherConstraintResults, 90
 - other, 90
 - setRandom, 89
- Constraints, 86
 - ~Constraints, 87
 - con, 87
 - Constraints, 87
 - IsEqual, 87
 - numberOfConstraints, 87
- constraints
 - InstanceData, 215
 - OptimizationOption, 376
 - OptimizationSolution, 382
 - TimeDomainStage, 991
- constraintsPresent
 - OSoLParserData, 663
- constructADTape
 - OSnLNode, 557
 - OSnLNodeAbs, 562
 - OSnLNodeAllDiff, 565
 - OSnLNodeCos, 568
 - OSnLNodeDivide, 571
 - OSnLNodeE, 574
 - OSnLNodeErf, 577
 - OSnLNodeExp, 580
 - OSnLNodeIf, 583
 - OSnLNodeLn, 586
 - OSnLNodeMatrixDeterminant, 589
 - OSnLNodeMatrixToScalar, 592
 - OSnLNodeMatrixTrace, 595
 - OSnLNodeMax, 598
 - OSnLNodeMin, 601
 - OSnLNodeMinus, 604
 - OSnLNodeNegate, 607
 - OSnLNodeNumber, 611
 - OSnLNodePI, 614
 - OSnLNodePlus, 617
 - OSnLNodePower, 620
 - OSnLNodeProduct, 623
 - OSnLNodeSin, 626
 - OSnLNodeSqrt, 629
 - OSnLNodeSquare, 632
 - OSnLNodeSum, 635
 - OSnLNodeTimes, 638
 - OSnLNodeVariable, 642
- contact
 - GeneralOption, 153
- ContactOption, 90
 - ~ContactOption, 92
 - ContactOption, 92
 - ContactOption, 92
 - deepCopyFrom, 92
 - IsEqual, 92
 - setRandom, 92
 - transportType, 93
 - value, 93
- contactPresent
 - OSoLParserData, 659
- convertLinearConstraintCoefficientMatrixToTheOther-
Major
 - MathUtil, 272
- convertSolverNameToLowerCase
 - OSCommandLine, 395
- convertSolverNameToUpperCase
 - OSCommandLine, 395
- CopositiveMatricesCone, 93
 - ~CopositiveMatricesCone, 95
 - CopositiveMatricesCone, 95
 - CopositiveMatricesCone, 95
 - deepCopyFrom, 95
 - getConeInXML, 95
 - getConeName, 95
 - IsEqual, 95
 - setRandom, 95
- copyLinearConstraintCoefficients
 - OSInstance, 466
- copyNodeAndDescendants
 - OSnLMNode, 509
 - OSnLNode, 559
- Couenne, 30
- couenne
 - CouenneSolver, 99
- CouenneSolver, 96
 - ~CouenneSolver, 99
 - app_, 100
 - bb, 100
 - buildSolverInstance, 99
 - con_body, 100
 - couenne, 99
 - CouenneSolver, 99
 - CouenneSolver, 99
 - dataEchoCheck, 99
 - m_osilreader, 99
 - m_osolreader, 99
 - obj_body, 100
 - setSolverOptions, 99
 - solve, 99
 - status, 100
 - tminlp, 100
 - writeResult, 99
- createConstructorTreeFromPrefix
 - OSMatrix, 491
- createExpressionTreeFromPostfix
 - OSnLMNode, 508
 - OSnLNode, 558
- createExpressionTreeFromPrefix
 - OSnLMNode, 507
 - OSnLNode, 557

- createFormDataUpload
 - WSUtil, [1037](#)
- createOSADFun
 - OSInstance, [479](#)
- createOSInstance
 - OSgams2osil, [405](#)
 - OSMatlab, [486](#)
 - OSmps2osil, [499](#)
- createOSObjects
 - OSmps2OS, [496](#)
 - OSnl2OS, [503](#)
- createSOAPMessage
 - WSUtil, [1036](#)
- CsdpSolver, [105](#)
 - ~CsdpSolver, [108](#)
 - buildSolverInstance, [109](#)
 - CsdpSolver, [108](#)
 - CsdpSolver, [108](#)
 - dataEchoCheck, [109](#)
 - m_osilreader, [109](#)
 - m_osolreader, [109](#)
 - setSolverOptions, [109](#)
 - solve, [109](#)
 - verifyForm, [109](#)
- currentJobCount
 - ServiceResult, [953](#)
- currentSOS
 - OSoLParserData, [665](#)
- currentState
 - ServiceResult, [953](#)
- DEPENDENCIESEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- DEPENDENCIESSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- DESCRIPTIONATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- DIRECTIONEND
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- DIRECTIONSTART
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- DIRECTORIESTODELETEEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1216](#)
- OSParseosrl.tab.hpp, [1289](#)
- DIRECTORIESTODELETESTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- DIRECTORIESTOMAKEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- DIRECTORIESTOMAKESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- DISTORTIONMATRIXIDXATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1279](#)
- DIVIDEEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- DIVIDESTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- DOUBLE
 - OSParseosil.tab.hpp, [1131](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- DUALCONEEND
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- DUALCONESTART
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- DUALVALUESEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1298](#)
- DUALVALUESSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1298](#)
- DUMMY
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- DEPENDENCIESEND
 - OSParseosol.tab.hpp, [1184](#)
- DEPENDENCIESSTART
 - OSParseosol.tab.hpp, [1184](#)
- DESCRIPTIONATT

- OSParseosol.tab.hpp, 1178
- OSParseosrl.tab.hpp, 1252
- DIRECTIONEND
 - OSParseosil.tab.hpp, 1110
- DIRECTIONSTART
 - OSParseosil.tab.hpp, 1110
- DIRECTORIESTOMAKEEND
 - OSParseosol.tab.hpp, 1185
- DIVIDEEND
 - OSParseosil.tab.hpp, 1124
 - OSParseosol.tab.hpp, 1199
 - OSParseosrl.tab.hpp, 1272
- DIVIDESTART
 - OSParseosil.tab.hpp, 1124
 - OSParseosol.tab.hpp, 1199
 - OSParseosrl.tab.hpp, 1272
- DOUBLE
 - OSParseosil.tab.hpp, 1107
 - OSParseosol.tab.hpp, 1175
 - OSParseosrl.tab.hpp, 1249
- DUALCONEEND
 - OSParseosil.tab.hpp, 1109
- DUALCONESTART
 - OSParseosil.tab.hpp, 1109
- DUALVALUESEND
 - OSParseosrl.tab.hpp, 1257
- DUALVALUESSTART
 - OSParseosrl.tab.hpp, 1257
- DUMMY
 - OSParseosil.tab.hpp, 1122
 - OSParseosol.tab.hpp, 1197
 - OSParseosrl.tab.hpp, 1270
- dat
 - OSCommandLine, 398
 - osOptionsStruc, 718
- datFile
 - OSCommandLine, 398
 - osOptionsStruc, 718
- dataEchoCheck
 - BonminSolver, 49
 - CoinSolver, 56
 - CouenneSolver, 99
 - CsdpSolver, 109
 - IpoptSolver, 237
 - KnitroSolver, 254
 - LindoSolver, 258
- deSOAPify
 - WSUtil, 1036
- decodeb64
 - Base64, 32
- deepCopyFrom
 - BasisStatus, 38
 - BranchingWeight, 52
 - CompletelyPositiveMatricesCone, 59
 - Cone, 64
 - Cones, 66
 - ConReferenceMatrixElement, 68
 - ConReferenceMatrixElements, 72
 - ConReferenceMatrixValues, 74
 - ConstantMatrixElements, 78
 - ConstantMatrixValues, 80
 - ConstraintOption, 85
 - ContactOption, 92
 - CopositiveMatricesCone, 95
 - CPUNumber, 102
 - CPUSpeed, 105
 - DirectoriesAndFiles, 116
 - DualCone, 121
 - GeneralFileHeader, 142
 - GeneralMatrixElements, 146
 - GeneralMatrixValues, 148
 - GeneralOption, 152
 - InitBasStatus, 168
 - InitConstraintValues, 172
 - InitConValue, 175
 - InitDualVariableValues, 178
 - InitDualVarValue, 181
 - InitialBasisStatus, 185
 - InitObjBound, 187
 - InitObjectiveBounds, 191
 - InitObjectiveValues, 195
 - InitObjValue, 198
 - InitVariableValues, 202
 - InitVariableValuesString, 206
 - InitVarValue, 209
 - InitVarValueString, 212
 - InstanceLocationOption, 218
 - IntegerVariableBranchingWeights, 221
 - IntersectionCone, 225
 - IntVector, 228
 - JobDependencies, 239
 - JobOption, 243
 - LinearMatrixElement, 262
 - LinearMatrixElements, 267
 - LinearMatrixElementTerm, 268
 - LinearMatrixValues, 270
 - Matrices, 274
 - MatrixBlock, 278
 - MatrixBlocks, 282
 - MatrixNode, 306
 - MatrixProgramming, 313
 - MatrixTransformation, 317
 - MatrixType, 322
 - NonnegativeCone, 347
 - NonpositiveCone, 350
 - ObjectiveOption, 356
 - ObjReferenceMatrixElements, 368
 - ObjReferenceMatrixValues, 370

- OptimizationOption, 376
- OrthantCone, 390
- OSMatrix, 494
- OSOption, 686
- OtherConOption, 842
- OtherConstraintOption, 848
- OtherObjectiveOption, 856
- OtherObjOption, 864
- OtherOption, 869
- OtherOptionEnumeration, 872
- OtherOptions, 875
- OtherVariableOption, 891
- OtherVarOption, 900
- PathPair, 905
- PathPairs, 908
- PolarCone, 911
- PolyhedralCone, 915
- Processes, 918
- ProductCone, 922
- QuadraticCone, 927
- RotatedQuadraticCone, 934
- RowReferenceMatrixElements, 939
- SemidefiniteCone, 946
- ServiceOption, 950
- SolverOption, 956
- SolverOptions, 960
- SOSVariableBranchingWeights, 966
- SOSWeights, 969
- StorageCapacity, 981
- SystemOption, 984
- TimeSpan, 1004
- VariableOption, 1012
- VarReferenceMatrixElements, 1027
- VarReferenceMatrixValues, 1029
- DefaultSolver, 110
 - ~DefaultSolver, 113
 - bCallbuildSolverInstance, 114
 - bSetSolverOptions, 114
 - buildSolverInstance, 113
 - DefaultSolver, 113
 - DefaultSolver, 113
 - osil, 113
 - osinstance, 113
 - osol, 113
 - osoption, 113
 - osresult, 114
 - osrl, 113
 - sSolverName, 114
 - setSolverOptions, 113
 - solve, 113
- DeleteChannel
 - OSOutput, 724
- dependencies
 - JobOption, 243
- dependenciesPresent
 - OSoLParserData, 660
- description
 - CPUNumber, 103
 - CPUSpeed, 105
 - GeneralFileHeader, 142
 - GeneralStatus, 162
 - GeneralSubstatus, 164
 - MinCPUNumber, 332
 - MinCPUSpeed, 334
 - MinDiskSpace, 336
 - MinMemorySize, 338
 - OptimizationSolutionStatus, 385
 - OptimizationSolutionSubstatus, 387
 - OSgLPParserData, 410
 - OtherConstraintOption, 849
 - OtherConstraintResult, 853
 - OtherObjectiveOption, 858
 - OtherObjectiveResult, 861
 - OtherOption, 869
 - OtherOptionEnumeration, 872
 - OtherResult, 878
 - OtherSolutionResult, 883
 - OtherVariableOption, 892
 - OtherVariableResult, 895
 - OtherVariableResultStruct, 897
 - SolverOption, 956
 - SolverOutput, 963
 - StorageCapacity, 981
 - TimeMeasurement, 1002
- descriptionAttribute
 - OSnLPParserData, 647
 - OSoLParserData, 669
 - OSrLPParserData, 822
- descriptionAttributePresent
 - OSnLPParserData, 647
 - OSoLParserData, 669
 - OSrLPParserData, 821
- descriptionPresent
 - OSgLPParserData, 410
- direction
 - OS_AMPL_SUFFIX, 391
- DirectoriesAndFiles, 114
 - ~DirectoriesAndFiles, 116
 - addPath, 117
 - deepCopyFrom, 116
 - DirectoriesAndFiles, 116
 - DirectoriesAndFiles, 116
 - IsEqual, 116
 - numberOfPaths, 117
 - path, 117
 - setPath, 117
 - setRandom, 116
- directoriesToDelete

- JobOption, 244
- directoriesToDeletePresent
 - OSoLParserData, 662
- directoriesToMake
 - JobOption, 243
- directoriesToMakePresent
 - OSoLParserData, 661
- display
 - ExpandedMatrixBlocks, 130
 - GeneralSparseMatrix, 158
 - SparseMatrix, 976
- distortionMatrix
 - OSiLParserData, 427
- distortionMatrixIdx
 - QuadraticCone, 928
 - RotatedQuadraticCone, 935
- distortionMatrixPresent
 - OSiLParserData, 427
- DoubleVector, 117
 - ~DoubleVector, 118
 - bDeleteArrays, 118
 - DoubleVector, 118
 - DoubleVector, 118
 - el, 118
 - IsEqual, 118
 - numberOfEl, 118
- DualCone, 118
 - ~DualCone, 120
 - coneType, 121
 - deepCopyFrom, 121
 - DualCone, 120
 - DualCone, 120
 - getConeName, 120
 - idx, 122
 - IsEqual, 120
 - numberOfColumns, 121
 - numberOfOtherIndexes, 121
 - numberOfRows, 121
 - otherIndexes, 121
 - referenceConelIdx, 122
 - setRandom, 121
- dualValPair
 - OSrLParserData, 826
- dualVals
 - OSResult, 811
 - OSrLParserData, 826
- dualValues
 - ConstraintSolution, 90
- DualVarValue, 124
 - ~DualVarValue, 126
 - DualVarValue, 126
 - DualVarValue, 126
 - idx, 126
 - IsEqual, 126

- name, 126
- setRandom, 126
- value, 127
- DualVariableValues, 122
 - ~DualVariableValues, 123
 - con, 124
 - DualVariableValues, 123
 - DualVariableValues, 123
 - IsEqual, 124
 - numberOfCon, 124
 - setRandom, 124
- duplicateExpressionTreesMap
 - OSInstance, 479
- dval
 - YYSTYPE, 1038
- EEND
 - OSParseosil.tab.hpp, 1137, 1147, 1156
 - OSParseosol.tab.hpp, 1212, 1222, 1231
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- ELEMENTSEND
 - OSParseosil.tab.hpp, 1135, 1145, 1154
 - OSParseosol.tab.hpp, 1210, 1220, 1229
 - OSParseosrl.tab.hpp, 1283, 1293, 1302
- ELEMENTSSTART
 - OSParseosil.tab.hpp, 1135, 1145, 1154
 - OSParseosol.tab.hpp, 1210, 1220, 1229
 - OSParseosrl.tab.hpp, 1283, 1292, 1302
- ELEMENTTEXT
 - OSParseosil.tab.hpp, 1130, 1138, 1148
 - OSParseosol.tab.hpp, 1205, 1213, 1223
 - OSParseosrl.tab.hpp, 1278, 1286, 1296
- ELEND
 - OSParseosil.tab.hpp, 1134, 1143, 1151
 - OSParseosol.tab.hpp, 1209, 1218, 1226
 - OSParseosrl.tab.hpp, 1282, 1291, 1299
- ELSTART
 - OSParseosil.tab.hpp, 1134, 1143, 1151
 - OSParseosol.tab.hpp, 1209, 1218, 1226
 - OSParseosrl.tab.hpp, 1282, 1291, 1298
- EMPTYBASETRANSPOSEATT
 - OSParseosil.tab.hpp, 1135, 1145, 1154
 - OSParseosol.tab.hpp, 1210, 1220, 1229
 - OSParseosrl.tab.hpp, 1283, 1292, 1302
- EMPTYCATEGORYATT
 - OSParseosil.tab.hpp, 1140, 1149
 - OSParseosol.tab.hpp, 1214, 1224
 - OSParseosrl.tab.hpp, 1287, 1297
- EMPTYCONTYPEATT
 - OSParseosil.tab.hpp, 1139, 1149
 - OSParseosol.tab.hpp, 1214, 1224
 - OSParseosrl.tab.hpp, 1287, 1297
- EMPTYDESCRIPTIONATT
 - OSParseosil.tab.hpp, 1140, 1149

- OSParseosol.tab.hpp, [1214](#), [1224](#)
- OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYENUMTYPEATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYIDATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1278](#)
- EMPTYINCLUDEDIAGONALATT
 - OSParseosil.tab.hpp, [1138](#), [1148](#), [1158](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)
- EMPTYLBDUALVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1287](#)
- EMPTYLBVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1287](#)
- EMPTYNAMEATT
 - OSParseosil.tab.hpp, [1134](#), [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1297](#)
- EMPTYNEGATIVEPATTERNATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- EMPTYOBJTYPEATT
 - OSParseosil.tab.hpp, [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYROWMAJORATT
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- EMPTYSEMIDEFINITENESSATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- EMPTYSHAPEATT
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- EMPTYSOLVERATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- EMPTYSYMMETRYATT
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- EMPTYTARGETOBJECTIVENAMEATT
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1297](#)
- EMPTYTYPEATT
 - OSParseosil.tab.hpp, [1134](#), [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1297](#)
- EMPTYUBDUALVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- EMPTYUBVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1287](#)
- EMPTYUNITATT
 - OSParseosil.tab.hpp, [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYVALUEATT
 - OSParseosil.tab.hpp, [1140](#), [1150](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYVARTYPEATT
 - OSParseosil.tab.hpp, [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- EMPTYWEIGHTATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- EMPTYWEIGHTEDOBJECTIVESATT
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1297](#)
- ENDOFELEMENT
 - OSParseosil.tab.hpp, [1131](#), [1139](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- ENDTIMEEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- ENDTIMESTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- ENUM_BASIS_STATUS_NUMBER_OF_STATES
 - OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_atEquality
 - OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_atLower
 - OSParameters.h, [1341](#)

- ENUM_BASIS_STATUS_atUpper
 OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_basic
 OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_isFree
 OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_superbasic
 OSParameters.h, [1341](#)
- ENUM_BASIS_STATUS_unknown
 OSParameters.h, [1341](#)
- ENUM_COMBINE_ARRAYS_ignore
 OSParameters.h, [1344](#)
- ENUM_COMBINE_ARRAYS_merge
 OSParameters.h, [1344](#)
- ENUM_COMBINE_ARRAYS_replace
 OSParameters.h, [1344](#)
- ENUM_COMBINE_ARRAYS_throw
 OSParameters.h, [1344](#)
- ENUM_CONE_TYPE_completelyPositiveMatrices
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_copositiveMatrices
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_dual
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_hyperbolicity
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_intersection
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_moments
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_nonnegative
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_nonnegativePolynomials
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_nonpositive
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_normed
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_orthant
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_polar
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_polyhedral
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_product
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_quadratic
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_rotatedNormed
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_rotatedQuadratic
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_semidefinite
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_sumOfSquaresPolynomials
 OSParameters.h, [1345](#)
- ENUM_CONE_TYPE_unknown
 OSParameters.h, [1345](#)
- ENUM_CONREFERENCE_VALUETYPE_shortage
 OSParameters.h, [1343](#)
- ENUM_CONREFERENCE_VALUETYPE_status
 OSParameters.h, [1343](#)
- ENUM_CONREFERENCE_VALUETYPE_surplus
 OSParameters.h, [1343](#)
- ENUM_CONREFERENCE_VALUETYPE_value
 OSParameters.h, [1343](#)
- ENUM_CPUSPEEDUNIT_flops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_gigaflops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_gigahertz
 OSParameters.h, [1338](#)
- ENUM_CPUSPEEDUNIT_hertz
 OSParameters.h, [1338](#)
- ENUM_CPUSPEEDUNIT_kiloflops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_kilohertz
 OSParameters.h, [1338](#)
- ENUM_CPUSPEEDUNIT_megaflops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_megahertz
 OSParameters.h, [1338](#)
- ENUM_CPUSPEEDUNIT_petaflops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_teraflops
 OSParameters.h, [1339](#)
- ENUM_CPUSPEEDUNIT_terahertz
 OSParameters.h, [1338](#)
- ENUM_GENERAL_RESULT_STATUS_error
 OSParameters.h, [1341](#)
- ENUM_GENERAL_RESULT_STATUS_normal
 OSParameters.h, [1341](#)
- ENUM_GENERAL_RESULT_STATUS_warning
 OSParameters.h, [1341](#)
- ENUM_JOB_STATUS_finished
 OSParameters.h, [1341](#)
- ENUM_JOB_STATUS_killed
 OSParameters.h, [1341](#)
- ENUM_JOB_STATUS_running
 OSParameters.h, [1341](#)
- ENUM_JOB_STATUS_unknown
 OSParameters.h, [1341](#)
- ENUM_JOB_STATUS_waiting
 OSParameters.h, [1341](#)
- ENUM_LOCATIONTYPE_ftp
 OSParameters.h, [1340](#)
- ENUM_LOCATIONTYPE_http
 OSParameters.h, [1340](#)

- ENUM_LOCATIONTYPE_local
 OSParameters.h, [1340](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_baseMatrix
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_block
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_blocks
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_conRef-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_constant-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_general-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_linearElements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_matrix
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_objRef-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_rowRef-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_transformation
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_unknown
 OSParameters.h, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_varRef-
 Elements
 OSParameters.h, [1344](#)
- ENUM_MATRIX_SYMMETRY_HermitianLower
 OSParameters.h, [1344](#)
- ENUM_MATRIX_SYMMETRY_HermitianUpper
 OSParameters.h, [1344](#)
- ENUM_MATRIX_SYMMETRY_lower
 OSParameters.h, [1343](#)
- ENUM_MATRIX_SYMMETRY_none
 OSParameters.h, [1343](#)
- ENUM_MATRIX_SYMMETRY_skewLower
 OSParameters.h, [1344](#)
- ENUM_MATRIX_SYMMETRY_skewUpper
 OSParameters.h, [1344](#)
- ENUM_MATRIX_SYMMETRY_upper
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_conref
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_constant
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_general
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_jumbled
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_linear
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_mixedref
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_objref
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_quadratic
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_unknown
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_varref
 OSParameters.h, [1343](#)
- ENUM_MATRIX_TYPE_zero
 OSParameters.h, [1343](#)
- ENUM_NL_EXPR_SHAPE_constant
 OSParameters.h, [1344](#)
- ENUM_NL_EXPR_SHAPE_convex
 OSParameters.h, [1344](#)
- ENUM_NL_EXPR_SHAPE_general
 OSParameters.h, [1344](#)
- ENUM_NL_EXPR_SHAPE_linear
 OSParameters.h, [1344](#)
- ENUM_NL_EXPR_SHAPE_quadratic
 OSParameters.h, [1344](#)
- ENUM_OUTPUT_AREA_Command_line_parser
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_NUMBER_OF_AREAS
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSAgent
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSGeneral
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSInstance
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSMatrix
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSModelInterfaces
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSOption
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSResult
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSSolverInterfaces
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSUtils
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSiL_parser
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSiLwriter
 OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSoL_parser
 OSParameters.h, [1338](#)

- ENUM_OUTPUT_AREA_OSoLwriter
OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSrL_parser
OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_OSrLwriter
OSParameters.h, [1338](#)
- ENUM_OUTPUT_AREA_main
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_NUMBER_OF_LEVELS
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_always
OSParameters.h, [1337](#)
- ENUM_OUTPUT_LEVEL_debug
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_detailed_trace
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_error
OSParameters.h, [1337](#)
- ENUM_OUTPUT_LEVEL_info
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_summary
OSParameters.h, [1337](#)
- ENUM_OUTPUT_LEVEL_trace
OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL_warning
OSParameters.h, [1338](#)
- ENUM_PATHPAIR_input_dir
OSParameters.h, [1343](#)
- ENUM_PATHPAIR_input_file
OSParameters.h, [1343](#)
- ENUM_PATHPAIR_output_dir
OSParameters.h, [1343](#)
- ENUM_PATHPAIR_output_file
OSParameters.h, [1343](#)
- ENUM_PROBLEM_COMPONENT_constraints
OSParameters.h, [1342](#)
- ENUM_PROBLEM_COMPONENT_objectives
OSParameters.h, [1342](#)
- ENUM_PROBLEM_COMPONENT_variables
OSParameters.h, [1342](#)
- ENUM_SERVICE_TYPE_agent
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_analyzer
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_modeler
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_registry
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_scheduler
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_simulations
OSParameters.h, [1340](#)
- ENUM_SERVICE_TYPE_solver
OSParameters.h, [1340](#)
- ENUM_SOLUTION_STATUS_bestSoFar
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_error
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_feasible
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_globallyOptimal
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_infeasible
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_locallyOptimal
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_optimal
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_other
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_unbounded
OSParameters.h, [1342](#)
- ENUM_SOLUTION_STATUS_unsure
OSParameters.h, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_other
OSParameters.h, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_stoppedBy-
Bounds
OSParameters.h, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByLimit
OSParameters.h, [1342](#)
- ENUM_STORAGEUNIT_byte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_exabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_gigabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_kilobyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_megabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_petabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_terabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_yottabyte
OSParameters.h, [1339](#)
- ENUM_STORAGEUNIT_zettabyte
OSParameters.h, [1339](#)
- ENUM_SYSTEM_CURRENT_STATE_busy
OSParameters.h, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_busyButAccepting
OSParameters.h, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_idle
OSParameters.h, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_idleButNotAccepting
OSParameters.h, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_noResponse

OSParameters.h, [1341](#)
ENUM_TIMECATEGORY_input
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_optimization
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_other
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_output
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_postprocessing
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_preprocessing
OSParameters.h, [1340](#)
ENUM_TIMECATEGORY_total
OSParameters.h, [1340](#)
ENUM_TIMETYPE_cpuTime
OSParameters.h, [1339](#)
ENUM_TIMETYPE_elapsedTime
OSParameters.h, [1339](#)
ENUM_TIMETYPE_other
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_day
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_hour
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_millisecond
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_minute
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_month
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_second
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_tick
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_week
OSParameters.h, [1339](#)
ENUM_TIMEUNIT_year
OSParameters.h, [1339](#)
ENUM_TRANSPORT_TYPE_ftp
OSParameters.h, [1340](#)
ENUM_TRANSPORT_TYPE_http
OSParameters.h, [1340](#)
ENUM_TRANSPORT_TYPE_osp
OSParameters.h, [1340](#)
ENUM_TRANSPORT_TYPE_other
OSParameters.h, [1340](#)
ENUM_TRANSPORT_TYPE_smtp
OSParameters.h, [1340](#)
ENUM_VARTYPE_binary
OSParameters.h, [1342](#)
ENUM_VARTYPE_continuous
OSParameters.h, [1342](#)
ENUM_VARTYPE_integer

OSParameters.h, [1342](#)
ENUM_VARTYPE_semicontinuous
OSParameters.h, [1342](#)
ENUM_VARTYPE_semiinteger
OSParameters.h, [1342](#)
ENUM_VARTYPE_string
OSParameters.h, [1342](#)
ENUMERATIONEND
OSParseosil.tab.hpp, [1134](#), [1143](#), [1151](#)
OSParseosol.tab.hpp, [1209](#), [1218](#), [1226](#)
OSParseosrl.tab.hpp, [1282](#), [1291](#), [1299](#)
ENUMERATIONSTART
OSParseosil.tab.hpp, [1134](#), [1143](#), [1151](#)
OSParseosol.tab.hpp, [1209](#), [1218](#), [1226](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1299](#)
ENUMTYPEATT
OSParseosil.tab.hpp, [1139](#), [1149](#)
OSParseosol.tab.hpp, [1214](#), [1224](#)
OSParseosrl.tab.hpp, [1287](#), [1297](#)
ERFEND
OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
ERFSTART
OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
ESTART
OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
EXPEND
OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
EXPEND
OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
EXPRSTART
OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
EXPSTART
OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
EEND
OSParseosil.tab.hpp, [1126](#)
OSParseosol.tab.hpp, [1201](#)
OSParseosrl.tab.hpp, [1274](#)
ELEMENTSEND
OSParseosil.tab.hpp, [1120](#)
OSParseosol.tab.hpp, [1195](#)

- OSParseosrl.tab.hpp, [1268](#)
- ELEMENTSSTART
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- ELEMENTTEXT
 - OSParseosil.tab.hpp, [1106](#)
 - OSParseosol.tab.hpp, [1175](#)
 - OSParseosrl.tab.hpp, [1249](#)
- ELEND
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1257](#)
- ELSTART
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1257](#)
- EMPTYCATEGORYATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1252](#)
- EMPTYCONTYPEATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYDESCRIPTIONATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1252](#)
- EMPTYENUMTYPEATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYIDATT
 - OSParseosil.tab.hpp, [1107](#)
- EMPTYLBDUALVALUEATT
 - OSParseosol.tab.hpp, [1180](#)
- EMPTYLBVALUEATT
 - OSParseosol.tab.hpp, [1179](#)
- EMPTYNAMEATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1252](#)
- EMPTYOBJTYPEATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYROWMAJORATT
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- EMPTYSHAPEATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- EMPTYSOLVERATT
 - OSParseosol.tab.hpp, [1180](#)
- EMPTYSYMMETRYATT
 - OSParseosil.tab.hpp, [1118](#)
- OSParseosol.tab.hpp, [1193](#)
- OSParseosrl.tab.hpp, [1266](#)
- EMPTYTYPEATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1252](#)
- EMPTYUBDUALVALUEATT
 - OSParseosol.tab.hpp, [1180](#)
- EMPTYUBVALUEATT
 - OSParseosol.tab.hpp, [1179](#)
- EMPTYUNITATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYVALUEATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYVARTYPEATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- EMPTYWEIGHTATT
 - OSParseosol.tab.hpp, [1180](#)
- ENDOFELEMENT
 - OSParseosil.tab.hpp, [1107](#)
 - OSParseosol.tab.hpp, [1176](#)
 - OSParseosrl.tab.hpp, [1250](#)
- ENDTIMEEND
 - OSParseosrl.tab.hpp, [1258](#)
- ENDTIMESTART
 - OSParseosrl.tab.hpp, [1257](#)
- ENUM_BASIS_STATUS
 - OSParameters.h, [1341](#)
- ENUM_CONE_TYPE
 - OSParameters.h, [1344](#)
- ENUM_CPUSPEEDUNIT
 - OSParameters.h, [1338](#)
- ENUM_JOB_STATUS
 - OSParameters.h, [1341](#)
- ENUM_LOCATIONTYPE
 - OSParameters.h, [1340](#)
- ENUM_MATRIX_TYPE
 - OSParameters.h, [1343](#)
- ENUM_OUTPUT_AREA
 - OSParameters.h, [1338](#)
- ENUM_OUTPUT_LEVEL
 - OSParameters.h, [1337](#)
- ENUM_PATHPAIR
 - OSParameters.h, [1342](#)
- ENUM_SERVICE_TYPE
 - OSParameters.h, [1340](#)
- ENUM_STORAGEUNIT
 - OSParameters.h, [1339](#)
- ENUM_TIMECATEGORY
 - OSParameters.h, [1339](#)
- ENUM_TIMETYPE

- OSParameters.h, [1339](#)
- ENUM_TIMEUNIT
 - OSParameters.h, [1339](#)
- ENUM_VARTYPE
 - OSParameters.h, [1342](#)
- ENUMERATIONEND
 - OSParseosil.tab.hpp, [1116](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1257](#)
- ENUMERATIONSTART
 - OSParseosil.tab.hpp, [1116](#)
 - OSParseosol.tab.hpp, [1189](#)
 - OSParseosrl.tab.hpp, [1257](#)
- ENUMTYPEATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1253](#)
- ERFEND
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1273](#)
- ERFSTART
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1273](#)
- ESTART
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1274](#)
- EXPEND
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- EXPEND
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- EXPRSTART
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- EXPSTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- el
 - ConReferenceMatrixValues, [74](#)
 - ConstantMatrixValues, [81](#)
 - DoubleVector, [118](#)
 - GeneralMatrixValues, [149](#)
 - IntVector, [229](#)
 - LinearMatrixValues, [270](#)
 - ObjReferenceMatrixValues, [371](#)
 - VarReferenceMatrixValues, [1029](#)
- elCounter
 - OSILParserData, [427](#)
- encodeb64
 - Base64, [31](#)
- endTime
 - JobResult, [247](#)
- enumType
 - OtherConstraintOption, [850](#)
 - OtherConstraintResult, [854](#)
 - OtherObjectiveOption, [858](#)
 - OtherObjectiveResult, [862](#)
 - OtherVariableOption, [892](#)
 - OtherVariableResult, [896](#)
- enumTypeAttribute
 - OSnLParserData, [647](#)
 - OSoLParserData, [668](#)
 - OSrLParserData, [823](#)
- enumTypeAttributePresent
 - OSnLParserData, [647](#)
 - OSoLParserData, [668](#)
 - OSrLParserData, [822](#)
- enumeration
 - OtherConstraintOption, [850](#)
 - OtherConstraintResult, [853](#)
 - OtherObjectiveOption, [858](#)
 - OtherObjectiveResult, [861](#)
 - OtherVariableOption, [892](#)
 - OtherVariableResult, [896](#)
- ErrorClass, [127](#)
 - ErrorClass, [128](#)
 - ErrorClass, [128](#)
 - errmsg, [128](#)
- errorText
 - OSgLParserData, [410](#)
 - OSILParserData, [431](#)
 - OSnLParserData, [651](#)
 - OSoLParserData, [671](#)
 - OSrLParserData, [826](#)
- errmsg
 - ErrorClass, [128](#)
- eval_f
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- eval_g
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- eval_grad_f
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- eval_h
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- eval_jac_g
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)

- evalFC
 - KnitroProblem, [250](#)
- evalGA
 - KnitroProblem, [250](#)
- evalH
 - KnitroProblem, [250](#)
- evalHV
 - KnitroProblem, [250](#)
- ExpandedMatrixBlocks, [128](#)
 - ~ExpandedMatrixBlocks, [130](#)
 - bDeleteArrays, [130](#)
 - blockColumns, [131](#)
 - blockNumber, [131](#)
 - blockRows, [131](#)
 - blocks, [131](#)
 - colOffsets, [131](#)
 - display, [130](#)
 - ExpandedMatrixBlocks, [130](#)
 - ExpandedMatrixBlocks, [130](#)
 - isColumnMajor, [130](#)
 - rowOffsets, [131](#)
- expr
 - MatrixExpressions, [298](#)
- ExprNode, [132](#)
 - ~ExprNode, [134](#)
 - cloneExprNode, [136](#)
 - ExprNode, [134](#)
 - ExprNode, [134](#)
 - getNonlinearExpressionInXML, [135](#)
 - getPostfixFromExpressionTree, [135](#)
 - getPrefixFromExpressionTree, [135](#)
 - getTokenName, [134](#)
 - getTokenNumber, [134](#)
 - inodeInt, [136](#)
 - inodeType, [136](#)
 - inumberOfChildren, [137](#)
 - inumberOfMatrixChildren, [137](#)
 - isEqual, [136](#)
 - m_mChildren, [137](#)
 - m_mMatrixChildren, [137](#)
 - postOrderOSnLNodeTraversal, [136](#)
 - preOrderOSnLNodeTraversal, [135](#)
- extendIntVector
 - IntVector, [229](#)
- extractBlock
 - MatrixType, [321](#)
- FACTORSEND
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- FACTORSSTART
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
- OSParseosrl.tab.hpp, [1279](#)
- FILECREATOREMPTY
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILECREATOREND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILECREATORSTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILECREATORSTARTANDEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILEDESCRIPTIONEMPTY
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILEDESCRIPTIONEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
- FILEDESCRIPTIONSTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
- FILEDESCRIPTIONSTARTANDEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILELICENCEEMPTY
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILELICENCEEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILELICENCESTART
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILELICENCESTARTANDEND
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1292](#), [1301](#)
- FILENAMEEMPTY
 - OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
- FILENAMEEND

OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILENAMESTART
OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILENAMESTARTANDEND
OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILESOURCEEMPTY
OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILESOURCEEND
OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILESOURCESTART
OSParseosil.tab.hpp, [1133](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILESOURCESTARTANDEND
OSParseosil.tab.hpp, [1134](#), [1144](#), [1153](#)
OSParseosol.tab.hpp, [1208](#), [1219](#), [1228](#)
OSParseosrl.tab.hpp, [1281](#), [1291](#), [1301](#)
FILESTODELETEEND
OSParseosil.tab.hpp, [1142](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)
FILESTODELETESTART
OSParseosil.tab.hpp, [1142](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)
FILESTOMAKEEND
OSParseosil.tab.hpp, [1142](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)
FILESTOMAKESTART
OSParseosil.tab.hpp, [1141](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)
FIRSTAXISDIRECTIONATT
OSParseosil.tab.hpp, [1132](#)
OSParseosol.tab.hpp, [1207](#)
OSParseosrl.tab.hpp, [1280](#)
FROMATT
OSParseosil.tab.hpp, [1139](#)
OSParseosol.tab.hpp, [1214](#)
OSParseosrl.tab.hpp, [1286](#)
FACTORSEND
OSParseosil.tab.hpp, [1110](#)
FACTORSTART

OSParseosil.tab.hpp, [1110](#)
FILECREATOREMPTY
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILECREATOREND
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILECREATORSTART
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILEDESCRIPTIONEMPTY
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILEDESCRIPTIONEND
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILEDESCRIPTIONSTART
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILELICENCEEMPTY
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILELICENCEEND
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILELICENCESTART
OSParseosil.tab.hpp, [1116](#)
OSParseosol.tab.hpp, [1192](#)
OSParseosrl.tab.hpp, [1265](#)
FILENAMEEMPTY
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1191](#)
OSParseosrl.tab.hpp, [1264](#)
FILENAMEEND
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1191](#)
OSParseosrl.tab.hpp, [1264](#)
FILENAMESTART
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1191](#)
OSParseosrl.tab.hpp, [1264](#)
FILENAMESTARTANDEND
OSParseosil.tab.hpp, [1115](#)
OSParseosol.tab.hpp, [1191](#)
OSParseosrl.tab.hpp, [1264](#)
FILESOURCEEMPTY

- OSParseosil.tab.hpp, [1115](#)
 - OSParseosol.tab.hpp, [1191](#)
 - OSParseosrl.tab.hpp, [1264](#)
- FILESOURCEEND
 - OSParseosil.tab.hpp, [1115](#)
 - OSParseosol.tab.hpp, [1191](#)
 - OSParseosrl.tab.hpp, [1264](#)
- FILESOURCESTART
 - OSParseosil.tab.hpp, [1115](#)
 - OSParseosol.tab.hpp, [1191](#)
 - OSParseosrl.tab.hpp, [1264](#)
- FILESTODELETEEND
 - OSParseosol.tab.hpp, [1185](#)
- FILESTODELETESTART
 - OSParseosol.tab.hpp, [1185](#)
- FILESTOMAKEEND
 - OSParseosol.tab.hpp, [1185](#)
- FILESTOMAKESTART
 - OSParseosol.tab.hpp, [1185](#)
- FROMATT
 - OSParseosol.tab.hpp, [1176](#)
- factors
 - ProductCone, [923](#)
- fileCreator
 - GeneralFileHeader, [142](#)
 - OSgLParseData, [410](#)
- fileCreatorPresent
 - OSgLParseData, [410](#)
- fileName
 - OSgLParseData, [410](#)
- fileNamePresent
 - OSgLParseData, [410](#)
- filePrintLevel
 - OSCommandLine, [398](#)
 - osOptionsStruc, [719](#)
- fileUpload
 - OSSolverAgent, [840](#)
- FileUtil, [137](#)
 - ~FileUtil, [138](#)
 - FileUtil, [138](#)
 - FileUtil, [138](#)
 - getFileAsChar, [139](#)
 - getFileAsString, [138](#)
 - writeFileFromChar, [139](#)
 - writeFileFromString, [139](#)
- filesToDelete
 - JobOption, [244](#)
- filesToDeletePresent
 - OSoLParseData, [662](#)
- filesToMake
 - JobOption, [243](#)
- filesToMakePresent
 - OSoLParseData, [661](#)
- finalize_solution
 - BonminProblem, [45](#)
 - IpoptProblem, [233](#)
- FindChannel
 - OSOutput, [724](#)
- first_column
 - YYLTYPE, [1038](#)
- first_line
 - YYLTYPE, [1038](#)
- firstAxisDirection
 - OSILParserData, [427](#)
 - RotatedQuadraticCone, [935](#)
- firstAxisDirectionPresent
 - OSILParserData, [427](#)
- FlushAllBuffers
 - OSOutput, [723](#)
- flushBuffer
 - OSOutputChannel, [727](#)
- format_os_dtoa
 - MathUtil, [272](#)
- forwardAD
 - OSInstance, [479](#)
- from
 - PathPair, [905](#)
- fromPaths
 - OSoLParseData, [669](#)
- GENERALELEMENTSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- GENERALELEMENTSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- GENERALEND
 - OSParseosil.tab.hpp, [1140](#), [1150](#)
 - OSParseosol.tab.hpp, [1215](#), [1224](#)
 - OSParseosrl.tab.hpp, [1288](#), [1297](#)
- GENERALSTART
 - OSParseosil.tab.hpp, [1140](#), [1150](#)
 - OSParseosol.tab.hpp, [1215](#), [1224](#)
 - OSParseosrl.tab.hpp, [1288](#), [1297](#)
- GENERALSTATUSEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- GENERALSTATUSSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- GENERALSUBSTATUSEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)

- GENERALSUBSTATUSSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- GREATERTHAN
 - OSParseosil.tab.hpp, [1131](#), [1139](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- GROUPWEIGHTATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- GENERALELEMENTSEND
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- GENERALELEMENTSSTART
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- GENERALEND
 - OSParseosol.tab.hpp, [1180](#)
 - OSParseosrl.tab.hpp, [1254](#)
- GENERALSTART
 - OSParseosol.tab.hpp, [1180](#)
 - OSParseosrl.tab.hpp, [1254](#)
- GENERALSTATUSEND
 - OSParseosrl.tab.hpp, [1258](#)
- GENERALSTATUSSTART
 - OSParseosrl.tab.hpp, [1258](#)
- GENERSUBSTATUSSEND
 - OSParseosrl.tab.hpp, [1258](#)
- GREATERTHAN
 - OSParseosil.tab.hpp, [1107](#)
 - OSParseosol.tab.hpp, [1176](#)
 - OSParseosrl.tab.hpp, [1250](#)
- GROUPWEIGHTATT
 - OSParseosol.tab.hpp, [1177](#)
- gamsControlFile
 - OSCommandLine, [398](#)
 - osOptionsStruc, [719](#)
- general
 - OSOption, [713](#)
 - OSResult, [810](#)
- GeneralFileHeader, [140](#)
 - ~GeneralFileHeader, [141](#)
 - deepCopyFrom, [142](#)
 - description, [142](#)
 - fileCreator, [142](#)
 - GeneralFileHeader, [141](#)
 - GeneralFileHeader, [141](#)
 - getHeaderItem, [142](#)
 - IsEqual, [141](#)
 - licence, [143](#)
 - name, [142](#)
 - setHeader, [142](#)
 - setRandom, [141](#)
 - source, [142](#)
- generalInstanceNamePresent
 - OSrLParserData, [824](#)
- generalJobIDPresent
 - OSrLParserData, [824](#)
- GeneralMatrixElements, [143](#)
 - ~GeneralMatrixElements, [145](#)
 - alignsOnBlockBoundary, [145](#)
 - cloneMatrixNode, [146](#)
 - deepCopyFrom, [146](#)
 - GeneralMatrixElements, [145](#)
 - GeneralMatrixElements, [145](#)
 - getMatrixNodeInXML, [145](#)
 - getMatrixType, [145](#)
 - getNodeName, [145](#)
 - getNodeType, [145](#)
 - IsEqual, [146](#)
 - setRandom, [146](#)
 - values, [146](#)
- GeneralMatrixValues, [147](#)
 - ~GeneralMatrixValues, [148](#)
 - deepCopyFrom, [148](#)
 - el, [149](#)
 - GeneralMatrixValues, [148](#)
 - GeneralMatrixValues, [148](#)
 - IsEqual, [148](#)
 - setRandom, [148](#)
- generalMessagePresent
 - OSrLParserData, [823](#)
- GeneralOption, [149](#)
 - ~GeneralOption, [152](#)
 - contact, [153](#)
 - deepCopyFrom, [152](#)
 - GeneralOption, [152](#)
 - GeneralOption, [152](#)
 - instanceLocation, [153](#)
 - instanceName, [152](#)
 - IsEqual, [152](#)
 - jobID, [153](#)
 - license, [153](#)
 - otherOptions, [153](#)
 - password, [153](#)
 - serviceName, [152](#)
 - serviceURI, [152](#)
 - setRandom, [152](#)
 - solverToInvoke, [153](#)
 - userName, [153](#)
- GeneralResult, [153](#)
 - ~GeneralResult, [155](#)
 - GeneralResult, [155](#)
 - generalStatus, [156](#)

- GeneralResult, [155](#)
- instanceName, [156](#)
- IsEqual, [156](#)
- jobID, [156](#)
- message, [156](#)
- otherResults, [157](#)
- serviceName, [156](#)
- serviceURI, [156](#)
- setRandom, [156](#)
- solverInvoked, [156](#)
- timeStamp, [157](#)
- generalServiceNamePresent
 - OSrLParserData, [823](#)
- generalServiceURIPresent
 - OSrLParserData, [823](#)
- generalSolverInvokedPresent
 - OSrLParserData, [824](#)
- GeneralSparseMatrix, [157](#)
 - ~GeneralSparseMatrix, [158](#)
 - bDeleteArrays, [159](#)
 - display, [158](#)
 - GeneralSparseMatrix, [158](#)
 - GeneralSparseMatrix, [158](#)
 - indexes, [159](#)
 - isColumnMajor, [159](#)
 - startSize, [159](#)
 - starts, [159](#)
 - vType, [159](#)
 - valueSize, [159](#)
 - values, [159](#)
- GeneralStatus, [160](#)
 - ~GeneralStatus, [161](#)
 - description, [162](#)
 - GeneralStatus, [161](#)
 - GeneralStatus, [161](#)
 - IsEqual, [162](#)
 - numberOfSubstatuses, [162](#)
 - setRandom, [162](#)
 - substatus, [162](#)
 - type, [162](#)
- generalStatus
 - GeneralResult, [156](#)
- generalStatusPresent
 - OSrLParserData, [823](#)
- GeneralSubstatus, [162](#)
 - ~GeneralSubstatus, [164](#)
 - description, [164](#)
 - GeneralSubstatus, [164](#)
 - GeneralSubstatus, [164](#)
 - IsEqual, [164](#)
 - name, [164](#)
 - setRandom, [164](#)
- generalTimeStampPresent
 - OSrLParserData, [824](#)
- generateLindoModel
 - LindoSolver, [258](#)
- get_bounds_info
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- get_constraints_linearity
 - BonminProblem, [44](#)
- get_nlp_info
 - BonminProblem, [44](#)
 - IpoptProblem, [232](#)
- get_scaling_parameters
 - BonminProblem, [45](#)
 - IpoptProblem, [233](#)
- get_starting_point
 - BonminProblem, [45](#)
 - IpoptProblem, [232](#)
- get_variables_linearity
 - BonminProblem, [44](#)
- get_variables_types
 - BonminProblem, [44](#)
- getADSparsityHessian
 - OSInstance, [480](#)
- getASL
 - OSnl2OS, [502](#)
- getActualStartTime
 - OSResult, [749](#)
- getAllMatrixExpressionTrees
 - OSInstance, [459](#)
- getAllMatrixExpressionTreesMod
 - OSInstance, [459](#)
- getAllNonlinearExpressionTrees
 - OSInstance, [453](#)
- getAllNonlinearExpressionTreesMod
 - OSInstance, [453](#)
- getAllNonlinearVariablesIndexMap
 - OSInstance, [478](#)
- getAllOtherConstraintOptions
 - OSOption, [703](#)
- getAllOtherObjectiveOptions
 - OSOption, [701](#)
- getAllOtherOptions
 - OSOption, [693](#)
- getAllOtherVariableOptions
 - OSOption, [698](#)
- getAllSolverOptions
 - OSOption, [704](#)
- getAnOtherVariableResultNumberOfVar
 - OSResult, [753](#)
- getAvailableCPUNumberDescription
 - OSResult, [748](#)
- getAvailableCPUNumberValue
 - OSResult, [748](#)
- getAvailableCPUSpeedDescription
 - OSResult, [748](#)

- getAvailableCPUSpeedUnit
 - OSResult, 748
- getAvailableCPUSpeedValue
 - OSResult, 748
- getAvailableDiskSpaceDescription
 - OSResult, 748
- getAvailableDiskSpaceUnit
 - OSResult, 748
- getAvailableDiskSpaceValue
 - OSResult, 748
- getAvailableMemoryDescription
 - OSResult, 748
- getAvailableMemoryUnit
 - OSResult, 748
- getAvailableMemoryValue
 - OSResult, 748
- getBasisDense
 - BasisStatus, 40
- getBasisInformationDense
 - OSResult, 753
- getBasisStatusEl
 - OSResult, 753
- getBasisStatusNumberOfEl
 - OSResult, 752
- getCoinSolverType
 - CoinSolver, 56
- getColumnPartition
 - OSMatrix, 492
- getColumnPartitionSize
 - OSMatrix, 492
- getConeInXML
 - CompletelyPositiveMatricesCone, 59
 - Cone, 63
 - CopositiveMatricesCone, 95
 - IntersectionCone, 224
 - NonnegativeCone, 346
 - NonpositiveCone, 349
 - OrthantCone, 389
 - PolyhedralCone, 915
 - ProductCone, 921
 - QuadraticCone, 927
 - RotatedQuadraticCone, 934
 - SemidefiniteCone, 946
- getConeName
 - CompletelyPositiveMatricesCone, 59
 - Cone, 63
 - CopositiveMatricesCone, 95
 - DualCone, 120
 - IntersectionCone, 224
 - NonnegativeCone, 346
 - NonpositiveCone, 349
 - OrthantCone, 389
 - PolarCone, 911
 - PolyhedralCone, 915
 - ProductCone, 921
 - QuadraticCone, 927
 - RotatedQuadraticCone, 934
 - SemidefiniteCone, 946
- getConstraintConstants
 - OSInstance, 449
- getConstraintLowerBounds
 - OSInstance, 449
- getConstraintNames
 - OSInstance, 448
- getConstraintNumber
 - OSInstance, 448
 - OSResult, 750
- getConstraintTypes
 - OSInstance, 449
- getConstraintUpperBounds
 - OSInstance, 449
- getContact
 - OSOption, 687
- getContactTransportType
 - OSOption, 688
- getCurrentJobCount
 - OSResult, 748
- getCurrentState
 - OSResult, 748
- getDenseObjectiveCoefficients
 - OSInstance, 448
- getDescription
 - OtherOptionEnumeration, 872
- getDirectoriesToDelete
 - OSOption, 694
- getDirectoriesToMake
 - OSOption, 693
- getDualValue
 - OSResult, 757
- getDualValueIdx
 - OSResult, 757
- getDualValueName
 - OSResult, 757
- getEl
 - BasisStatus, 39
 - IntVector, 229
- getFileAsChar
 - FileUtil, 139
- getFileAsString
 - FileUtil, 138
- getFileCreator
 - OSOption, 687
- getFileDescription
 - OSOption, 687
- getFileLicence
 - OSOption, 687
- getFileName
 - OSOption, 686

getFileSource
 OSOption, 686
getFilesToDelete
 OSOption, 694
getFilesToMake
 OSOption, 694
getFirstOrderResults
 OSInstance, 481
getGeneralMessage
 OSResult, 746
getGeneralStatus
 OSResult, 745
getGeneralStatusDescription
 OSResult, 745
getGeneralStatusType
 OSResult, 745
getGeneralSubstatusDescription
 OSResult, 746
getGeneralSubstatusName
 OSResult, 746
getHeaderItem
 GeneralFileHeader, 142
getInitBasisStatusDense
 OSOption, 696
getInitBasisStatusSparse
 OSOption, 696
getInitConValuesDense
 OSOption, 701
getInitConValuesSparse
 OSOption, 701
getInitDualVarLowerBoundsDense
 OSOption, 701
getInitDualVarUpperBoundsDense
 OSOption, 702
getInitDualVarValuesSparse
 OSOption, 701
getInitObjBoundsSparse
 OSOption, 699
getInitObjLowerBoundsDense
 OSOption, 699
getInitObjUpperBoundsDense
 OSOption, 699, 700
getInitObjValuesDense
 OSOption, 698, 699
getInitObjValuesSparse
 OSOption, 698
getInitVarValuesDense
 OSOption, 695
getInitVarValuesSparse
 OSOption, 695
getInitVarValuesStringDense
 OSOption, 695, 696
getInitVarValuesStringSparse
 OSOption, 695
getInitialBasisElements
 OSOption, 697
getInitialX
 KnitroProblem, 250
getInputDirectoriesToMove
 OSOption, 694
getInputFilesToMove
 OSOption, 694
getInstanceCreator
 OSInstance, 444
getInstanceDescription
 OSInstance, 443
getInstanceLicence
 OSInstance, 444
getInstanceLocation
 OSOption, 687
getInstanceLocationType
 OSOption, 687
getInstanceName
 OSInstance, 443
 OSOption, 687
 OSResult, 747
getInstanceSource
 OSInstance, 443
getIntVector
 BasisStatus, 40
getIntegerVariableBranchingWeightsDense
 OSOption, 697
getIntegerVariableBranchingWeightsSparse
 OSOption, 697
getIterateResults
 OSInstance, 480
getJacobianSparsityPattern
 OSInstance, 479
getJobDependencies
 OSOption, 693
getJobEndTime
 OSResult, 749
getJobID
 OShL, 417
 OSOption, 687
 OSResult, 747
 OSSolverAgent, 838
getJobStatus
 OSResult, 748
getJobSubmitTime
 OSResult, 749
getLagrangianExpTree
 OSInstance, 478
getLagrangianHessianSparsityPattern
 OSInstance, 478
getLicense
 OSOption, 687
getLinearConstraintCoefficientMajor

- OSInstance, [450](#)
- getLinearConstraintCoefficientNumber
 - OSInstance, [450](#)
- getLinearConstraintCoefficientsInColumnMajor
 - OSInstance, [450](#)
- getLinearConstraintCoefficientsInRowMajor
 - OSInstance, [450](#)
- getM
 - KnitroProblem, [250](#)
- getMatrix
 - OSInstance, [456](#)
- getMatrixBlockInColumnMajorForm
 - MatrixType, [321](#)
 - OSInstance, [457](#)
- getMatrixCoefficientsInColumnMajor
 - OSInstance, [456](#)
- getMatrixCoefficientsInRowMajor
 - OSInstance, [457](#)
- getMatrixExpressionTree
 - OSInstance, [458](#)
- getMatrixExpressionTreeInInfix
 - OSInstance, [459](#)
- getMatrixExpressionTreeInPostfix
 - OSInstance, [458](#)
- getMatrixExpressionTreeInPrefix
 - OSInstance, [458](#)
- getMatrixExpressionTreeIndexes
 - OSInstance, [459](#)
- getMatrixExpressionTreeModInPostfix
 - OSInstance, [458](#)
- getMatrixExpressions
 - OSInstance, [458](#)
- getMatrixInColumnMajorForm
 - MatrixType, [321](#)
- getMatrixInRowMajorForm
 - MatrixType, [321](#)
- getMatrixName
 - OSInstance, [455](#)
- getMatrixNodeInXML
 - BaseMatrix, [35](#)
 - ConReferenceMatrixElements, [71](#)
 - ConstantMatrixElements, [77](#)
 - GeneralMatrixElements, [145](#)
 - LinearMatrixElements, [266](#)
 - MatrixBlock, [277](#)
 - MatrixBlocks, [281](#)
 - MatrixNode, [304](#)
 - MatrixTransformation, [316](#)
 - ObjReferenceMatrixElements, [367](#)
 - OSMatrix, [493](#)
 - RowReferenceMatrixElements, [938](#)
 - VarReferenceMatrixElements, [1026](#)
- getMatrixNumber
 - OSInstance, [454](#)
- getMatrixSymmetry
 - OSInstance, [455](#)
- getMatrixType
 - BaseMatrix, [34](#)
 - ConReferenceMatrixElements, [71](#)
 - ConstantMatrixElements, [77](#)
 - GeneralMatrixElements, [145](#)
 - LinearMatrixElements, [265](#)
 - MatrixBlock, [277](#)
 - MatrixBlocks, [281](#)
 - MatrixNode, [304](#)
 - MatrixTransformation, [316](#)
 - ObjReferenceMatrixElements, [367](#)
 - OSInstance, [454](#)
 - OSMatrix, [491](#)
 - RowReferenceMatrixElements, [938](#)
 - VarReferenceMatrixElements, [1025](#)
- getMaxTime
 - OSOption, [689](#)
- getMaxTimeUnit
 - OSOption, [688](#)
- getMinCPUNumber
 - OSOption, [689](#)
- getMinCPUNumberDescription
 - OSOption, [688](#)
- getMinCPUSpeed
 - OSOption, [689](#)
- getMinCPUSpeedDescription
 - OSOption, [688](#)
- getMinCPUSpeedUnit
 - OSOption, [688](#)
- getMinDiskSpace
 - OSOption, [688](#)
- getMinDiskSpaceDescription
 - OSOption, [688](#)
- getMinDiskSpaceUnit
 - OSOption, [688](#)
- getMinMemoryDescription
 - OSOption, [688](#)
- getMinMemorySize
 - OSOption, [688](#)
- getMinMemoryUnit
 - OSOption, [688](#)
- getMult
 - OSMathUtil.h, [1325](#)
- getMultIncr
 - OSMathUtil.h, [1325](#)
- getN
 - KnitroProblem, [250](#)
- getNodeName
 - BaseMatrix, [34](#)
 - ConReferenceMatrixElements, [71](#)
 - ConstantMatrixElements, [77](#)
 - GeneralMatrixElements, [145](#)

- LinearMatrixElements, [265](#)
- MatrixBlock, [277](#)
- MatrixBlocks, [281](#)
- MatrixNode, [304](#)
- MatrixTransformation, [316](#)
- ObjReferenceMatrixElements, [367](#)
- OSMatrix, [491](#)
- RowReferenceMatrixElements, [938](#)
- VarReferenceMatrixElements, [1026](#)
- getNodeType
 - BaseMatrix, [34](#)
 - ConReferenceMatrixElements, [71](#)
 - ConstantMatrixElements, [77](#)
 - GeneralMatrixElements, [145](#)
 - LinearMatrixElements, [265](#)
 - MatrixBlock, [277](#)
 - MatrixBlocks, [281](#)
 - MatrixNode, [304](#)
 - MatrixTransformation, [316](#)
 - ObjReferenceMatrixElements, [367](#)
 - OSMatrix, [491](#)
 - RowReferenceMatrixElements, [938](#)
 - VarReferenceMatrixElements, [1025](#)
- getNonlinearExpressionInXML
 - ExprNode, [135](#)
 - OSnLMNodeMatrixCon, [515](#)
 - OSnLMNodeMatrixLowerTriangle, [523](#)
 - OSnLMNodeMatrixObj, [530](#)
 - OSnLMNodeMatrixReference, [537](#)
 - OSnLMNodeMatrixUpperTriangle, [549](#)
 - OSnLMNodeMatrixVar, [552](#)
 - OSnLNodeE, [574](#)
 - OSnLNodeNumber, [610](#)
 - OSnLNodePI, [614](#)
 - OSnLNodeVariable, [641](#)
- getNonlinearExpressionTree
 - OSInstance, [452](#)
- getNonlinearExpressionTreeInInfix
 - OSInstance, [452](#)
- getNonlinearExpressionTreeInPostfix
 - OSInstance, [452](#)
- getNonlinearExpressionTreeInPrefix
 - OSInstance, [452](#)
- getNonlinearExpressionTreeIndexes
 - OSInstance, [453](#)
- getNonlinearExpressionTreeMod
 - OSInstance, [452](#)
- getNonlinearExpressionTreeModInPostfix
 - OSInstance, [452](#)
- getNonlinearExpressionTreeModInPrefix
 - OSInstance, [453](#)
- getNonlinearExpressionTreeModIndexes
 - OSInstance, [454](#)
- getNonlinearExpressions
 - OSInstance, [452](#)
- getNumberOfBinaryVariables
 - OSInstance, [445](#)
- getNumberOfBlocksConstructors
 - MatrixType, [321](#)
 - OSInstance, [456](#)
- getNumberOfColumnsForMatrix
 - OSInstance, [455](#)
- getNumberOfConstraints
 - OSOption, [690](#)
- getNumberOfDirectoriesToDelete
 - OSOption, [690](#)
- getNumberOfDirectoriesToMake
 - OSOption, [689](#)
- getNumberOfDualValues
 - OSResult, [757](#)
- getNumberOfEI
 - BasisStatus, [39](#)
 - IntVector, [229](#)
- getNumberOfElementConstructors
 - MatrixType, [321](#)
 - OSInstance, [456](#)
- getNumberOfFilesToDelete
 - OSOption, [690](#)
- getNumberOfFilesToMake
 - OSOption, [689](#)
- getNumberOfGeneralSubstatuses
 - OSResult, [746](#)
- getNumberOfInitConValues
 - OSOption, [692](#)
- getNumberOfInitDualVarValues
 - OSOption, [692](#)
- getNumberOfInitObjBounds
 - OSOption, [691](#)
- getNumberOfInitObjValues
 - OSOption, [691](#)
- getNumberOfInitVarValues
 - OSOption, [690](#)
- getNumberOfInitVarValuesString
 - OSOption, [690](#)
- getNumberOfInitialBasisElements
 - OSOption, [696](#)
- getNumberOfInputDirectoriesToMove
 - OSOption, [689](#)
- getNumberOfInputFilesToMove
 - OSOption, [690](#)
- getNumberOfIntegerVariableBranchingWeights
 - OSOption, [691](#)
- getNumberOfIntegerVariables
 - OSInstance, [445](#)
- getNumberOfJobDependencies
 - OSOption, [689](#)
- getNumberOfMatrixConstraints
 - OSInstance, [458](#)

- getNumberOfMatrixExpressionTreeIndexes
 - OSInstance, [459](#)
- getNumberOfMatrixExpressions
 - OSInstance, [458](#)
- getNumberOfMatrixObjectives
 - OSInstance, [457](#)
- getNumberOfMatrixVariables
 - OSInstance, [457](#)
- getNumberOfNonlinearConstraints
 - OSInstance, [453](#)
- getNumberOfNonlinearExpressionTreeIndexes
 - OSInstance, [453](#)
- getNumberOfNonlinearExpressionTreeModIndexes
 - OSInstance, [454](#)
- getNumberOfNonlinearExpressions
 - OSInstance, [451](#)
- getNumberOfNonlinearObjectives
 - OSInstance, [453](#)
- getNumberOfObjValues
 - OSResult, [755](#)
- getNumberOfObjectives
 - OSOption, [690](#)
- getNumberOfOtherConstraintOptions
 - OSOption, [692](#)
- getNumberOfOtherConstraintResults
 - OSResult, [758](#)
- getNumberOfOtherGeneralOptions
 - OSOption, [689](#)
- getNumberOfOtherGeneralResults
 - OSResult, [747](#)
- getNumberOfOtherJobOptions
 - OSOption, [689](#)
- getNumberOfOtherJobResults
 - OSResult, [750](#)
- getNumberOfOtherObjectiveOptions
 - OSOption, [691](#)
- getNumberOfOtherObjectiveResults
 - OSResult, [756](#)
- getNumberOfOtherServiceOptions
 - OSOption, [689](#)
- getNumberOfOtherServiceResults
 - OSResult, [748](#)
- getNumberOfOtherSolutionResults
 - OSResult, [760](#)
- getNumberOfOtherSystemOptions
 - OSOption, [689](#)
- getNumberOfOtherSystemResults
 - OSResult, [748](#)
- getNumberOfOtherVariableOptions
 - OSOption, [691](#)
- getNumberOfOtherVariableResults
 - OSResult, [753](#)
- getNumberOfOutputDirectoriesToMove
 - OSOption, [690](#)
- getNumberOfOutputFilesToMove
 - OSOption, [690](#)
- getNumberOfPrimalVariableValues
 - OSResult, [752](#)
- getNumberOfProcessesToKill
 - OSOption, [690](#)
- getNumberOfQuadraticRowIndexes
 - OSInstance, [451](#)
- getNumberOfQuadraticTerms
 - OSInstance, [451](#)
- getNumberOfRequiredDirectories
 - OSOption, [689](#)
- getNumberOfRequiredFiles
 - OSOption, [689](#)
- getNumberOfRowsForMatrix
 - OSInstance, [455](#)
- getNumberOfSOS
 - OSOption, [691](#)
- getNumberOfSOSVarBranchingWeights
 - OSOption, [691](#)
- getNumberOfSemiContinuousVariables
 - OSInstance, [445](#)
- getNumberOfSemiIntegerVariables
 - OSInstance, [445](#)
- getNumberOfSolutionSubstatuses
 - OSResult, [751](#)
- getNumberOfSolverOptions
 - OSOption, [692](#)
- getNumberOfSolverOutputs
 - OSResult, [760](#)
- getNumberOfStringVariables
 - OSInstance, [446](#)
- getNumberOfTimes
 - OSResult, [749](#)
- getNumberOfTransformationConstructors
 - MatrixType, [321](#)
 - OSInstance, [456](#)
- getNumberOfVarValues
 - OSResult, [752](#)
- getNumberOfVarValuesString
 - OSResult, [752](#)
- getNumberOfVariables
 - OSOption, [690](#)
- getOSInstance
 - OSgams2osil, [405](#)
- getOSxL
 - WSUtil, [1037](#)
- getObjValue
 - OSResult, [755](#)
- getObjValueIdx
 - OSResult, [755](#)
- getObjValueName
 - OSResult, [755](#)
- getObjectiveCoefficientNumbers

OSInstance, [447](#)
getObjectiveCoefficients
 OSInstance, [448](#)
getObjectiveConstants
 OSInstance, [447](#)
getObjectiveInitialBasisStatusDense
 OSOption, [700](#)
getObjectiveMaxOrMins
 OSInstance, [447](#)
getObjectiveNames
 OSInstance, [446](#)
getObjectiveNumber
 OSInstance, [446](#)
 OSResult, [750](#)
getObjectiveWeights
 OSInstance, [447](#)
getOptimalDualVariableValues
 OSResult, [757](#)
getOptimalObjValue
 OSResult, [755](#)
getOptimalPrimalVariableValues
 OSResult, [752](#)
getOptionDbl
 OSOption, [689](#)
getOptionInt
 OSOption, [692](#)
getOptionStr
 OSOption, [688](#)
getOtherConstraintOption
 OSOption, [703](#)
getOtherConstraintOptions
 OSOption, [702](#)
getOtherConstraintResultArrayDense
 OSResult, [759](#)
getOtherConstraintResultArrayType
 OSResult, [758](#)
getOtherConstraintResultCon
 OSResult, [758](#)
getOtherConstraintResultConIdx
 OSResult, [758](#)
getOtherConstraintResultDescription
 OSResult, [758](#)
getOtherConstraintResultEnumerationDescription
 OSResult, [759](#)
getOtherConstraintResultEnumerationEI
 OSResult, [759](#)
getOtherConstraintResultEnumerationNumberOfEI
 OSResult, [759](#)
getOtherConstraintResultEnumerationValue
 OSResult, [758](#)
getOtherConstraintResultName
 OSResult, [758](#)
getOtherConstraintResultNumberOfCon
 OSResult, [758](#)
getOtherConstraintResultNumberOfEnumerations
 OSResult, [758](#)
getOtherConstraintResultType
 OSResult, [758](#)
getOtherConstraintResultValue
 OSResult, [758](#)
getOtherGeneralOptions
 OSOption, [692](#)
getOtherGeneralResultDescription
 OSResult, [748](#)
getOtherGeneralResultName
 OSResult, [747](#)
getOtherGeneralResultValue
 OSResult, [748](#)
getOtherJobOptions
 OSOption, [693](#)
getOtherJobResultDescription
 OSResult, [750](#)
getOtherJobResultName
 OSResult, [750](#)
getOtherJobResultValue
 OSResult, [750](#)
getOtherObjectiveOption
 OSOption, [700](#)
getOtherObjectiveOptions
 OSOption, [700](#)
getOtherObjectiveResultArrayDense
 OSResult, [757](#)
getOtherObjectiveResultArrayType
 OSResult, [756](#)
getOtherObjectiveResultDescription
 OSResult, [756](#)
getOtherObjectiveResultEnumerationDescription
 OSResult, [756](#)
getOtherObjectiveResultEnumerationEI
 OSResult, [757](#)
getOtherObjectiveResultEnumerationNumberOfEI
 OSResult, [757](#)
getOtherObjectiveResultEnumerationValue
 OSResult, [756](#)
getOtherObjectiveResultName
 OSResult, [756](#)
getOtherObjectiveResultNumberOfEnumerations
 OSResult, [756](#)
getOtherObjectiveResultNumberOfObj
 OSResult, [756](#)
getOtherObjectiveResultObj
 OSResult, [756](#)
getOtherObjectiveResultObjIdx
 OSResult, [756](#)
getOtherObjectiveResultType
 OSResult, [756](#)
getOtherObjectiveResultValue
 OSResult, [756](#)

- getOtherOptions
 - OSOption, [693](#)
- getOtherServiceOptions
 - OSOption, [692](#)
- getOtherServiceResultDescription
 - OSResult, [748](#)
- getOtherServiceResultName
 - OSResult, [748](#)
- getOtherServiceResultValue
 - OSResult, [748](#)
- getOtherSolutionResultCategory
 - OSResult, [760](#)
- getOtherSolutionResultDescription
 - OSResult, [760](#)
- getOtherSolutionResultItem
 - OSResult, [760](#)
- getOtherSolutionResultName
 - OSResult, [760](#)
- getOtherSolutionResultNumberOfItems
 - OSResult, [760](#)
- getOtherSolutionResultValue
 - OSResult, [760](#)
- getOtherSystemOptions
 - OSOption, [692](#)
- getOtherSystemResultDescription
 - OSResult, [748](#)
- getOtherSystemResultName
 - OSResult, [748](#)
- getOtherSystemResultValue
 - OSResult, [748](#)
- getOtherVariableOption
 - OSOption, [698](#)
- getOtherVariableOptions
 - OSOption, [698](#)
- getOtherVariableResultArrayDense
 - OSResult, [755](#)
- getOtherVariableResultArrayType
 - OSResult, [754](#)
- getOtherVariableResultDescription
 - OSResult, [753](#)
- getOtherVariableResultEnumerationDescription
 - OSResult, [754](#)
- getOtherVariableResultEnumerationEI
 - OSResult, [755](#)
- getOtherVariableResultEnumerationNumberOfEI
 - OSResult, [754](#)
- getOtherVariableResultEnumerationValue
 - OSResult, [754](#)
- getOtherVariableResultName
 - OSResult, [753](#)
- getOtherVariableResultNumberOfEnumerations
 - OSResult, [753](#)
- getOtherVariableResultNumberOfVar
 - OSResult, [753](#)
- getOtherVariableResultType
 - OSResult, [753](#)
- getOtherVariableResultValue
 - OSResult, [753](#)
- getOtherVariableResultVar
 - OSResult, [754](#)
- getOtherVariableResultVarIdx
 - OSResult, [754](#)
- getOutputDirectoriesToMove
 - OSOption, [694](#)
- getOutputFilesToMove
 - OSOption, [694](#)
- getPassword
 - OSOption, [687](#)
- getPostfixFromExpressionTree
 - ExprNode, [135](#)
 - MatrixExpressionTree, [301](#)
 - OSnLMNode, [508](#)
 - OSnLNode, [558](#)
 - ScalarExpressionTree, [942](#)
- getPostfixFromNodeTree
 - MatrixNode, [305](#)
- getPrefixFromExpressionTree
 - ExprNode, [135](#)
 - MatrixExpressionTree, [301](#)
 - OSnLMNode, [507](#)
 - OSnLNode, [557](#)
 - ScalarExpressionTree, [942](#)
- getPrefixFromNodeTree
 - MatrixNode, [304](#)
- getProcessesToKill
 - OSOption, [695](#)
- getQuadraticRowIndex
 - OSInstance, [451](#)
- getQuadraticTerms
 - OSInstance, [451](#)
- GetRawPtr
 - OSSmartPtr, [836](#)
 - OSSmartPtr.hpp, [1352](#)
- getRequestedStartTime
 - OSOption, [688](#)
- getRequiredDirectories
 - OSOption, [693](#)
- getRequiredFiles
 - OSOption, [693](#)
- getRowMajor
 - MatrixElements, [292](#)
- getRowPartition
 - OSMatrix, [491](#)
- getRowPartitionSize
 - OSMatrix, [491](#)
- getSOSVariableBranchingWeightsSparse
 - OSOption, [698](#)
- getScheduledStartTime

- OSResult, [749](#)
- getSecondOrderResults
 - OSInstance, [481](#)
- getServiceName
 - OSOption, [687](#)
 - OSResult, [746](#)
- getServiceType
 - OSOption, [688](#)
- getServiceURI
 - OSOption, [687](#)
 - OSResult, [746](#)
- getServiceUtilization
 - OSResult, [748](#)
- getSlackVariableInitialBasisStatusDense
 - OSOption, [702](#)
- getSolutionMessage
 - OSResult, [752](#)
- getSolutionNumber
 - OSResult, [750](#)
- getSolutionStatus
 - OSResult, [750](#)
- getSolutionStatusDescription
 - OSResult, [751](#)
- getSolutionStatusType
 - OSResult, [751](#)
- getSolutionSubstatusDescription
 - OSResult, [751](#)
- getSolutionSubstatusType
 - OSResult, [751](#)
- getSolutionTargetObjectiveIdx
 - OSResult, [751](#)
- getSolutionTargetObjectiveName
 - OSResult, [751](#)
- getSolutionWeightedObjectives
 - OSResult, [751](#)
- getSolverInvoked
 - OSResult, [747](#)
- getSolverOptions
 - OSOption, [703](#)
- getSolverOutputCategory
 - OSResult, [760](#)
- getSolverOutputDescription
 - OSResult, [760](#)
- getSolverOutputItem
 - OSResult, [760](#)
- getSolverOutputName
 - OSResult, [760](#)
- getSolverOutputNumberOfItems
 - OSResult, [760](#)
- getSolverToInvoke
 - OSOption, [687](#)
- getSparseJacobianFromColumnMajor
 - OSInstance, [478](#)
- getSparseJacobianFromRowMajor
 - OSInstance, [478](#)
- getSystemInformation
 - OSResult, [748](#)
- getTimeDomainFormat
 - OSInstance, [459](#)
- getTimeDomainIntervalHorizon
 - OSInstance, [461](#)
- getTimeDomainIntervalStart
 - OSInstance, [461](#)
- getTimeDomainStageConList
 - OSInstance, [460](#)
- getTimeDomainStageNames
 - OSInstance, [460](#)
- getTimeDomainStageNumber
 - OSInstance, [459](#)
- getTimeDomainStageNumberOfConstraints
 - OSInstance, [460](#)
- getTimeDomainStageNumberOfObjectives
 - OSInstance, [460](#)
- getTimeDomainStageNumberOfVariables
 - OSInstance, [460](#)
- getTimeDomainStageObjList
 - OSInstance, [460](#)
- getTimeDomainStageVarList
 - OSInstance, [460](#)
- getTimeNumber
 - OSResult, [749](#)
- getTimeServiceStarted
 - OSResult, [748](#)
- getTimeStamp
 - OSResult, [747](#)
- getTimeValue
 - OSResult, [749](#)
- getTimingInfoCategory
 - OSResult, [749](#)
- getTimingInfoDescription
 - OSResult, [749](#)
- getTimingInfoType
 - OSResult, [749](#)
- getTimingInfoUnit
 - OSResult, [749](#)
- getTimingInfoValue
 - OSResult, [749](#)
- getTokenName
 - ExprNode, [134](#)
 - OSnLMNodeDiagonalMatrixFromVector, [511](#)
 - OSnLMNodeIdentityMatrix, [513](#)
 - OSnLMNodeMatrixCon, [515](#)
 - OSnLMNodeMatrixDiagonal, [517](#)
 - OSnLMNodeMatrixDotTimes, [519](#)
 - OSnLMNodeMatrixInverse, [521](#)
 - OSnLMNodeMatrixLowerTriangle, [523](#)
 - OSnLMNodeMatrixMinus, [526](#)
 - OSnLMNodeMatrixNegate, [528](#)

- OSnLMNodeMatrixObj, [530](#)
- OSnLMNodeMatrixPlus, [532](#)
- OSnLMNodeMatrixProduct, [534](#)
- OSnLMNodeMatrixReference, [537](#)
- OSnLMNodeMatrixScalarTimes, [539](#)
- OSnLMNodeMatrixSubmatrixAt, [541](#)
- OSnLMNodeMatrixSum, [543](#)
- OSnLMNodeMatrixTimes, [545](#)
- OSnLMNodeMatrixTranspose, [547](#)
- OSnLMNodeMatrixUpperTriangle, [549](#)
- OSnLMNodeMatrixVar, [552](#)
- OSnLNodeAbs, [561](#)
- OSnLNodeAllDiff, [564](#)
- OSnLNodeCos, [567](#)
- OSnLNodeDivide, [570](#)
- OSnLNodeE, [573](#)
- OSnLNodeErf, [577](#)
- OSnLNodeExp, [579](#)
- OSnLNodeIf, [582](#)
- OSnLNodeLn, [585](#)
- OSnLNodeMatrixDeterminant, [588](#)
- OSnLNodeMatrixToScalar, [591](#)
- OSnLNodeMatrixTrace, [594](#)
- OSnLNodeMax, [597](#)
- OSnLNodeMin, [600](#)
- OSnLNodeMinus, [603](#)
- OSnLNodeNegate, [606](#)
- OSnLNodeNumber, [610](#)
- OSnLNodePI, [613](#)
- OSnLNodePlus, [617](#)
- OSnLNodePower, [619](#)
- OSnLNodeProduct, [622](#)
- OSnLNodeSin, [625](#)
- OSnLNodeSqrt, [628](#)
- OSnLNodeSquare, [631](#)
- OSnLNodeSum, [634](#)
- OSnLNodeTimes, [637](#)
- OSnLNodeVariable, [641](#)
- getTokenNumber
 - ExprNode, [134](#)
 - OSnLMNodeMatrixCon, [515](#)
 - OSnLMNodeMatrixObj, [530](#)
 - OSnLMNodeMatrixReference, [537](#)
 - OSnLMNodeMatrixVar, [552](#)
 - OSnLNodeE, [573](#)
 - OSnLNodeNumber, [610](#)
 - OSnLNodePI, [613](#)
 - OSnLNodeVariable, [641](#)
- getTotalJobsSoFar
 - OSResult, [748](#)
- getUsedCPUNumberDescription
 - OSResult, [749](#)
- getUsedCPUNumberValue
 - OSResult, [750](#)
- getUsedCPUSpeedDescription
 - OSResult, [749](#)
- getUsedCPUSpeedUnit
 - OSResult, [749](#)
- getUsedCPUSpeedValue
 - OSResult, [749](#)
- getUsedDiskSpaceDescription
 - OSResult, [749](#)
- getUsedDiskSpaceUnit
 - OSResult, [749](#)
- getUsedDiskSpaceValue
 - OSResult, [749](#)
- getUsedMemoryDescription
 - OSResult, [749](#)
- getUsedMemoryUnit
 - OSResult, [749](#)
- getUsedMemoryValue
 - OSResult, [749](#)
- getUserName
 - OSOption, [687](#)
- getValue
 - OtherOptionEnumeration, [872](#)
- getVarValue
 - OSResult, [752](#)
- getVarValueIdx
 - OSResult, [752](#)
- getVarValueName
 - OSResult, [752](#)
- getVarValueString
 - OSResult, [752](#)
- getVarValueStringIdx
 - OSResult, [752](#)
- getVarValueStringName
 - OSResult, [752](#)
- getVariableIndexMap
 - OSnLNode, [556](#)
 - OSnLNodeVariable, [641](#)
- getVariableIndicesMap
 - ScalarExpressionTree, [942](#)
- getVariableInitialBasisStatusDense
 - OSOption, [696](#)
- getVariableLowerBounds
 - OSInstance, [446](#)
- getVariableNames
 - OSInstance, [444](#)
- getVariableNumber
 - OSInstance, [444](#)
 - OSResult, [750](#)
- getVariableTypes
 - OSInstance, [444](#)
- getVariableUpperBounds
 - OSInstance, [446](#)
- getZeroOrderResults
 - OSInstance, [480](#)

- groupWeight
 - OSoLParserData, [666](#)
 - SOSWeights, [969](#)
- groupWeightAttributePresent
 - OSoLParserData, [664](#)
- HEADEREND
 - OSParseosil.tab.hpp, [1133](#), [1144](#), [1150](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1224](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1297](#)
- HEADERSTART
 - OSParseosil.tab.hpp, [1133](#), [1144](#), [1150](#)
 - OSParseosol.tab.hpp, [1208](#), [1218](#), [1224](#)
 - OSParseosrl.tab.hpp, [1281](#), [1291](#), [1297](#)
- HORIZONATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1280](#)
- HEADEREND
 - OSParseosil.tab.hpp, [1115](#)
 - OSParseosol.tab.hpp, [1191](#)
 - OSParseosrl.tab.hpp, [1254](#)
- HEADERSTART
 - OSParseosil.tab.hpp, [1115](#)
 - OSParseosol.tab.hpp, [1191](#)
 - OSParseosrl.tab.hpp, [1254](#)
- HORIZONATT
 - OSParseosil.tab.hpp, [1113](#)
- haveExpandedForm
 - MatrixType, [323](#)
- hessColIdx
 - SparseHessianMatrix, [971](#)
- hessDimension
 - SparseHessianMatrix, [971](#)
- hessRowIdx
 - SparseHessianMatrix, [971](#)
- hessValues
 - SparseHessianMatrix, [971](#)
- horizon
 - TimeDomainInterval, [989](#)
- IDATT
 - OSParseosil.tab.hpp, [1138](#), [1148](#), [1158](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)
- IDENTITYMATRIXEND
 - OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)
- IDENTITYMATRIXSTART
 - OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)
- IDXATT
 - OSParseosil.tab.hpp, [1135](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1210](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1297](#)
- IDXEND
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- IDXONEATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1278](#)
- IDXSTART
 - OSParseosil.tab.hpp, [1151](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- IDXTWOATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1278](#)
- IFEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- IFSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- INCLUDEDIAGONALATT
 - OSParseosil.tab.hpp, [1138](#), [1148](#), [1158](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)
- INCRATT
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1297](#)
- INDEXESEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- INDEXESSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- INITIALBASISSTATUSEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALBASISSTATUSSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALCONSTRAINTVALUESSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)

- INITIALCONSTRAINTVALUESSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- INITIALDUALVALUESEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- INITIALDUALVALUESSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- INITIALOBJECTIVEBOUNDSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALOBJECTIVEBOUNDSSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALOBJECTIVEVALUESEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALOBJECTIVEVALUESSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALVARIABLEVALUESEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALVARIABLEVALUESSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALVARIABLEVALUESSTRINGEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INITIALVARIABLEVALUESSTRINGSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INPUTDIRECTORIESTOMOVEEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- INPUTDIRECTORIESTOMOVESTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- INPUTFILESTOMOVEEND
 - OSParseosil.tab.hpp, [1142](#)
- INPUTFILESTOMOVESTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- INSTANCEDATAEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- INSTANCEDATASTARTEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1278](#)
- INSTANCELOCATIONEND
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- INSTANCELOCATIONSTART
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- INSTANCENAMEEND
 - OSParseosil.tab.hpp, [1140](#), [1151](#)
 - OSParseosol.tab.hpp, [1215](#), [1226](#)
 - OSParseosrl.tab.hpp, [1288](#), [1299](#)
- INSTANCENAMESTART
 - OSParseosil.tab.hpp, [1140](#), [1151](#)
 - OSParseosol.tab.hpp, [1215](#), [1226](#)
 - OSParseosrl.tab.hpp, [1288](#), [1299](#)
- INTEGER
 - OSParseosil.tab.hpp, [1131](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- INTEGERVARIABLEBRANCHINGWEIGHTSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INTEGERVARIABLEBRANCHINGWEIGHTSSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- INTERSECTIONCONEEND
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- INTERSECTIONCONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- INTERVALEND
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1281](#)

- INTERVALSTART
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1281](#)
- ISFREEEND
 - OSParseosil.tab.hpp, [1142](#), [1151](#)
 - OSParseosol.tab.hpp, [1217](#), [1226](#)
 - OSParseosrl.tab.hpp, [1290](#), [1299](#)
- ISFREESTART
 - OSParseosil.tab.hpp, [1142](#), [1151](#)
 - OSParseosol.tab.hpp, [1217](#), [1226](#)
 - OSParseosrl.tab.hpp, [1290](#), [1299](#)
- ITEMEMPTY
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- ITEMEND
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- ITEMSTART
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- ITEMSTARTANDEND
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1150](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1225](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1298](#)
- ITEMTEXT
 - OSParseosil.tab.hpp, [1131](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- IDATT
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1277](#)
- IDENTITYMATRIXEND
 - OSParseosil.tab.hpp, [1129](#)
 - OSParseosol.tab.hpp, [1204](#)
 - OSParseosrl.tab.hpp, [1277](#)
- IDENTITYMATRIXSTART
 - OSParseosil.tab.hpp, [1129](#)
 - OSParseosol.tab.hpp, [1204](#)
 - OSParseosrl.tab.hpp, [1277](#)
- IDXATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1252](#)
- IDXEND
 - OSParseosrl.tab.hpp, [1258](#)
- IDXONEATT
 - OSParseosil.tab.hpp, [1107](#)
- IDXSTART
 - OSParseosrl.tab.hpp, [1258](#)
- IDXTWOATT
 - OSParseosil.tab.hpp, [1107](#)
- IFEND
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- IFSTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- INCLUDEDIAGONALATT
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1204](#)
 - OSParseosrl.tab.hpp, [1277](#)
- INCRATT
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1252](#)
- INDEXESEND
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- INDEXESSTART
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- INITIALDUALVALUESEND
 - OSParseosol.tab.hpp, [1189](#)
- INPUTFILESTOMOVEEND
 - OSParseosol.tab.hpp, [1185](#)
- INSTANCEDATAEND
 - OSParseosil.tab.hpp, [1107](#)
- INSTANCEDATASTARTEND
 - OSParseosil.tab.hpp, [1107](#)
- INSTANCELOCATIONEND
 - OSParseosol.tab.hpp, [1182](#)
- INSTANCENAMEEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1258](#)
- INSTANCENAMESTART
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1258](#)
- INTEGER
 - OSParseosil.tab.hpp, [1106](#)
 - OSParseosol.tab.hpp, [1175](#)
 - OSParseosrl.tab.hpp, [1249](#)
- INTERSECTIONCONEEND
 - OSParseosil.tab.hpp, [1109](#)
- INTERVALEND
 - OSParseosil.tab.hpp, [1115](#)
- INTERVALSTART
 - OSParseosil.tab.hpp, [1115](#)
- iNumberOfStartElements
 - LinearConstraintCoefficients, [261](#)

- iOption
 - OSnLParserData, [648](#)
 - OSoLParserData, [670](#)
- iOther
 - OSnLParserData, [648](#)
 - OSoLParserData, [670](#)
 - OSrLParserData, [819](#)
- ISFREEEND
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1258](#)
- ISFREESTART
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1258](#)
- ITEMEMPTY
 - OSParseosil.tab.hpp, [1116](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1255](#)
- ITEMEND
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1255](#)
- ITEMSTART
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1254](#)
- ITEMSTARTANDEND
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1255](#)
- ITEMTEXT
 - OSParseosil.tab.hpp, [1106](#)
 - OSParseosol.tab.hpp, [1175](#)
 - OSParseosrl.tab.hpp, [1249](#)
- id
 - OSnLNodeNumber, [611](#)
- idx
 - BranchingWeight, [52](#)
 - Cone, [65](#)
 - DualCone, [122](#)
 - DualVarValue, [126](#)
 - IndexStringPair, [165](#)
 - IndexValuePair, [166](#)
 - InitBasStatus, [169](#)
 - InitConValue, [175](#)
 - InitDualVarValue, [182](#)
 - InitObjBound, [188](#)
 - InitObjValue, [199](#)
 - InitVarValue, [210](#)
 - InitVarValueString, [213](#)
 - IntersectionCone, [226](#)
 - LinearMatrixElementTerm, [268](#)
 - MatrixExpression, [295](#)
 - NI, [341](#)
 - ObjCoef, [351](#)
 - ObjValue, [373](#)
 - OSgLParserData, [412](#)
 - OSMatrix, [494](#)
 - OSnLMNodeMatrixCon, [515](#)
 - OSnLMNodeMatrixObj, [530](#)
 - OSnLMNodeMatrixReference, [537](#)
 - OSnLMNodeMatrixVar, [552](#)
 - OSnLNodeVariable, [642](#)
 - OSrLParserData, [819](#)
 - OtherConOption, [842](#)
 - OtherConResult, [845](#)
 - OtherObjOption, [864](#)
 - OtherObjResult, [867](#)
 - OtherVarOption, [900](#)
 - OtherVarResult, [902](#)
 - PolarCone, [912](#)
 - PolyhedralCone, [916](#)
 - ProductCone, [923](#)
 - QuadraticCone, [928](#)
 - QuadraticTerm, [930](#)
 - RotatedQuadraticCone, [935](#)
 - SemidefiniteCone, [947](#)
 - TimeDomainStageCon, [992](#)
 - TimeDomainStageObj, [994](#)
 - TimeDomainStageVar, [998](#)
 - VarValue, [1031](#)
 - VarValueString, [1034](#)
- idxArray
 - OSoLParserData, [670](#)
- idxAttribute
 - OSnLParserData, [648](#)
 - OSoLParserData, [669](#)
- idxAttributePresent
 - OSnLParserData, [648](#)
 - OSoLParserData, [663](#)
 - OSrLParserData, [821](#)
- idxOne
 - QuadraticTerm, [930](#)
- idxPresent
 - OSgLParserData, [412](#)
- idxTwo
 - QuadraticTerm, [930](#)
- ignoreDataAfterErrors
 - OSgLParserData, [411](#)
 - OSiLParserData, [430](#)
 - OSnLParserData, [652](#)
 - OSoLParserData, [671](#)
 - OSrLParserData, [827](#)
- includeDiagonal
 - OSnLMNodeMatrixLowerTriangle, [524](#)
 - OSnLMNodeMatrixUpperTriangle, [550](#)
- includeDiagonalAttribute
 - OSnLParserData, [651](#)
- includeDiagonalAttributePresent

- OSnLParserData, 651
- incr
 - OSrLParserData, 821
- incrPresent
 - OSrLParserData, 821
- IndexStringPair, 165
 - idx, 165
 - value, 166
- IndexValuePair, 166
 - idx, 166
 - value, 166
- indexes
 - GeneralSparseMatrix, 159
 - MatrixElements, 292
 - SparseIntVector, 973
 - SparseJacobianMatrix, 975
 - SparseMatrix, 977
 - SparseVector, 979
- InitBasStatus, 166
 - ~InitBasStatus, 168
 - deepCopyFrom, 168
 - idx, 169
 - InitBasStatus, 168
 - InitBasStatus, 168
 - IsEqual, 168
 - setRandom, 168
 - value, 169
- InitConValue, 173
 - ~InitConValue, 174
 - deepCopyFrom, 175
 - idx, 175
 - InitConValue, 174
 - InitConValue, 174
 - IsEqual, 175
 - name, 175
 - setRandom, 175
 - value, 175
- InitConstraintValues, 169
 - ~InitConstraintValues, 171
 - addCon, 172
 - con, 173
 - deepCopyFrom, 172
 - InitConstraintValues, 171
 - InitConstraintValues, 171
 - IsEqual, 171
 - numberOfCon, 173
 - setCon, 172
 - setRandom, 171
- InitDualVarValue, 179
 - ~InitDualVarValue, 181
 - deepCopyFrom, 181
 - idx, 182
 - InitDualVarValue, 181
 - InitDualVarValue, 181
- IsEqual, 181
- lbDualValue, 182
- name, 182
- setRandom, 181
- ubDualValue, 182
- InitDualVariableValues, 176
 - ~InitDualVariableValues, 177
 - addCon, 179
 - con, 179
 - deepCopyFrom, 178
 - InitDualVariableValues, 177
 - InitDualVariableValues, 177
 - IsEqual, 177
 - numberOfCon, 179
 - setCon, 178
 - setRandom, 177
- initForAlgDiff
 - OSInstance, 481
- initGMO
 - OSgams2osil, 405
- InitObjBound, 185
 - ~InitObjBound, 187
 - deepCopyFrom, 187
 - idx, 188
 - InitObjBound, 187
 - InitObjBound, 187
 - IsEqual, 187
 - lbValue, 188
 - name, 188
 - setRandom, 187
 - ubValue, 188
- initObjGradients
 - OSInstance, 481
- InitObjValue, 196
 - ~InitObjValue, 198
 - deepCopyFrom, 198
 - idx, 199
 - InitObjValue, 198
 - InitObjValue, 198
 - IsEqual, 198
 - name, 199
 - setRandom, 198
 - value, 199
- InitObjectiveBounds, 188
 - ~InitObjectiveBounds, 190
 - addObj, 191
 - deepCopyFrom, 191
 - InitObjectiveBounds, 190
 - InitObjectiveBounds, 190
 - IsEqual, 190
 - numberOfObj, 192
 - obj, 192
 - setObj, 191
 - setRandom, 190

- InitObjectiveValues, 192
 - ~InitObjectiveValues, 194
 - addObj, 195
 - deepCopyFrom, 195
 - InitObjectiveValues, 194
 - InitObjectiveValues, 194
 - IsEqual, 194
 - numberOfObj, 196
 - obj, 196
 - setObj, 195
 - setRandom, 194
- InitVarValue, 207
 - ~InitVarValue, 209
 - deepCopyFrom, 209
 - idx, 210
 - InitVarValue, 209
 - InitVarValue, 209
 - IsEqual, 209
 - name, 210
 - setRandom, 209
 - value, 210
- InitVarValueString, 210
 - ~InitVarValueString, 212
 - deepCopyFrom, 212
 - idx, 213
 - InitVarValueString, 212
 - InitVarValueString, 212
 - IsEqual, 212
 - name, 213
 - setRandom, 212
 - value, 213
- InitVariableValues, 199
 - ~InitVariableValues, 201
 - addVar, 202
 - deepCopyFrom, 202
 - InitVariableValues, 201
 - InitVariableValues, 201
 - IsEqual, 201
 - numberOfVar, 203
 - setRandom, 201
 - setVar, 202
 - var, 203
- InitVariableValuesString, 203
 - ~InitVariableValuesString, 205
 - addVar, 206, 207
 - deepCopyFrom, 206
 - InitVariableValuesString, 205
 - InitVariableValuesString, 205
 - IsEqual, 205
 - numberOfVar, 207
 - setRandom, 205
 - setVar, 206
 - var, 207
- InitialBasisStatus, 182
 - ~InitialBasisStatus, 184
 - addVar, 185
 - deepCopyFrom, 185
 - InitialBasisStatus, 184
 - InitialBasisStatus, 184
 - IsEqual, 184
 - numberOfVar, 185
 - setRandom, 184
 - setVar, 185
 - var, 185
- initialBasisStatus
 - ConstraintOption, 86
 - ObjectiveOption, 357
 - VariableOption, 1013
- initialBasisStatusPresent
 - OSoLParserData, 664
- initialConstraintValues
 - ConstraintOption, 85
- initialConstraintValuesPresent
 - OSoLParserData, 665
- initialDualValues
 - ConstraintOption, 85
- initialDualVariableValuesPresent
 - OSoLParserData, 665
- initialObjectiveBounds
 - ObjectiveOption, 357
- initialObjectiveBoundsPresent
 - OSoLParserData, 665
- initialObjectiveValues
 - ObjectiveOption, 356
- initialObjectiveValuesPresent
 - OSoLParserData, 665
- initialVariableValues
 - VariableOption, 1013
- initialVariableValuesPresent
 - OSoLParserData, 664
- initialVariableValuesString
 - VariableOption, 1013
- initialVariableValuesStringPresent
 - OSoLParserData, 664
- initializeNonLinearStructures
 - OSInstance, 474
- inodeInt
 - ExprNode, 136
- inodeType
 - ExprNode, 136
- inputDirectoriesToMove
 - JobOption, 244
- inputDirectoriesToMovePresent
 - OSoLParserData, 661
- inputFilesToMove
 - JobOption, 244
- inputFilesToMovePresent
 - OSoLParserData, 661

- insList
 - OSCommandLine, 397
 - osOptionsStruc, 717
- insListFile
 - OSCommandLine, 397
 - osOptionsStruc, 717
- InstanceData, 213
 - ~InstanceData, 215
 - cones, 216
 - constraints, 215
 - InstanceData, 215
 - InstanceData, 215
 - IsEqual, 215
 - linearConstraintCoefficients, 215
 - matrices, 215
 - matrixProgramming, 216
 - nonlinearExpressions, 215
 - objectives, 215
 - quadraticCoefficients, 215
 - timeDomain, 216
 - variables, 215
- instanceData
 - OSInstance, 483
- instanceHeader
 - OSInstance, 482
- instanceLocation
 - GeneralOption, 153
- InstanceLocationOption, 216
 - ~InstanceLocationOption, 217
 - deepCopyFrom, 218
 - InstanceLocationOption, 217
 - InstanceLocationOption, 217
 - IsEqual, 218
 - locationType, 218
 - setRandom, 218
 - value, 218
- instanceLocationPresent
 - OSoLParserData, 658
- instanceLocationTypeattON
 - OSoLParserData, 658
- instanceName
 - GeneralOption, 152
 - GeneralResult, 156
 - OSMatlab, 488
- instanceNamePresent
 - OSoLParserData, 658
- IntVector, 226
 - ~IntVector, 228
 - bDeleteArrays, 229
 - deepCopyFrom, 228
 - el, 229
 - extendIntVector, 229
 - getEI, 229
 - getNumberOfEI, 229
 - IntVector, 228
 - IntVector, 228
 - IsEqual, 228
 - numberOfEI, 229
 - setIntVector, 228
 - setRandom, 228
- IntegerVariableBranchingWeights, 218
 - ~IntegerVariableBranchingWeights, 220
 - addVar, 221, 222
 - deepCopyFrom, 221
 - IntegerVariableBranchingWeights, 220
 - IntegerVariableBranchingWeights, 220
 - IsEqual, 220
 - numberOfVar, 222
 - setRandom, 220
 - setVar, 221
 - var, 222
- integerVariableBranchingWeights
 - VariableOption, 1013
- IntersectionCone, 222
 - ~IntersectionCone, 224
 - components, 226
 - coneType, 226
 - deepCopyFrom, 225
 - getConeInXML, 224
 - getConeName, 224
 - idx, 226
 - IntersectionCone, 224
 - IntersectionCone, 224
 - IsEqual, 225
 - numberOfColumns, 225
 - numberOfOtherIndexes, 225
 - numberOfRows, 225
 - otherIndexes, 226
 - setRandom, 225
- Interval, 226
- interval
 - TimeDomain, 988
- intervalhorizon
 - OSiLParserData, 426
- intervalhorizonON
 - OSiLParserData, 425
- intervalstart
 - OSiLParserData, 426
- intervalstartON
 - OSiLParserData, 426
- inumberOfChildren
 - ExprNode, 137
 - MatrixNode, 306
- inumberOfMatrixChildren
 - ExprNode, 137
- invokeHelp
 - OSCommandLine, 399
 - osOptionsStruc, 719

- ipoptErrorMsg
 - IpoptProblem, 233
- IpoptProblem, 230
 - ~IpoptProblem, 232
 - eval_f, 232
 - eval_g, 232
 - eval_grad_f, 232
 - eval_h, 232
 - eval_jac_g, 232
 - finalize_solution, 233
 - get_bounds_info, 232
 - get_nlp_info, 232
 - get_scaling_parameters, 233
 - get_starting_point, 232
 - ipoptErrorMsg, 233
 - IpoptProblem, 232
 - IpoptProblem, 232
 - osinstance, 233
 - osoption, 233
 - osresult, 233
- IpoptSolver, 233
 - ~IpoptSolver, 236
 - app, 237
 - buildSolverInstance, 237
 - dataEchoCheck, 237
 - IpoptSolver, 236
 - IpoptSolver, 236
 - m_osilreader, 237
 - m_osolreader, 237
 - nlp, 237
 - setSolverOptions, 237
 - solve, 237
- isAccepted
 - OSOutputChannel, 727
- isBlockDiagonal
 - OSMatrix, 493
- isColumnMajor
 - ExpandedMatrixBlocks, 130
 - GeneralSparseMatrix, 159
 - SparseMatrix, 977
- IsEqual
 - BaseMatrix, 35
 - BasisStatus, 38
 - BranchingWeight, 51
 - CompletelyPositiveMatricesCone, 59
 - Cone, 63
 - Cones, 66
 - ConReferenceMatrixElement, 68
 - ConReferenceMatrixElements, 72
 - ConReferenceMatrixValues, 74
 - ConstantMatrixElements, 78
 - ConstantMatrixValues, 80
 - Constraint, 82
 - ConstraintOption, 84
 - Constraints, 87
 - ConstraintSolution, 89
 - ContactOption, 92
 - CopositiveMatricesCone, 95
 - CPUNumber, 102
 - CPU Speed, 105
 - DirectoriesAndFiles, 116
 - DoubleVector, 118
 - DualCone, 120
 - DualVariableValues, 124
 - DualVarValue, 126
 - ExprNode, 136
 - GeneralFileHeader, 141
 - GeneralMatrixElements, 146
 - GeneralMatrixValues, 148
 - GeneralOption, 152
 - GeneralResult, 156
 - GeneralStatus, 162
 - GeneralSubstatus, 164
 - InitBasStatus, 168
 - InitConstraintValues, 171
 - InitConValue, 175
 - InitDualVariableValues, 177
 - InitDualVarValue, 181
 - InitialBasisStatus, 184
 - InitObjBound, 187
 - InitObjectiveBounds, 190
 - InitObjectiveValues, 194
 - InitObjValue, 198
 - InitVariableValues, 201
 - InitVariableValuesString, 205
 - InitVarValue, 209
 - InitVarValueString, 212
 - InstanceData, 215
 - InstanceLocationOption, 218
 - IntegerVariableBranchingWeights, 220
 - IntersectionCone, 225
 - IntVector, 228
 - JobDependencies, 239
 - JobOption, 242
 - JobResult, 247
 - LinearConstraintCoefficients, 260
 - LinearMatrixElement, 262
 - LinearMatrixElements, 266
 - LinearMatrixElementTerm, 268
 - LinearMatrixValues, 270
 - Matrices, 274
 - MatrixBlock, 278
 - MatrixBlocks, 282
 - MatrixCon, 285
 - MatrixConstraints, 288
 - MatrixElements, 292
 - MatrixExpression, 295
 - MatrixExpressions, 298

- MatrixExpressionTree, [301](#)
- MatrixNode, [305](#)
- MatrixObj, [308](#)
- MatrixObjectives, [311](#)
- MatrixProgramming, [313](#)
- MatrixTransformation, [317](#)
- MatrixType, [322](#)
- MatrixVar, [325](#)
- MatrixVariables, [328](#)
- MaxTime, [330](#)
- MinCPUNumber, [332](#)
- MinCPUSpeed, [334](#)
- MinDiskSpace, [336](#)
- MinMemorySize, [338](#)
- NI, [341](#)
- NonlinearExpressions, [344](#)
- NonnegativeCone, [347](#)
- NonpositiveCone, [349](#)
- ObjCoef, [351](#)
- Objective, [353](#)
- ObjectiveOption, [356](#)
- Objectives, [359](#)
- ObjectiveSolution, [361](#)
- ObjectiveValues, [364](#)
- ObjReferenceMatrixElements, [368](#)
- ObjReferenceMatrixValues, [370](#)
- ObjValue, [372](#)
- OptimizationOption, [375](#)
- OptimizationResult, [378](#)
- OptimizationSolution, [381](#)
- OptimizationSolutionStatus, [384](#)
- OptimizationSolutionSubstatus, [387](#)
- OrthantCone, [390](#)
- OSExpressionTree, [403](#)
- OSInstance, [443](#)
- OSMatrix, [493](#)
- OSnLMNode, [509](#)
- OSnLMNodeMatrixCon, [515](#)
- OSnLMNodeMatrixLowerTriangle, [523](#)
- OSnLMNodeMatrixObj, [530](#)
- OSnLMNodeMatrixReference, [537](#)
- OSnLMNodeMatrixUpperTriangle, [549](#)
- OSnLMNodeMatrixVar, [552](#)
- OSnLNode, [559](#)
- OSnLNodeNumber, [611](#)
- OSnLNodeVariable, [642](#)
- OSOption, [686](#)
- OSResult, [745](#)
- OtherConOption, [842](#)
- OtherConResult, [844](#)
- OtherConstraintOption, [848](#)
- OtherConstraintResult, [852](#)
- OtherObjectiveOption, [856](#)
- OtherObjectiveResult, [860](#)
- OtherObjOption, [864](#)
- OtherObjResult, [866](#)
- OtherOption, [869](#)
- OtherOptionEnumeration, [871](#)
- OtherOptions, [874](#)
- OtherResult, [877](#)
- OtherResults, [880](#)
- OtherSolutionResult, [883](#)
- OtherSolutionResults, [885](#)
- OtherSolverOutput, [888](#)
- OtherVariableOption, [890](#)
- OtherVariableResult, [895](#)
- OtherVarOption, [899](#)
- OtherVarResult, [902](#)
- PathPair, [905](#)
- PathPairs, [907](#)
- PolarCone, [911](#)
- PolyhedralCone, [915](#)
- Processes, [918](#)
- ProductCone, [922](#)
- QuadraticCoefficients, [924](#)
- QuadraticCone, [927](#)
- QuadraticTerm, [930](#)
- RotatedQuadraticCone, [934](#)
- RowReferenceMatrixElements, [939](#)
- ScalarExpressionTree, [942](#)
- SemidefiniteCone, [946](#)
- ServiceOption, [949](#)
- ServiceResult, [952](#)
- SolverOption, [955](#)
- SolverOptions, [959](#)
- SolverOutput, [962](#)
- SOSVariableBranchingWeights, [965](#)
- SOSWeights, [968](#)
- StorageCapacity, [980](#)
- SystemOption, [983](#)
- SystemResult, [986](#)
- TimeMeasurement, [1001](#)
- TimeSpan, [1004](#)
- TimingInformation, [1007](#)
- Variable, [1009](#)
- VariableOption, [1012](#)
- Variables, [1015](#)
- VariableSolution, [1017](#)
- VariableValues, [1020](#)
- VariableValuesString, [1022](#)
- VarReferenceMatrixElements, [1026](#)
- VarReferenceMatrixValues, [1029](#)
- VarValue, [1031](#)
- VarValueString, [1033](#)
- isFree
 - BasisStatus, [41](#)
- IsNull
 - OSSmartPtr, [836](#)

- OSSmartPtr.hpp, 1352
- isPositiveSemiDefinite
 - SemidefiniteCone, 947
- IsValid
 - OSSmartPtr, 836
 - OSSmartPtr.hpp, 1352
- item
 - OtherSolutionResult, 883
 - SolverOption, 957
 - SolverOutput, 963
- itemContent
 - OSoLParserData, 667
 - OSrLParserData, 818
- itemList
 - OSoLParserData, 670
- ival
 - YYSTYPE, 1038
- ivar
 - OSrLParserData, 819
- JOBEND
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1298
- JOBIDEND
 - OSParseosil.tab.hpp, 1141, 1151
 - OSParseosol.tab.hpp, 1215, 1226
 - OSParseosrl.tab.hpp, 1288, 1299
- JOBIDSTART
 - OSParseosil.tab.hpp, 1140, 1151
 - OSParseosol.tab.hpp, 1215, 1226
 - OSParseosrl.tab.hpp, 1288, 1299
- JOBSTART
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1297
- JOBEND
 - OSParseosol.tab.hpp, 1181
 - OSParseosrl.tab.hpp, 1254
- JOBIDEND
 - OSParseosol.tab.hpp, 1182
 - OSParseosrl.tab.hpp, 1258
- JOBIDSTART
 - OSParseosol.tab.hpp, 1182
 - OSParseosrl.tab.hpp, 1258
- JOBSTART
 - OSParseosol.tab.hpp, 1181
 - OSParseosrl.tab.hpp, 1254
- job
 - OSOption, 713
 - OSResult, 810
- JobDependencies, 238
 - ~JobDependencies, 239
 - addJobID, 240
 - deepCopyFrom, 239
 - IsEqual, 239
 - JobDependencies, 239
 - jobID, 240
 - JobDependencies, 239
 - numberOfJobIDs, 240
 - setJobID, 240
 - setRandom, 239
- jobDependencies
 - OSoLParserData, 669
- jobEndTimePresent
 - OSrLParserData, 825
- jobID
 - GeneralOption, 153
 - GeneralResult, 156
 - JobDependencies, 240
 - OSCommandLine, 398
 - OSmps2OS, 497
 - OSnl2OS, 504
 - osOptionsStruc, 719
- jobIDPresent
 - OSoLParserData, 658
- JobOption, 240
 - ~JobOption, 242
 - deepCopyFrom, 243
 - dependencies, 243
 - directoriesToDelete, 244
 - directoriesToMake, 243
 - filesToDelete, 244
 - filesToMake, 243
 - inputDirectoriesToMove, 244
 - inputFilesToMove, 244
 - IsEqual, 242
 - JobOption, 242
 - JobOption, 242
 - maxTime, 243
 - otherOptions, 244
 - outputDirectoriesToMove, 244
 - outputFilesToMove, 244
 - processesToKill, 244
 - requestedStartTime, 243
 - requiredDirectories, 243
 - requiredFiles, 243
 - setRandom, 242
- JobResult, 244
 - ~JobResult, 246
 - actualStartTime, 247
 - endTime, 247
 - IsEqual, 247
 - JobResult, 246
 - JobResult, 246
 - otherResults, 248
 - scheduledStartTime, 247
 - setRandom, 247

- status, [247](#)
- submitTime, [247](#)
- timingInformation, [247](#)
- usedCPUNumber, [248](#)
- usedCPUSpeed, [248](#)
- usedDiskSpace, [247](#)
- usedMemory, [247](#)
- jobStatusPresent
 - OSrLParserData, [825](#)
- jobSubmitTimePresent
 - OSrLParserData, [825](#)
- jobTimingInformationPresent
 - OSrLParserData, [825](#)
- jobUsedCPUNumberPresent
 - OSrLParserData, [825](#)
- jobUsedCPUSpeedPresent
 - OSrLParserData, [825](#)
- jobUsedDiskSpacePresent
 - OSrLParserData, [825](#)
- jobUsedMemoryPresent
 - OSrLParserData, [825](#)
- kill
 - OShL, [418](#)
 - OSSolverAgent, [839](#)
- knitroErrorMsg
 - KnitroProblem, [250](#)
- KnitroProblem, [248](#)
 - ~KnitroProblem, [250](#)
 - areDerivativesImplemented, [250](#)
 - evalFC, [250](#)
 - evalGA, [250](#)
 - evalH, [250](#)
 - evalHV, [250](#)
 - getInitialX, [250](#)
 - getM, [250](#)
 - getN, [250](#)
 - knitroErrorMsg, [250](#)
 - KnitroProblem, [250](#)
 - KnitroProblem, [250](#)
 - loadProblemIntoKnitro, [250](#)
 - osinstance, [250](#)
 - osresult, [250](#)
- KnitroSolver, [251](#)
 - ~KnitroSolver, [253](#)
 - buildSolverInstance, [253](#)
 - dataEchoCheck, [254](#)
 - KnitroSolver, [253](#)
 - KnitroSolver, [253](#)
 - setSolverOptions, [253](#)
 - solve, [254](#)
- knock
 - OShL, [418](#)
 - OSSolverAgent, [839](#)
- kount2
 - OSiLParserData, [430](#)
- kounter
 - OSiLParserData, [430](#)
 - OSnLParserData, [648](#)
 - OSoLParserData, [670](#)
 - OSrLParserData, [819](#)
- LBCONEIDXATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- LBDUALVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- LBMATRIXIDXATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- LBVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1287](#)
- LICENSEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- LICENSESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- LINEARELEMENTSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- LINEARELEMENTSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- LNEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- LNSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- LOCATIONTYPEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- LBCONEIDXATT
 - OSParseosil.tab.hpp, [1112](#)

- LBDUALVALUEATT
 - OSParseosol.tab.hpp, [1180](#)
- LBMATRIXIDXATT
 - OSParseosil.tab.hpp, [1112](#)
- LBVALUEATT
 - OSParseosol.tab.hpp, [1179](#)
- LICENSEEND
 - OSParseosol.tab.hpp, [1182](#)
- LICENSESTART
 - OSParseosol.tab.hpp, [1182](#)
- LINEARELEMENTSEND
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- LINEARELEMENTSSTART
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- LNEND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- LNSTART
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- LOCATIONTYPEATT
 - OSParseosol.tab.hpp, [1180](#)
- last_column
 - YYLTYPE, [1038](#)
- last_line
 - YYLTYPE, [1038](#)
- lb
 - Constraint, [82](#)
 - OrthantCone, [390](#)
 - Variable, [1009](#)
- lbConeldx
 - MatrixCon, [286](#)
 - MatrixVar, [326](#)
 - OSILParserData, [429](#)
- lbConeldxPresent
 - OSILParserData, [428](#)
- lbDualValue
 - InitDualVarValue, [182](#)
 - OSoLParserData, [666](#)
- lbMatrixIdx
 - MatrixCon, [285](#)
 - MatrixVar, [325](#)
 - OSILParserData, [429](#)
- lbMatrixIdxPresent
 - OSILParserData, [428](#)
- lbValArray
 - OSoLParserData, [670](#)
- lbValAttributePresent
 - OSoLParserData, [663](#)
- lbValue
 - InitObjBound, [188](#)
 - OtherConOption, [842](#)
 - OtherObjOption, [864](#)
 - OtherVarOption, [900](#)
- lbValueAttribute
 - OSnLParserData, [647](#)
 - OSoLParserData, [668](#)
- lbValueAttributePresent
 - OSnLParserData, [647](#)
 - OSoLParserData, [668](#)
- lbValueString
 - OSoLParserData, [670](#)
- licence
 - GeneralFileHeader, [143](#)
 - OSgLParserData, [410](#)
- licencePresent
 - OSgLParserData, [410](#)
- license
 - GeneralOption, [153](#)
- licensePresent
 - OSoLParserData, [658](#)
- lindoAPIErrorCheck
 - LindoSolver, [258](#)
- LindoSolver, [254](#)
 - ~LindoSolver, [257](#)
 - addSlackVars, [258](#)
 - buildSolverInstance, [257](#)
 - dataEchoCheck, [258](#)
 - generateLindoModel, [258](#)
 - lindoAPIErrorCheck, [258](#)
 - LindoSolver, [257](#)
 - LindoSolver, [257](#)
 - m_osilreader, [259](#)
 - optimize, [257](#)
 - processConstraints, [258](#)
 - processNonlinearExpressions, [258](#)
 - processQuadraticTerms, [258](#)
 - processVariables, [257](#)
 - setSolverOptions, [257](#)
 - solve, [257](#)
- LinearConstraintCoefficients, [259](#)
 - ~LinearConstraintCoefficients, [260](#)
- colIdx, [261](#)
- iNumberOfStartElements, [261](#)
- isEqual, [260](#)
- LinearConstraintCoefficients, [260](#)
- LinearConstraintCoefficients, [260](#)
- numberOfValues, [260](#)
- rowIdx, [260](#)
- start, [260](#)
- value, [261](#)

- InstanceData, 215
- LinearMatrixElement, 261
 - ~LinearMatrixElement, 262
 - constant, 263
 - deepCopyFrom, 262
 - IsEqual, 262
 - LinearMatrixElement, 262
 - LinearMatrixElement, 262
 - numberOfVarIdx, 263
 - setRandom, 262
 - varIdx, 263
- LinearMatrixElementTerm, 267
 - ~LinearMatrixElementTerm, 268
 - coef, 268
 - deepCopyFrom, 268
 - idx, 268
 - IsEqual, 268
 - LinearMatrixElementTerm, 268
 - LinearMatrixElementTerm, 268
 - setRandom, 268
- LinearMatrixElements, 263
 - ~LinearMatrixElements, 265
 - alignsOnBlockBoundary, 266
 - cloneMatrixNode, 266
 - deepCopyFrom, 267
 - getMatrixNodeInXML, 266
 - getMatrixType, 265
 - getNodeName, 265
 - getNodeType, 265
 - IsEqual, 266
 - LinearMatrixElements, 265
 - LinearMatrixElements, 265
 - setRandom, 266
 - values, 267
- LinearMatrixValues, 269
 - ~LinearMatrixValues, 270
 - deepCopyFrom, 270
 - el, 270
 - IsEqual, 270
 - LinearMatrixValues, 270
 - LinearMatrixValues, 270
 - setRandom, 270
- list_options
 - OSCommandLine, 395
- listOptions
 - OSCommandLine, 399
- loadProblemIntoKnitro
 - KnitroProblem, 250
- locationType
 - InstanceLocationOption, 218
- logFile
 - OSCommandLine, 398
 - osOptionsStruc, 719
- MAKECOPYATT
 - OSParseosil.tab.hpp, 1139
 - OSParseosol.tab.hpp, 1214
 - OSParseosrl.tab.hpp, 1286
- MATRICESEND
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- MATRICESSTART
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- MATRIXCONEND
 - OSParseosil.tab.hpp, 1132, 1144, 1153
 - OSParseosol.tab.hpp, 1207, 1218, 1228
 - OSParseosrl.tab.hpp, 1280, 1291, 1301
- MATRIXCONSTART
 - OSParseosil.tab.hpp, 1132, 1144, 1153
 - OSParseosol.tab.hpp, 1207, 1218, 1228
 - OSParseosrl.tab.hpp, 1280, 1291, 1301
- MATRIXCONSTRAINTSEND
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- MATRIXCONSTRAINTSSTART
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- MATRIXDETERMINANTEND
 - OSParseosil.tab.hpp, 1137, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1231
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXDETERMINANTSTART
 - OSParseosil.tab.hpp, 1137, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1231
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXDIAGONALEND
 - OSParseosil.tab.hpp, 1138, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1232
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXDIAGONALSTART
 - OSParseosil.tab.hpp, 1138, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1232
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXDOTTIMESEND
 - OSParseosil.tab.hpp, 1138, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1232
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXDOTTIMESSTART
 - OSParseosil.tab.hpp, 1138, 1147, 1157
 - OSParseosol.tab.hpp, 1212, 1222, 1232
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- MATRIXEND
 - OSParseosil.tab.hpp, 1134, 1144, 1154

OSParseosol.tab.hpp, 1209, 1219, 1228
OSParseosrl.tab.hpp, 1282, 1292, 1301
MATRIXEXPRESSIONSEND
 OSParseosil.tab.hpp, 1136, 1146, 1155
 OSParseosol.tab.hpp, 1211, 1221, 1230
 OSParseosrl.tab.hpp, 1284, 1294, 1303
MATRIXEXPRESSIONSSTART
 OSParseosil.tab.hpp, 1136, 1146, 1155
 OSParseosol.tab.hpp, 1211, 1221, 1230
 OSParseosrl.tab.hpp, 1284, 1294, 1303
MATRIXIDXATT
 OSParseosil.tab.hpp, 1132
 OSParseosol.tab.hpp, 1207
 OSParseosrl.tab.hpp, 1280
MATRIXINVERSEEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1296, 1305
MATRIXINVERSESTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1296, 1305
MATRIXLOWERTRIANGLEEND
 OSParseosil.tab.hpp, 1138, 1147, 1157
 OSParseosol.tab.hpp, 1212, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXLOWERTRIANGLESTART
 OSParseosil.tab.hpp, 1138, 1147, 1157
 OSParseosol.tab.hpp, 1212, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXMERGEEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXMERGESTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1212, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXMINUSEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXMINUSSTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXNEGATEEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305
MATRIXNEGATESTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1285, 1295, 1305

MATRIXOBJECTIVESEND
 OSParseosil.tab.hpp, 1132
 OSParseosol.tab.hpp, 1207
 OSParseosrl.tab.hpp, 1280
MATRIXOBJECTIVESSTART
 OSParseosil.tab.hpp, 1132
 OSParseosol.tab.hpp, 1207
 OSParseosrl.tab.hpp, 1280
MATRIXOBJEND
 OSParseosil.tab.hpp, 1132, 1143, 1153
 OSParseosol.tab.hpp, 1207, 1218, 1228
 OSParseosrl.tab.hpp, 1280, 1291, 1301
MATRIXOBJSTART
 OSParseosil.tab.hpp, 1132, 1143, 1153
 OSParseosol.tab.hpp, 1207, 1218, 1228
 OSParseosrl.tab.hpp, 1280, 1291, 1301
MATRIXPLUSEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1286, 1295, 1305
MATRIXPLUSSTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1222, 1232
 OSParseosrl.tab.hpp, 1286, 1295, 1305
MATRIXPRODUCTEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1295, 1305
MATRIXPRODUCTSTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1295, 1305
MATRIXPROGRAMMINGEND
 OSParseosil.tab.hpp, 1132
 OSParseosol.tab.hpp, 1207
 OSParseosrl.tab.hpp, 1280
MATRIXPROGRAMMINGSTART
 OSParseosil.tab.hpp, 1132
 OSParseosol.tab.hpp, 1207
 OSParseosrl.tab.hpp, 1280
MATRIXREFERENCEEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1296, 1305
MATRIXREFERENCESTART
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1296, 1305
MATRIXSCALARTIMESEND
 OSParseosil.tab.hpp, 1138, 1148, 1157
 OSParseosol.tab.hpp, 1213, 1223, 1232
 OSParseosrl.tab.hpp, 1286, 1295, 1305
MATRIXSCALARTIMESSTART
 OSParseosil.tab.hpp, 1138, 1148, 1157

OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1295](#), [1305](#)

MATRIXSTART
OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
OSParseosol.tab.hpp, [1209](#), [1219](#), [1228](#)
OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)

MATRIXSUBMATRIXATEND
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)

MATRIXSUBMATRIXATSTART
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)

MATRIXTERMEND
OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)

MATRIXTERMSTART
OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)

MATRIXTIMESEND
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1295](#), [1305](#)

MATRIXTIMESSTART
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1222](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1295](#), [1305](#)

MATRIXTOSCALAREND
OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1232](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MATRIXTOSCALARSTART
OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MATRIXTRACEEND
OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MATRIXTRACESTART
OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MATRIXTRANPOSEEND
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)

MATRIXTRANPOSESTART
OSParseosil.tab.hpp, [1138](#), [1148](#), [1157](#)
OSParseosol.tab.hpp, [1213](#), [1223](#), [1232](#)
OSParseosrl.tab.hpp, [1286](#), [1296](#), [1305](#)

MATRIXUPPERTRIANGLEEND
OSParseosil.tab.hpp, [1138](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1232](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1305](#)

MATRIXUPPERTRIANGLESTART
OSParseosil.tab.hpp, [1138](#), [1147](#), [1157](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1232](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1305](#)

MATRIXVAREND
OSParseosil.tab.hpp, [1132](#), [1143](#), [1153](#)
OSParseosol.tab.hpp, [1207](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1280](#), [1291](#), [1301](#)

MATRIXVARIABLESEND
OSParseosil.tab.hpp, [1132](#)
OSParseosol.tab.hpp, [1207](#)
OSParseosrl.tab.hpp, [1280](#)

MATRIXVARIABLESSTART
OSParseosil.tab.hpp, [1132](#)
OSParseosol.tab.hpp, [1207](#)
OSParseosrl.tab.hpp, [1280](#)

MATRIXVARSTART
OSParseosil.tab.hpp, [1132](#), [1143](#), [1153](#)
OSParseosol.tab.hpp, [1207](#), [1218](#), [1228](#)
OSParseosrl.tab.hpp, [1280](#), [1291](#), [1301](#)

MAXEND
OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MAXSTART
OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)

MAXTIMEEND
OSParseosil.tab.hpp, [1141](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)

MAXTIMESTART
OSParseosil.tab.hpp, [1141](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)

MESSAGEEND
OSParseosil.tab.hpp, [1151](#)
OSParseosol.tab.hpp, [1226](#)
OSParseosrl.tab.hpp, [1299](#)

MESSAGESTART
OSParseosil.tab.hpp, [1151](#)
OSParseosol.tab.hpp, [1226](#)
OSParseosrl.tab.hpp, [1299](#)

MINCPUNUMBEREND
OSParseosil.tab.hpp, [1141](#)
OSParseosol.tab.hpp, [1216](#)
OSParseosrl.tab.hpp, [1289](#)

MINCPUNUMBERSTART
OSParseosil.tab.hpp, [1141](#)

- OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINCPUSPEEDEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINCPUSPEEDSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINDISKSPACEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINDISKSPACESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- MINMEMORYEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINMEMORYSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- MINSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- MINUSEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- MINUSSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- MULTATT
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1297](#)
- m_ADTPe
 - OSnLNode, [559](#)
- m_bDeleteExpressionTree
 - MatrixExpression, [296](#)
 - NI, [341](#)
- m_bIndexMapGenerated
 - OSExpressionTree, [403](#)
- m_bWhiteSpace
 - OSiLWriter, [433](#)
 - OSoLWriter, [674](#)
 - OSrLWriter, [830](#)
- m_bWriteBase64
 - OSiLWriter, [433](#)
 - OSoLWriter, [674](#)
 - OSrLWriter, [830](#)
- m_dFunctionValue
 - OSnLNode, [559](#)
- m_iConstraintNumber
 - OSResult, [810](#)
- m_iNumberOfOtherVariableResults
 - OSResult, [810](#)
- m_iObjectiveNumber
 - OSResult, [810](#)
- m_iVariableNumber
 - OSResult, [810](#)
- m_mChildren
 - ExprNode, [137](#)
 - MatrixNode, [306](#)
- m_mMatrixChildren
 - ExprNode, [137](#)
- m_mdDualValues
 - OSResult, [811](#)
- m_mdPrimalValues
 - OSResult, [811](#)
- m_miConStageInfo
 - OSiLParserData, [425](#)
- m_miObjStageInfo
 - OSiLParserData, [425](#)
- m_miVarStageInfo
 - OSiLParserData, [425](#)
- m_osilreader
 - BonminSolver, [49](#)
 - CoinSolver, [57](#)
 - CouenneSolver, [99](#)
 - CsdpSolver, [109](#)
 - IpoptSolver, [237](#)
 - LindoSolver, [259](#)
- m_osolreader
 - BonminSolver, [49](#)
 - CoinSolver, [57](#)
 - CouenneSolver, [99](#)
 - CsdpSolver, [109](#)
 - IpoptSolver, [237](#)
- m_sB64encoded
 - OSiLWriter, [434](#)
 - OSoLWriter, [674](#)
 - OSrLWriter, [830](#)
- m_treeRoot
 - MatrixExpressionTree, [302](#)
 - ScalarExpressionTree, [943](#)
- MAKECOPYATT
 - OSParseosol.tab.hpp, [1177](#)

MATRICESEND
 OSParseosil.tab.hpp, 1108

MATRICESSTART
 OSParseosil.tab.hpp, 1108

MATRIXCONEND
 OSParseosil.tab.hpp, 1112
 OSParseosol.tab.hpp, 1191
 OSParseosrl.tab.hpp, 1264

MATRIXCONSTART
 OSParseosil.tab.hpp, 1112
 OSParseosol.tab.hpp, 1191
 OSParseosrl.tab.hpp, 1264

MATRIXCONSTRAINTSEND
 OSParseosil.tab.hpp, 1112

MATRIXDETERMINANTEND
 OSParseosil.tab.hpp, 1127
 OSParseosol.tab.hpp, 1202
 OSParseosrl.tab.hpp, 1275

MATRIXDIAGONALEND
 OSParseosil.tab.hpp, 1127
 OSParseosol.tab.hpp, 1202
 OSParseosrl.tab.hpp, 1275

MATRIXDIAGONALSTART
 OSParseosil.tab.hpp, 1127
 OSParseosol.tab.hpp, 1202
 OSParseosrl.tab.hpp, 1275

MATRIXDOTTIMESEND
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1202
 OSParseosrl.tab.hpp, 1275

MATRIXDOTTIMESSTART
 OSParseosil.tab.hpp, 1127
 OSParseosol.tab.hpp, 1202
 OSParseosrl.tab.hpp, 1275

MATRIXEND
 OSParseosil.tab.hpp, 1117
 OSParseosol.tab.hpp, 1192
 OSParseosrl.tab.hpp, 1266

MATRIXEXPRESSIONSEND
 OSParseosil.tab.hpp, 1123
 OSParseosol.tab.hpp, 1198
 OSParseosrl.tab.hpp, 1271

MATRIXIDXATT
 OSParseosil.tab.hpp, 1112

MATRIXINVERSEEND
 OSParseosil.tab.hpp, 1130
 OSParseosol.tab.hpp, 1204
 OSParseosrl.tab.hpp, 1277

MATRIXINVERSESTART
 OSParseosil.tab.hpp, 1129
 OSParseosol.tab.hpp, 1204
 OSParseosrl.tab.hpp, 1277

MATRIXMERGEEND
 OSParseosil.tab.hpp, 1128

 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXMERGESTART
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXMINUSEND
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXMINUSSTART
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXNEGATEEND
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXNEGATESTART
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXOBJECTIVESEND
 OSParseosil.tab.hpp, 1111

MATRIXOBJEND
 OSParseosil.tab.hpp, 1112
 OSParseosol.tab.hpp, 1191
 OSParseosrl.tab.hpp, 1264

MATRIXOBJSTART
 OSParseosil.tab.hpp, 1111
 OSParseosol.tab.hpp, 1191
 OSParseosrl.tab.hpp, 1264

MATRIXPLUSEND
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXPLUSSTART
 OSParseosil.tab.hpp, 1128
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXPRODUCTEND
 OSParseosil.tab.hpp, 1129
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXPRODUCTSTART
 OSParseosil.tab.hpp, 1129
 OSParseosol.tab.hpp, 1203
 OSParseosrl.tab.hpp, 1276

MATRIXPROGRAMMINGEND
 OSParseosil.tab.hpp, 1111

MATRIXREFERENCEEND
 OSParseosil.tab.hpp, 1129
 OSParseosol.tab.hpp, 1204
 OSParseosrl.tab.hpp, 1277

MATRIXREFERENCESTART
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1204](#)
 OSParseosrl.tab.hpp, [1277](#)

MATRIXSCALARTIMESEND
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1204](#)
 OSParseosrl.tab.hpp, [1276](#)

MATRIXSTART
 OSParseosil.tab.hpp, [1117](#)
 OSParseosol.tab.hpp, [1192](#)
 OSParseosrl.tab.hpp, [1265](#)

MATRIXSUBMATRIXATEND
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1204](#)
 OSParseosrl.tab.hpp, [1277](#)

MATRIXTERMEND
 OSParseosil.tab.hpp, [1123](#)
 OSParseosol.tab.hpp, [1198](#)
 OSParseosrl.tab.hpp, [1271](#)

MATRIXTERMSTART
 OSParseosil.tab.hpp, [1123](#)
 OSParseosol.tab.hpp, [1198](#)
 OSParseosrl.tab.hpp, [1271](#)

MATRIXTIMESEND
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1203](#)
 OSParseosrl.tab.hpp, [1276](#)

MATRIXTIMESSTART
 OSParseosil.tab.hpp, [1128](#)
 OSParseosol.tab.hpp, [1203](#)
 OSParseosrl.tab.hpp, [1276](#)

MATRIXTOSCALAREND
 OSParseosil.tab.hpp, [1127](#)
 OSParseosol.tab.hpp, [1202](#)
 OSParseosrl.tab.hpp, [1275](#)

MATRIXTOSCALARSTART
 OSParseosil.tab.hpp, [1127](#)
 OSParseosol.tab.hpp, [1202](#)
 OSParseosrl.tab.hpp, [1275](#)

MATRIXTRACEEND
 OSParseosil.tab.hpp, [1127](#)
 OSParseosol.tab.hpp, [1202](#)
 OSParseosrl.tab.hpp, [1275](#)

MATRIXTRACESTART
 OSParseosil.tab.hpp, [1127](#)
 OSParseosol.tab.hpp, [1202](#)
 OSParseosrl.tab.hpp, [1275](#)

MATRIXTRANPOSEEND
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1204](#)
 OSParseosrl.tab.hpp, [1277](#)

MATRIXTRANPOSESTART
 OSParseosil.tab.hpp, [1129](#)
 OSParseosol.tab.hpp, [1204](#)
 OSParseosrl.tab.hpp, [1277](#)

MATRIXVAREND
 OSParseosil.tab.hpp, [1111](#)
 OSParseosol.tab.hpp, [1190](#)
 OSParseosrl.tab.hpp, [1264](#)

MATRIXVARIABLESEND
 OSParseosil.tab.hpp, [1111](#)

MATRIXVARIABLESSTART
 OSParseosil.tab.hpp, [1111](#)

MATRIXVARSTART
 OSParseosil.tab.hpp, [1111](#)
 OSParseosol.tab.hpp, [1190](#)
 OSParseosrl.tab.hpp, [1264](#)

MAXEND
 OSParseosil.tab.hpp, [1126](#)
 OSParseosol.tab.hpp, [1201](#)
 OSParseosrl.tab.hpp, [1274](#)

MAXSTART
 OSParseosil.tab.hpp, [1126](#)
 OSParseosol.tab.hpp, [1201](#)
 OSParseosrl.tab.hpp, [1274](#)

MAXTIMEEND
 OSParseosol.tab.hpp, [1184](#)

MAXTIMESTART
 OSParseosol.tab.hpp, [1183](#)

MESSAGEEND
 OSParseosrl.tab.hpp, [1259](#)

MESSAGESTART
 OSParseosrl.tab.hpp, [1258](#)

MINCPUNUMBEREND
 OSParseosol.tab.hpp, [1183](#)

MINCPUNUMBERSTART
 OSParseosol.tab.hpp, [1183](#)

MINCPUSPEEDEND
 OSParseosol.tab.hpp, [1183](#)

MINCPUSPEEDSTART
 OSParseosol.tab.hpp, [1183](#)

MINDISKSPACEEND
 OSParseosol.tab.hpp, [1183](#)

MINDISKSPACESTART
 OSParseosol.tab.hpp, [1183](#)

MINEND
 OSParseosil.tab.hpp, [1126](#)
 OSParseosol.tab.hpp, [1201](#)
 OSParseosrl.tab.hpp, [1274](#)

MINMEMORYEND
 OSParseosol.tab.hpp, [1183](#)

MINMEMORYSTART
 OSParseosol.tab.hpp, [1183](#)

MINSTART
 OSParseosil.tab.hpp, [1126](#)
 OSParseosol.tab.hpp, [1201](#)
 OSParseosrl.tab.hpp, [1274](#)

- MINUSEND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1271](#)
- MINUSSTART
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1271](#)
- MULTATT
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1252](#)
- makeCopy
 - OSoLParserData, [670](#)
 - PathPair, [905](#)
- makeStringFromInt
 - OSStringUtil.h, [1354](#)
- mapVarIdx
 - OSExpressionTree, [403](#)
- MathUtil, [271](#)
 - ~MathUtil, [272](#)
 - convertLinearConstraintCoefficientMatrixToThe-
OtherMajor, [272](#)
 - format_os_dtoa, [272](#)
 - MathUtil, [272](#)
 - MathUtil, [272](#)
- Matrices, [273](#)
 - ~Matrices, [274](#)
 - deepCopyFrom, [274](#)
 - IsEqual, [274](#)
 - Matrices, [274](#)
 - matrix, [275](#)
 - numberOfMatrices, [275](#)
 - setRandom, [274](#)
- matrices
 - InstanceData, [215](#)
- matrix
 - Matrices, [275](#)
 - OSgLParseData, [411](#)
- MatrixBlock, [275](#)
 - ~MatrixBlock, [277](#)
 - alignsOnBlockBoundary, [277](#)
 - blockColIdx, [278](#)
 - blockRowIdx, [278](#)
 - cloneMatrixNode, [277](#)
 - deepCopyFrom, [278](#)
 - getMatrixNodeInXML, [277](#)
 - getMatrixType, [277](#)
 - getNodeName, [277](#)
 - getNodeType, [277](#)
 - IsEqual, [278](#)
 - MatrixBlock, [277](#)
 - MatrixBlock, [277](#)
 - setRandom, [278](#)
- matrixBlockNumberOfCols
 - OSgLParseData, [415](#)
- matrixBlockNumberOfRows
 - OSgLParseData, [415](#)
- MatrixBlocks, [279](#)
 - ~MatrixBlocks, [281](#)
 - alignsOnBlockBoundary, [282](#)
 - block, [283](#)
 - cloneMatrixNode, [282](#)
 - colOffsets, [283](#)
 - deepCopyFrom, [282](#)
 - getMatrixNodeInXML, [281](#)
 - getMatrixType, [281](#)
 - getNodeName, [281](#)
 - getNodeType, [281](#)
 - IsEqual, [282](#)
 - MatrixBlocks, [281](#)
 - MatrixBlocks, [281](#)
 - numberOfBlocks, [283](#)
 - rowOffsets, [283](#)
 - setRandom, [282](#)
- MatrixCon, [283](#)
 - ~MatrixCon, [285](#)
 - conReferenceMatrixIdx, [285](#)
 - IsEqual, [285](#)
 - lbConelIdx, [286](#)
 - lbMatrixIdx, [285](#)
 - MatrixCon, [285](#)
 - MatrixCon, [285](#)
 - name, [286](#)
 - numberOfColumns, [285](#)
 - numberOfRows, [285](#)
 - templateMatrixIdx, [285](#)
 - ubConelIdx, [286](#)
 - ubMatrixIdx, [286](#)
- matrixCon
 - MatrixConstraints, [288](#)
- MatrixConstraints, [286](#)
 - ~MatrixConstraints, [288](#)
 - IsEqual, [288](#)
 - matrixCon, [288](#)
 - MatrixConstraints, [288](#)
 - MatrixConstraints, [288](#)
 - numberOfMatrixCon, [288](#)
- matrixConstraints
 - MatrixProgramming, [314](#)
- MatrixConstructor, [288](#)
 - ~MatrixConstructor, [290](#)
 - MatrixConstructor, [290](#)
 - MatrixConstructor, [290](#)
- matrixCounter
 - OSgLParseData, [411](#)
- MatrixElementValues, [292](#)
 - ~MatrixElementValues, [293](#)

- MatrixElementValues, 293
- MatrixElementValues, 293
- numberOfEl, 293
- MatrixElements, 290
 - ~MatrixElements, 292
 - getRowMajor, 292
 - indexes, 292
 - IsEqual, 292
 - MatrixElements, 292
 - MatrixElements, 292
 - numberOfValues, 292
 - rowMajor, 292
 - start, 292
- MatrixExpression, 294
 - ~MatrixExpression, 295
 - idx, 295
 - IsEqual, 295
 - m_bDeleteExpressionTree, 296
 - MatrixExpression, 295
 - matrixExpressionTree, 296
 - MatrixExpression, 295
 - shape, 296
- MatrixExpressionTree, 298
 - ~MatrixExpressionTree, 301
 - getPostfixFromExpressionTree, 301
 - getPrefixFromExpressionTree, 301
 - IsEqual, 301
 - m_treeRoot, 302
 - MatrixExpressionTree, 301
 - MatrixExpressionTree, 301
- matrixExpressionTree
 - MatrixExpression, 296
- MatrixExpressions, 296
 - ~MatrixExpressions, 298
 - expr, 298
 - IsEqual, 298
 - MatrixExpressions, 298
 - MatrixExpressions, 298
 - numberOfExpr, 298
- matrixExpressions
 - MatrixProgramming, 314
- matrixHasBase
 - MatrixType, 320
 - OSInstance, 456
- matrixHasBlocks
 - MatrixType, 320
 - OSInstance, 456
- matrixHasElements
 - MatrixType, 320
 - OSInstance, 456
- matrixHasTransformations
 - MatrixType, 320
 - OSInstance, 456
- matrixIdx
 - OSILParserData, 429
- matrixIdxPresent
 - OSILParserData, 428
- MatrixNode, 302
 - ~MatrixNode, 303
 - alignsOnBlockBoundary, 305
 - cloneMatrixNode, 305
 - deepCopyFrom, 306
 - getMatrixNodeInXML, 304
 - getMatrixType, 304
 - getNodeName, 304
 - getNodeType, 304
 - getPostfixFromNodeTree, 305
 - getPrefixFromNodeTree, 304
 - inumberOfChildren, 306
 - IsEqual, 305
 - m_mChildren, 306
 - MatrixNode, 303
 - matrixType, 306
 - MatrixNode, 303
 - nType, 306
 - postOrderMatrixNodeTraversal, 305
 - preOrderMatrixNodeTraversal, 304
 - setRandom, 306
- MatrixObj, 307
 - ~MatrixObj, 308
 - constantMatrixIdx, 309
 - IsEqual, 308
 - MatrixObj, 308
 - MatrixObj, 308
 - name, 309
 - numberOfColumns, 308
 - numberOfRows, 308
 - objReferenceMatrixIdx, 308
 - orderConelIdx, 309
 - templateMatrixIdx, 308
- matrixObj
 - MatrixObjectives, 311
- MatrixObjectives, 309
 - ~MatrixObjectives, 311
 - IsEqual, 311
 - matrixObj, 311
 - MatrixObjectives, 311
 - MatrixObjectives, 311
 - numberOfMatrixObj, 311
- matrixObjectives
 - MatrixProgramming, 314
- matrixProductVec
 - OSnLParserData, 651
- MatrixProgramming, 311
 - ~MatrixProgramming, 313
 - deepCopyFrom, 313
 - IsEqual, 313
 - matrixConstraints, 314

- matrixExpressions, 314
- matrixObjectives, 314
- MatrixProgramming, 313
- matrixVariables, 313
- MatrixProgramming, 313
- setRandom, 313
- matrixProgramming
 - InstanceData, 216
- matrixSumVec
 - OSnLParserData, 651
- matrixTermInObj
 - OSiLParserData, 430
- MatrixTransformation, 314
 - ~MatrixTransformation, 316
 - alignsOnBlockBoundary, 316
 - cloneMatrixNode, 317
 - deepCopyFrom, 317
 - getMatrixNodeInXML, 316
 - getMatrixType, 316
 - getNodeName, 316
 - getNodeType, 316
 - isEqual, 317
 - MatrixTransformation, 316
 - MatrixTransformation, 316
 - setRandom, 317
 - shape, 318
 - transformation, 318
- MatrixType, 318
 - ~MatrixType, 320
 - alignsOnBlockBoundary, 320
 - deepCopyFrom, 322
 - extractBlock, 321
 - getMatrixBlockInColumnMajorForm, 321
 - getMatrixInColumnMajorForm, 321
 - getMatrixInRowMajorForm, 321
 - getNumberOfBlocksConstructors, 321
 - getNumberOfElementConstructors, 321
 - getNumberOfTransformationConstructors, 321
 - haveExpandedForm, 323
 - isEqual, 322
 - matrixHasBase, 320
 - matrixHasBlocks, 320
 - matrixHasElements, 320
 - matrixHasTransformations, 320
 - MatrixType, 320
 - MatrixType, 320
 - numberOfColumns, 323
 - numberOfRows, 323
 - processBlocks, 321
 - setRandom, 322
 - symmetry, 322
 - type, 323
- matrixType
 - MatrixNode, 306
- MatrixVar, 323
 - ~MatrixVar, 325
 - isEqual, 325
 - lbConeldx, 326
 - lbMatrixIdx, 325
 - MatrixVar, 325
 - MatrixVar, 325
 - name, 326
 - numberOfColumns, 325
 - numberOfRows, 325
 - templateMatrixIdx, 325
 - ubConeldx, 326
 - ubMatrixIdx, 326
 - varReferenceMatrixIdx, 325
 - varType, 326
- matrixVar
 - MatrixVariables, 328
- MatrixVariables, 326
 - ~MatrixVariables, 328
 - isEqual, 328
 - matrixVar, 328
 - MatrixVariables, 328
 - MatrixVariables, 328
 - numberOfMatrixVar, 328
- matrixVariables
 - MatrixProgramming, 313
- matrixidxattON
 - OSnLParserData, 651
- matrixreftypeattON
 - OSnLParserData, 651
- maxOrMin
 - Objective, 353
- MaxTime, 328
 - ~MaxTime, 330
 - isEqual, 330
 - MaxTime, 330
 - MaxTime, 330
 - unit, 330
 - value, 330
- maxTime
 - JobOption, 243
- maxTimePresent
 - OSoLParserData, 660
- maxTimeUnit
 - OSoLParserData, 660
- maxTimeUnitPresent
 - OSoLParserData, 660
- maxTimeValue
 - OSoLParserData, 660
- maxVec
 - OSnLParserData, 650
- mergeMatrixType
 - OSParameters.h, 1348
- message

- GeneralResult, [156](#)
- OptimizationSolution, [382](#)
- MinCPUNumber, [330](#)
 - ~MinCPUNumber, [332](#)
 - description, [332](#)
 - IsEqual, [332](#)
 - MinCPUNumber, [332](#)
 - MinCPUNumber, [332](#)
 - value, [332](#)
- minCPUNumber
 - SystemOption, [984](#)
- minCPUNumberPresent
 - OSoLParserData, [659](#)
- MinCPUSpeed, [332](#)
 - ~MinCPUSpeed, [334](#)
 - description, [334](#)
 - IsEqual, [334](#)
 - MinCPUSpeed, [334](#)
 - MinCPUSpeed, [334](#)
 - unit, [334](#)
 - value, [334](#)
- minCPUSpeed
 - SystemOption, [984](#)
- minCPUSpeedPresent
 - OSoLParserData, [659](#)
- minCPUSpeedUnitPresent
 - OSoLParserData, [659](#)
- MinDiskSpace, [335](#)
 - ~MinDiskSpace, [336](#)
 - description, [336](#)
 - IsEqual, [336](#)
 - MinDiskSpace, [336](#)
 - MinDiskSpace, [336](#)
 - unit, [336](#)
 - value, [336](#)
- minDiskSpace
 - SystemOption, [984](#)
- minDiskSpacePresent
 - OSoLParserData, [659](#)
- minDiskSpaceUnitPresent
 - OSoLParserData, [659](#)
- minMemoryPresent
 - OSoLParserData, [659](#)
- MinMemorySize, [337](#)
 - ~MinMemorySize, [338](#)
 - description, [338](#)
 - IsEqual, [338](#)
 - MinMemorySize, [338](#)
 - MinMemorySize, [338](#)
 - unit, [338](#)
 - value, [339](#)
- minMemorySize
 - SystemOption, [984](#)
- minMemoryUnitPresent
 - OSoLParserData, [659](#)
- minVec
 - OSnLParserData, [651](#)
- mps
 - OSCommandLine, [397](#)
 - osOptionsStruc, [718](#)
- mpsFile
 - OSCommandLine, [397](#)
 - osOptionsStruc, [718](#)
- mtxBlkVec
 - OSgLParserData, [411](#)
- mtxBlocksVec
 - OSgLParserData, [411](#)
- mtxConstructorVec
 - OSgLParserData, [411](#)
- mult
 - OSrLParserData, [820](#)
- multPresent
 - OSrLParserData, [821](#)
- NAMEATT
 - OSParseosil.tab.hpp, [1134](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1297](#)
- NEGATEEND
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- NEGATESTART
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- NEGATIVEPATTERNATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- NLEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- NLSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- NONLINEAREXPRESSIONSEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- NONLINEAREXPRESSIONSSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- NONNEGATIVECONEEND
 - OSParseosil.tab.hpp, [1131](#)

- OSParseosol.tab.hpp, [1206](#)
- OSParseosrl.tab.hpp, [1279](#)
- NONNEGATIVECONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NONPOSITIVECONEEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NONPOSITIVECONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NONZEROSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- NONZEROSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- NORMSCALEFACTORATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NUMBEREND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- NUMBEROFBLOCKSATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- NUMBEROFCOLUMNSATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- NUMBEROFCONATT
 - OSParseosil.tab.hpp, [1139](#), [1148](#)
 - OSParseosol.tab.hpp, [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1287](#), [1296](#)
- NUMBEROFCONESATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NUMBEROFCONSTRAINTSATT
 - OSParseosil.tab.hpp, [1133](#), [1139](#), [1148](#)
 - OSParseosol.tab.hpp, [1208](#), [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1281](#), [1287](#), [1296](#)
- NUMBEROFELATT
 - OSParseosil.tab.hpp, [1134](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1296](#)
- NUMBEROFENUMERATIONSATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1286](#), [1296](#)
- NUMBEROFEXPR
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- NUMBEROFIDXATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFITEMSATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1287](#), [1296](#)
- NUMBEROFJOBIDSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- NUMBEROFMATRICESATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NUMBEROFMATRIXCONATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- NUMBEROFMATRIXOBJATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- NUMBEROFMATRIXTERMSATT
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- NUMBEROFMATRIXVARATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- NUMBEROFNONLINEAREXPRESSIONS
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1293](#), [1303](#)
- NUMBEROFOBJATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1287](#), [1296](#)
- NUMBEROFOBJECTIVESATT
 - OSParseosil.tab.hpp, [1133](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1208](#), [1214](#), [1223](#)
 - OSParseosrl.tab.hpp, [1281](#), [1287](#), [1296](#)
- NUMBEROFOTHERCONSTRAINTOPTIONSATT
 - OSParseosil.tab.hpp, [1139](#)

- OSParseosol.tab.hpp, [1214](#)
- OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFOTHERCONSTRAINTRESULTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFOTHEROBJECTIVEOPTIONSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFOTHEROBJECTIVERESULTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFOTHEROPTIONSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- NUMBEROFOTHERRESULTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFOTHERSOLUTIONRESULTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFOTHERVARIABLEOPTIONSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFOTHERVARIABLELERESULTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFPATHPAIRSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1286](#)
- NUMBEROFPATHSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1286](#)
- NUMBEROFPROCESSESATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFQTERMSATT
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- NUMBEROFFROWSATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- NUMBEROFSOLUTIONSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFSOLVEROPTIONSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFSOLVEROUTPUTSATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFSOSATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- NUMBEROFSTAGESATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1280](#)
- NUMBEROFSUBSTATUSESATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1296](#)
- NUMBEROFTIMESATT
 - OSParseosil.tab.hpp, [1149](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1297](#)
- NUMBEROFVALUESATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- NUMBEROFVARATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- NUMBEROFVARIABLESATT
 - OSParseosil.tab.hpp, [1133](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1208](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1281](#), [1287](#), [1297](#)
- NUMBEROFVARIDXATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1297](#)
- NUMBERSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- NAMEATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1252](#)
- nConPresent
 - OSrLParserData, [823](#)

- NEGATEEND
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1272](#)
- NEGATESTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1272](#)
- NEGATIVEPATTERNATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- NLEND
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1270](#)
- NLSTART
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1270](#)
- NONNEGATIVECONEEND
 - OSParseosil.tab.hpp, [1108](#)
- NONNEGATIVECONESTART
 - OSParseosil.tab.hpp, [1108](#)
- NONPOSITIVECONEEND
 - OSParseosil.tab.hpp, [1108](#)
- NONPOSITIVECONESTART
 - OSParseosil.tab.hpp, [1108](#)
- NONZEROSEND
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- NONZEROSSTART
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- NORMSCALEFACTORATT
 - OSParseosil.tab.hpp, [1110](#)
- nObjPresent
 - OSrLParserData, [823](#)
- nType
 - MatrixNode, [306](#)
- NUMBEREND
 - OSParseosil.tab.hpp, [1127](#)
 - OSParseosol.tab.hpp, [1202](#)
 - OSParseosrl.tab.hpp, [1274](#)
- NUMBEROFBLOCKSATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1267](#)
- NUMBEROFCOLUMNSATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- NUMBEROFCONATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1250](#)
- NUMBEROFCONESATT
 - OSParseosil.tab.hpp, [1108](#)
- NUMBEROFELATT
 - OSParseosil.tab.hpp, [1116](#)
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1250](#)
- NUMBEROFEXPR
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- NUMBEROFIDXATT
 - OSParseosrl.tab.hpp, [1250](#)
- NUMBEROFITEMSATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1250](#)
- NUMBEROFJOBIDSATT
 - OSParseosol.tab.hpp, [1176](#)
- NUMBEROFMATRICESATT
 - OSParseosil.tab.hpp, [1108](#)
- NUMBEROFMATRIXCONATT
 - OSParseosil.tab.hpp, [1112](#)
- NUMBEROFMATRIXOBJATT
 - OSParseosil.tab.hpp, [1111](#)
- NUMBEROFMATRIXVARATT
 - OSParseosil.tab.hpp, [1111](#)
- NUMBEROFOBJATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBEROFPATHPAIRSATT
 - OSParseosol.tab.hpp, [1176](#)
- NUMBEROFPATHSATT
 - OSParseosol.tab.hpp, [1176](#)
- NUMBEROFPROCESSESATT
 - OSParseosol.tab.hpp, [1177](#)
- NUMBEROFQTERMSATT
 - OSParseosil.tab.hpp, [1107](#)
- NUMBEROFROWSATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1267](#)
- NUMBEROFSOLUTIONSATT
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBEROFSOSATT
 - OSParseosol.tab.hpp, [1177](#)
- NUMBEROFSTAGESATT
 - OSParseosil.tab.hpp, [1113](#)
- NUMBEROFTIMESATT
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBEROFVALUESATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)

- OSParseosrl.tab.hpp, [1267](#)
- NUMBEROFVARATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBEROFVARIABLESATT
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1177](#)
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBEROFVARIDXATT
 - OSParseosil.tab.hpp, [1119](#)
 - OSParseosol.tab.hpp, [1194](#)
 - OSParseosrl.tab.hpp, [1251](#)
- NUMBERSTART
 - OSParseosil.tab.hpp, [1127](#)
 - OSParseosol.tab.hpp, [1202](#)
 - OSParseosrl.tab.hpp, [1274](#)
- nVarPresent
 - OSrLPParserData, [823](#)
- namArray
 - OSoLPParserData, [670](#)
- Name
 - OSOutputChannel, [726](#)
- name
 - BranchingWeight, [52](#)
 - Cone, [64](#)
 - Constraint, [82](#)
 - DualVarValue, [126](#)
 - GeneralFileHeader, [142](#)
 - GeneralSubstatus, [164](#)
 - InitConValue, [175](#)
 - InitDualVarValue, [182](#)
 - InitObjBound, [188](#)
 - InitObjValue, [199](#)
 - InitVarValue, [210](#)
 - InitVarValueString, [213](#)
 - MatrixCon, [286](#)
 - MatrixObj, [309](#)
 - MatrixVar, [326](#)
 - Objective, [353](#)
 - ObjValue, [373](#)
 - OS_AMPL_SUFFIX, [391](#)
 - OSgLPParserData, [412](#)
 - OSiLPParserData, [427](#)
 - OSMatrix, [494](#)
 - OSrLPParserData, [820](#)
 - OtherConOption, [842](#)
 - OtherConResult, [845](#)
 - OtherConstraintOption, [849](#)
 - OtherConstraintResult, [853](#)
 - OtherObjectiveOption, [857](#)
 - OtherObjectiveResult, [861](#)
 - OtherObjOption, [864](#)
 - OtherObjResult, [867](#)
 - OtherOption, [869](#)
 - OtherResult, [878](#)
 - OtherSolutionResult, [883](#)
 - OtherVariableOption, [891](#)
 - OtherVariableResult, [895](#)
 - OtherVariableResultStruct, [897](#)
 - OtherVarOption, [900](#)
 - OtherVarResult, [903](#)
 - SolverOption, [956](#)
 - SolverOutput, [963](#)
 - TimeDomainStage, [991](#)
 - Variable, [1010](#)
 - VarValue, [1032](#)
 - VarValueString, [1034](#)
- nameAttribute
 - OSnLPParserData, [647](#)
 - OSoLPParserData, [668](#)
 - OSrLPParserData, [822](#)
- nameAttributePresent
 - OSnLPParserData, [647](#)
 - OSoLPParserData, [668](#)
 - OSrLPParserData, [821](#)
- namePresent
 - OSgLPParserData, [412](#)
 - OSiLPParserData, [427](#)
- nconcovered
 - OSiLPParserData, [425](#)
- NI, [339](#)
 - ~NI, [341](#)
 - idx, [341](#)
 - IsEqual, [341](#)
 - m_bDeleteExpressionTree, [341](#)
 - NI, [341](#)
 - osExpressionTree, [342](#)
 - shape, [341](#)
- nl
 - NonlinearExpressions, [344](#)
 - OSCommandLine, [398](#)
 - osOptionsStruc, [718](#)
- nlFile
 - OSCommandLine, [398](#)
 - osOptionsStruc, [718](#)
- nlMNodeMatrixCon
 - OSnLPParserData, [649](#)
- nlMNodeMatrixObj
 - OSnLPParserData, [649](#)
- nlMNodeMatrixRef
 - OSnLPParserData, [649](#)
- nlMNodeMatrixVar
 - OSnLPParserData, [649](#)
- nlNodeNumberPoint
 - OSnLPParserData, [649](#)
- nlNodePoint
 - OSnLPParserData, [648](#)
- nlNodeVariablePoint

- OSnLParserData, 649
- nINodeVec
 - OSnLParserData, 650
- nInodenumber
 - OSnLParserData, 649
- nlp
 - IpoptSolver, 237
- NonlinearExpressions, 342
 - ~NonlinearExpressions, 344
 - IsEqual, 344
 - nl, 344
 - NonlinearExpressions, 344
 - NonlinearExpressions, 344
 - numberOfNonlinearExpressions, 344
- nonlinearExpressions
 - InstanceData, 215
- NonnegativeCone, 345
 - ~NonnegativeCone, 346
 - deepCopyFrom, 347
 - getConeInXML, 346
 - getConeName, 346
 - IsEqual, 347
 - NonnegativeCone, 346
 - NonnegativeCone, 346
 - setRandom, 347
- NonpositiveCone, 347
 - ~NonpositiveCone, 349
 - deepCopyFrom, 350
 - getConeInXML, 349
 - getConeName, 349
 - IsEqual, 349
 - NonpositiveCone, 349
 - NonpositiveCone, 349
 - setRandom, 350
- normScaleFactor
 - OSiLParserData, 427
 - QuadraticCone, 928
 - RotatedQuadraticCone, 935
- normScaleFactorPresent
 - OSiLParserData, 427
- numCon
 - OSMatlab, 487
- numQTerms
 - OSMatlab, 487
- numVar
 - OSMatlab, 487
- number
 - SparsenIntVector, 973
 - SparseVector, 979
- numberAttributePresent
 - OSrLParserData, 821
- numberOf
 - OSiLParserData, 427
 - OSnLParserData, 648
 - OSoLParserData, 670
 - OSrLParserData, 819
- numberOfBasVar
 - OSoLParserData, 664
- numberOfBlocks
 - MatrixBlocks, 283
 - OSgLPParserData, 412
- numberOfBlocksPresent
 - OSgLPParserData, 414
- numberOfColumns
 - Cone, 64
 - DualCone, 121
 - IntersectionCone, 225
 - MatrixCon, 285
 - MatrixObj, 308
 - MatrixType, 323
 - MatrixVar, 325
 - OSgLPParserData, 412
 - OSiLParserData, 426
 - PolarCone, 912
 - PolyhedralCone, 916
 - ProductCone, 922
 - QuadraticCone, 928
 - RotatedQuadraticCone, 935
 - SemidefiniteCone, 946
- numberOfColumnsPresent
 - OSgLPParserData, 414
 - OSiLParserData, 426
- numberOfCon
 - DualVariableValues, 124
 - InitConstraintValues, 173
 - InitDualVariableValues, 179
 - OSoLParserData, 665
 - OSrLParserData, 819
 - OtherConstraintOption, 849
 - OtherConstraintResult, 853
- numberOfConAttributePresent
 - OSoLParserData, 664
 - OSrLParserData, 821
- numberOfConIdxAttributePresent
 - OSrLParserData, 822
- numberOfCones
 - Cones, 67
 - OSiLParserData, 426
- numberOfConesPresent
 - OSiLParserData, 426
- numberOfConstraints
 - Constraints, 87
 - OptimizationOption, 376
 - OptimizationResult, 379
 - OSoLParserData, 663
 - OSrLParserData, 818
 - TimeDomainStageConstraints, 993
- numberOfConstraintsPresent

- OSoLParserData, 663
- numberOfDependencies
 - OSoLParserData, 660
- numberOfDirectoriesToDelete
 - OSoLParserData, 662
- numberOfDirectoriesToMake
 - OSoLParserData, 661
- numberOfDuals
 - OSoLParserData, 666
- numberOfEI
 - DoubleVector, 118
 - IntVector, 229
 - MatrixElementValues, 293
 - OSgLPParserData, 415
 - OSiLParserData, 426
- numberOfEIPresent
 - OSgLPParserData, 415
- numberOfEnumerations
 - OSoLParserData, 666
 - OSrLPParserData, 819
 - OtherConstraintOption, 849
 - OtherConstraintResult, 853
 - OtherObjectiveOption, 857
 - OtherObjectiveResult, 861
 - OtherVariableOption, 891
 - OtherVariableResult, 895
- numberOfEnumerationsAttributePresent
 - OSoLParserData, 664
 - OSrLPParserData, 822
- numberOfExpr
 - MatrixExpressions, 298
- numberOfFilesToDelete
 - OSoLParserData, 662
- numberOfFilesToMake
 - OSoLParserData, 661
- numberOfIdx
 - OSrLPParserData, 819
- numberOfInputDirectoriesToMove
 - OSoLParserData, 661
- numberOfInputFilesToMove
 - OSoLParserData, 662
- numberOfIntWt
 - OSoLParserData, 664
- numberOfItems
 - OSoLParserData, 667
 - OSrLPParserData, 826
 - OtherSolutionResult, 883
 - SolverOption, 956
 - SolverOutput, 963
- numberOfItemsPresent
 - OSoLParserData, 667
 - OSrLPParserData, 826
- numberOfJobIDs
 - JobDependencies, 240
- numberOfMatrices
 - Matrices, 275
 - OSgLPParserData, 411
 - OSiLParserData, 426
- numberOfMatricesPresent
 - OSiLParserData, 426
- numberOfMatrixCon
 - MatrixConstraints, 288
 - OSiLParserData, 428
- numberOfMatrixExpr
 - OSiLParserData, 428
- numberOfMatrixObj
 - MatrixObjectives, 311
 - OSiLParserData, 428
- numberOfMatrixTerms
 - OSiLParserData, 428
- numberOfMatrixTermsPresent
 - OSiLParserData, 428
- numberOfMatrixVar
 - MatrixVariables, 328
 - OSiLParserData, 428
- numberOfNonlinearExpressions
 - NonlinearExpressions, 344
- numberOfObj
 - InitObjectiveBounds, 192
 - InitObjectiveValues, 196
 - ObjectiveValues, 364
 - OSoLParserData, 665
 - OSrLPParserData, 819
 - OtherObjectiveOption, 857
 - OtherObjectiveResult, 861
- numberOfObjAttributePresent
 - OSoLParserData, 664
 - OSrLPParserData, 821
- numberOfObjBounds
 - OSoLParserData, 665
- numberOfObjCoef
 - Objective, 353
- numberOfObjIdxAttributePresent
 - OSrLPParserData, 821
- numberOfObjValues
 - OSoLParserData, 665
- numberOfObjectives
 - Objectives, 359
 - OptimizationOption, 376
 - OptimizationResult, 379
 - OSoLParserData, 663
 - OSrLPParserData, 818
 - TimeDomainStageObjectives, 996
- numberOfObjectivesPresent
 - OSoLParserData, 663
- numberOfOtherConstraintOptions
 - ConstraintOption, 85
 - OSoLParserData, 665

- numberOfOtherConstraintResults
 - ConstraintSolution, 90
 - OSrLParserData, 820
- numberOfOtherGeneralOptions
 - OSoLParserData, 659
- numberOfOtherIndexes
 - Cone, 64
 - DualCone, 121
 - IntersectionCone, 225
 - PolarCone, 912
 - PolyhedralCone, 916
 - ProductCone, 922
 - QuadraticCone, 928
 - RotatedQuadraticCone, 935
 - SemidefiniteCone, 947
- numberOfOtherJobOptions
 - OSoLParserData, 662
- numberOfOtherObjectiveOptions
 - ObjectiveOption, 356
 - OSoLParserData, 665
- numberOfOtherObjectiveResults
 - ObjectiveSolution, 362
 - OSrLParserData, 820
- numberOfOtherOptions
 - OtherOptions, 875
- numberOfOtherResults
 - OtherResults, 880
- numberOfOtherServiceOptions
 - OSoLParserData, 660
- numberOfOtherSolutionResults
 - OtherSolutionResults, 886
- numberOfOtherSystemOptions
 - OSoLParserData, 659
- numberOfOtherVariableOptions
 - OSoLParserData, 664
 - VariableOption, 1012
- numberOfOtherVariableResults
 - OSrLParserData, 820
 - VariableSolution, 1017
- numberOfOutputDirectoriesToMove
 - OSoLParserData, 662
- numberOfOutputFilesToMove
 - OSoLParserData, 662
- numberOfPathPairs
 - OSoLParserData, 662
 - PathPairs, 909
- numberOfPaths
 - DirectoriesAndFiles, 117
- numberOfProcesses
 - Processes, 919
- numberOfProcessesToKill
 - OSoLParserData, 662
- numberOfQuadraticTerms
 - QuadraticCoefficients, 924
- numberOfRequiredDirectories
 - OSoLParserData, 660
- numberOfRequiredFiles
 - OSoLParserData, 661
- numberOfRows
 - Cone, 64
 - DualCone, 121
 - IntersectionCone, 225
 - MatrixCon, 285
 - MatrixObj, 308
 - MatrixType, 323
 - MatrixVar, 325
 - OSgLParserData, 412
 - OSiLParserData, 426
 - PolarCone, 912
 - PolyhedralCone, 915
 - ProductCone, 922
 - QuadraticCone, 928
 - RotatedQuadraticCone, 934
 - SemidefiniteCone, 946
- numberOfRowsPresent
 - OSgLParserData, 414
 - OSiLParserData, 426
- numberOfSOS
 - OSoLParserData, 665
 - SOSVariableBranchingWeights, 966
- numberOfSOSVar
 - OSoLParserData, 665
- numberOfSolutions
 - OptimizationResult, 379
 - OSrLParserData, 818
- numberOfSolverOptions
 - OSoLParserData, 666
 - SolverOptions, 960
- numberOfSolverOutputs
 - OtherSolverOutput, 888
- numberOfStages
 - TimeDomainStages, 997
- numberOfSubstatuses
 - GeneralStatus, 162
 - OptimizationSolutionStatus, 385
- numberOfTimes
 - OSrLParserData, 818
 - TimingInformation, 1008
- numberOfValues
 - LinearConstraintCoefficients, 260
 - MatrixElements, 292
 - OSgLParserData, 415
- numberOfValuesPresent
 - OSgLParserData, 414
- numberOfVar
 - InitialBasisStatus, 185
 - InitVariableValues, 203
 - InitVariableValuesString, 207

- IntegerVariableBranchingWeights, 222
- OSoLParserData, 664
- OSrLParserData, 819
- OtherVariableOption, 891
- OtherVariableResult, 895
- OtherVariableResultStruct, 897
- SOSWeights, 969
- VariableValues, 1020
- VariableValuesString, 1023
- numberOfVarAttributePresent
 - OSoLParserData, 664
 - OSrLParserData, 821
- numberOfVarIdx
 - LinearMatrixElement, 263
 - OSgLParserData, 415
 - OSrLParserData, 819
- numberOfVarIdxAttributePresent
 - OSrLParserData, 821
- numberOfVarIdxPresent
 - OSgLParserData, 415
- numberOfVarStr
 - OSoLParserData, 664
- numberOfVariables
 - OptimizationOption, 376
 - OptimizationResult, 379
 - OSoLParserData, 663
 - OSrLParserData, 818
 - TimeDomainStageVariables, 999
 - Variables, 1015
- numberOfVariablesPresent
 - OSoLParserData, 663
- numberidattON
 - OSnLParserData, 650
- numbertypeattON
 - OSnLParserData, 649
- numbervalueattON
 - OSnLParserData, 650
- numkount
 - OSnL2OS, 504
- nvarcovered
 - OSiLParserData, 425
- OBJECTIVESEND
 - OSParseosil.tab.hpp, 1133, 1143, 1151
 - OSParseosol.tab.hpp, 1208, 1217, 1226
 - OSParseosrl.tab.hpp, 1281, 1290, 1299
- OBJECTIVESSTART
 - OSParseosil.tab.hpp, 1133, 1143, 1151
 - OSParseosol.tab.hpp, 1208, 1217, 1226
 - OSParseosrl.tab.hpp, 1281, 1290, 1299
- OBJEND
 - OSParseosil.tab.hpp, 1133, 1143, 1151
 - OSParseosol.tab.hpp, 1208, 1218, 1226
 - OSParseosrl.tab.hpp, 1281, 1290, 1299
- OBJREFERENCEELEMENTSEND
 - OSParseosil.tab.hpp, 1136, 1145, 1155
 - OSParseosol.tab.hpp, 1210, 1220, 1230
 - OSParseosrl.tab.hpp, 1283, 1293, 1302
- OBJREFERENCEELEMENTSSTART
 - OSParseosil.tab.hpp, 1136, 1145, 1155
 - OSParseosol.tab.hpp, 1210, 1220, 1230
 - OSParseosrl.tab.hpp, 1283, 1293, 1302
- OBJREFERENCEMATRIXIDXATT
 - OSParseosil.tab.hpp, 1133
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- OBJSTART
 - OSParseosil.tab.hpp, 1133, 1143, 1151
 - OSParseosol.tab.hpp, 1208, 1218, 1226
 - OSParseosrl.tab.hpp, 1281, 1290, 1299
- OBJTYPEATT
 - OSParseosil.tab.hpp, 1139, 1149
 - OSParseosol.tab.hpp, 1214, 1224
 - OSParseosrl.tab.hpp, 1287, 1297
- OPTIMIZATIONEND
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1298
- OPTIMIZATIONSOLUTIONSTATUSSEND
 - OSParseosil.tab.hpp, 1151
 - OSParseosol.tab.hpp, 1226
 - OSParseosrl.tab.hpp, 1299
- OPTIMIZATIONSOLUTIONSTATUSSTART
 - OSParseosil.tab.hpp, 1151
 - OSParseosol.tab.hpp, 1226
 - OSParseosrl.tab.hpp, 1299
- OPTIMIZATIONSOLUTIONSUBSTATUSSEND
 - OSParseosil.tab.hpp, 1151
 - OSParseosol.tab.hpp, 1226
 - OSParseosrl.tab.hpp, 1299
- OPTIMIZATIONSOLUTIONSUBSTATUSSTART
 - OSParseosil.tab.hpp, 1151
 - OSParseosol.tab.hpp, 1226
 - OSParseosrl.tab.hpp, 1299
- OPTIMIZATIONSTART
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1298
- ORDERCONEIDXATT
 - OSParseosil.tab.hpp, 1133
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- ORTHANTCONEEND
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- ORTHANTCONESTART
 - OSParseosil.tab.hpp, 1131

- OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- OS_AMPL_SUFFIX_DIRECTION_both
 - OSnl2OS.h, [1082](#)
- OS_AMPL_SUFFIX_DIRECTION_local
 - OSnl2OS.h, [1082](#)
- OS_AMPL_SUFFIX_DIRECTION_toAMPL
 - OSnl2OS.h, [1082](#)
- OS_AMPL_SUFFIX_DIRECTION_toSolver
 - OSnl2OS.h, [1082](#)
- OS_AMPL_SUFFIX_SCOPE_constraints
 - OSnl2OS.h, [1081](#)
- OS_AMPL_SUFFIX_SCOPE_objectives
 - OSnl2OS.h, [1081](#)
- OS_AMPL_SUFFIX_SCOPE_problems
 - OSnl2OS.h, [1081](#)
- OS_AMPL_SUFFIX_SCOPE_variables
 - OSnl2OS.h, [1081](#)
- OS_AMPL_SUFFIX_TYPE_double
 - OSnl2OS.h, [1081](#)
- OS_AMPL_SUFFIX_TYPE_integer
 - OSnl2OS.h, [1081](#)
- OSILEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- OSOLATTRIBUTETEXT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- OSOLEND
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- OSOLSTART
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- OSOLSTARTEMPTY
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1213](#)
 - OSParseosrl.tab.hpp, [1286](#)
- OSParameters.h
 - ENUM_BASIS_STATUS_NUMBER_OF_STATES, [1341](#)
 - ENUM_BASIS_STATUS_atEquality, [1341](#)
 - ENUM_BASIS_STATUS_atLower, [1341](#)
 - ENUM_BASIS_STATUS_atUpper, [1341](#)
 - ENUM_BASIS_STATUS_basic, [1341](#)
 - ENUM_BASIS_STATUS_isFree, [1341](#)
 - ENUM_BASIS_STATUS_superbasic, [1341](#)
 - ENUM_BASIS_STATUS_unknown, [1341](#)
 - ENUM_COMBINE_ARRAYS_ignore, [1344](#)
 - ENUM_COMBINE_ARRAYS_merge, [1344](#)
 - ENUM_COMBINE_ARRAYS_replace, [1344](#)
 - ENUM_COMBINE_ARRAYS_throw, [1344](#)
 - ENUM_CONE_TYPE_completelyPositiveMatrices, [1345](#)
 - ENUM_CONE_TYPE_copositiveMatrices, [1345](#)
 - ENUM_CONE_TYPE_dual, [1345](#)
 - ENUM_CONE_TYPE_hyperbolicity, [1345](#)
 - ENUM_CONE_TYPE_intersection, [1345](#)
 - ENUM_CONE_TYPE_moments, [1345](#)
 - ENUM_CONE_TYPE_nonnegative, [1345](#)
 - ENUM_CONE_TYPE_nonnegativePolynomials, [1345](#)
 - ENUM_CONE_TYPE_nonpositive, [1345](#)
 - ENUM_CONE_TYPE_normed, [1345](#)
 - ENUM_CONE_TYPE_orthant, [1345](#)
 - ENUM_CONE_TYPE_polar, [1345](#)
 - ENUM_CONE_TYPE_polyhedral, [1345](#)
 - ENUM_CONE_TYPE_product, [1345](#)
 - ENUM_CONE_TYPE_quadratic, [1345](#)
 - ENUM_CONE_TYPE_rotatedNormed, [1345](#)
 - ENUM_CONE_TYPE_rotatedQuadratic, [1345](#)
 - ENUM_CONE_TYPE_semidefinite, [1345](#)
 - ENUM_CONE_TYPE_sumOfSquaresPolynomials, [1345](#)
 - ENUM_CONE_TYPE_unknown, [1345](#)
 - ENUM_CONREFERENCE_VALUETYPE_shortage, [1343](#)
 - ENUM_CONREFERENCE_VALUETYPE_status, [1343](#)
 - ENUM_CONREFERENCE_VALUETYPE_surplus, [1343](#)
 - ENUM_CONREFERENCE_VALUETYPE_value, [1343](#)
 - ENUM_CPUSPEEDUNIT_flops, [1339](#)
 - ENUM_CPUSPEEDUNIT_gigaflops, [1339](#)
 - ENUM_CPUSPEEDUNIT_gigahertz, [1338](#)
 - ENUM_CPUSPEEDUNIT_hertz, [1338](#)
 - ENUM_CPUSPEEDUNIT_kiloflops, [1339](#)
 - ENUM_CPUSPEEDUNIT_kilohertz, [1338](#)
 - ENUM_CPUSPEEDUNIT_megaflops, [1339](#)
 - ENUM_CPUSPEEDUNIT_megahertz, [1338](#)
 - ENUM_CPUSPEEDUNIT_petaflops, [1339](#)
 - ENUM_CPUSPEEDUNIT_teraflops, [1339](#)
 - ENUM_CPUSPEEDUNIT_terahertz, [1338](#)
 - ENUM_GENERAL_RESULT_STATUS_error, [1341](#)
 - ENUM_GENERAL_RESULT_STATUS_normal, [1341](#)
 - ENUM_GENERAL_RESULT_STATUS_warning, [1341](#)
 - ENUM_JOB_STATUS_finished, [1341](#)
 - ENUM_JOB_STATUS_killed, [1341](#)
 - ENUM_JOB_STATUS_running, [1341](#)
 - ENUM_JOB_STATUS_unknown, [1341](#)
 - ENUM_JOB_STATUS_waiting, [1341](#)

- ENUM_LOCATIONTYPE_ftp, [1340](#)
- ENUM_LOCATIONTYPE_http, [1340](#)
- ENUM_LOCATIONTYPE_local, [1340](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_base-Matrix, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_block, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_blocks, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_conRef-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_constant-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_general-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_linear-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_matrix, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_objRef-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_rowRef-Elements, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_transformation, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_unknown, [1344](#)
- ENUM_MATRIX_CONSTRUCTOR_TYPE_varRef-Elements, [1344](#)
- ENUM_MATRIX_SYMMETRY_HermitianLower, [1344](#)
- ENUM_MATRIX_SYMMETRY_HermitianUpper, [1344](#)
- ENUM_MATRIX_SYMMETRY_lower, [1343](#)
- ENUM_MATRIX_SYMMETRY_none, [1343](#)
- ENUM_MATRIX_SYMMETRY_skewLower, [1344](#)
- ENUM_MATRIX_SYMMETRY_skewUpper, [1344](#)
- ENUM_MATRIX_SYMMETRY_upper, [1343](#)
- ENUM_MATRIX_TYPE_conref, [1343](#)
- ENUM_MATRIX_TYPE_constant, [1343](#)
- ENUM_MATRIX_TYPE_general, [1343](#)
- ENUM_MATRIX_TYPE_jumbled, [1343](#)
- ENUM_MATRIX_TYPE_linear, [1343](#)
- ENUM_MATRIX_TYPE_mixedref, [1343](#)
- ENUM_MATRIX_TYPE_objref, [1343](#)
- ENUM_MATRIX_TYPE_quadratic, [1343](#)
- ENUM_MATRIX_TYPE_unknown, [1343](#)
- ENUM_MATRIX_TYPE_varref, [1343](#)
- ENUM_MATRIX_TYPE_zero, [1343](#)
- ENUM_NL_EXPR_SHAPE_constant, [1344](#)
- ENUM_NL_EXPR_SHAPE_convex, [1344](#)
- ENUM_NL_EXPR_SHAPE_general, [1344](#)
- ENUM_NL_EXPR_SHAPE_linear, [1344](#)
- ENUM_NL_EXPR_SHAPE_quadratic, [1344](#)
- ENUM_OUTPUT_AREA_Command_line_parser, [1338](#)
- ENUM_OUTPUT_AREA_NUMBER_OF_AREAS, [1338](#)
- ENUM_OUTPUT_AREA_OSAgent, [1338](#)
- ENUM_OUTPUT_AREA_OSGeneral, [1338](#)
- ENUM_OUTPUT_AREA_OSInstance, [1338](#)
- ENUM_OUTPUT_AREA_OSMatrix, [1338](#)
- ENUM_OUTPUT_AREA_OSModelInterfaces, [1338](#)
- ENUM_OUTPUT_AREA_OSOption, [1338](#)
- ENUM_OUTPUT_AREA_OSResult, [1338](#)
- ENUM_OUTPUT_AREA_OSSolverInterfaces, [1338](#)
- ENUM_OUTPUT_AREA_OSUtils, [1338](#)
- ENUM_OUTPUT_AREA_OSiL_parser, [1338](#)
- ENUM_OUTPUT_AREA_OSiLwriter, [1338](#)
- ENUM_OUTPUT_AREA_OSoL_parser, [1338](#)
- ENUM_OUTPUT_AREA_OSoLwriter, [1338](#)
- ENUM_OUTPUT_AREA_OSrL_parser, [1338](#)
- ENUM_OUTPUT_AREA_OSrLwriter, [1338](#)
- ENUM_OUTPUT_AREA_main, [1338](#)
- ENUM_OUTPUT_LEVEL_NUMBER_OF_LEVELS, [1338](#)
- ENUM_OUTPUT_LEVEL_always, [1337](#)
- ENUM_OUTPUT_LEVEL_debug, [1338](#)
- ENUM_OUTPUT_LEVEL_detailed_trace, [1338](#)
- ENUM_OUTPUT_LEVEL_error, [1337](#)
- ENUM_OUTPUT_LEVEL_info, [1338](#)
- ENUM_OUTPUT_LEVEL_summary, [1337](#)
- ENUM_OUTPUT_LEVEL_trace, [1338](#)
- ENUM_OUTPUT_LEVEL_warning, [1338](#)
- ENUM_PATHPAIR_input_dir, [1343](#)
- ENUM_PATHPAIR_input_file, [1343](#)
- ENUM_PATHPAIR_output_dir, [1343](#)
- ENUM_PATHPAIR_output_file, [1343](#)
- ENUM_PROBLEM_COMPONENT_constraints, [1342](#)
- ENUM_PROBLEM_COMPONENT_objectives, [1342](#)
- ENUM_PROBLEM_COMPONENT_variables, [1342](#)
- ENUM_SERVICE_TYPE_agent, [1340](#)
- ENUM_SERVICE_TYPE_analyzer, [1340](#)
- ENUM_SERVICE_TYPE_modeler, [1340](#)
- ENUM_SERVICE_TYPE_registry, [1340](#)
- ENUM_SERVICE_TYPE_scheduler, [1340](#)
- ENUM_SERVICE_TYPE_simulations, [1340](#)
- ENUM_SERVICE_TYPE_solver, [1340](#)
- ENUM_SOLUTION_STATUS_bestSoFar, [1342](#)
- ENUM_SOLUTION_STATUS_error, [1342](#)
- ENUM_SOLUTION_STATUS_feasible, [1342](#)
- ENUM_SOLUTION_STATUS_globallyOptimal, [1342](#)
- ENUM_SOLUTION_STATUS_infeasible, [1342](#)
- ENUM_SOLUTION_STATUS_locallyOptimal, [1342](#)
- ENUM_SOLUTION_STATUS_optimal, [1342](#)
- ENUM_SOLUTION_STATUS_other, [1342](#)
- ENUM_SOLUTION_STATUS_unbounded, [1342](#)

- ENUM_SOLUTION_STATUS_unsure, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_other, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByBounds, [1342](#)
- ENUM_SOLUTION_SUBSTATUSTYPE_stoppedByLimit, [1342](#)
- ENUM_STORAGEUNIT_byte, [1339](#)
- ENUM_STORAGEUNIT_exabyte, [1339](#)
- ENUM_STORAGEUNIT_gigabyte, [1339](#)
- ENUM_STORAGEUNIT_kilobyte, [1339](#)
- ENUM_STORAGEUNIT_megabyte, [1339](#)
- ENUM_STORAGEUNIT_petabyte, [1339](#)
- ENUM_STORAGEUNIT_terabyte, [1339](#)
- ENUM_STORAGEUNIT_yottabyte, [1339](#)
- ENUM_STORAGEUNIT_zettabyte, [1339](#)
- ENUM_SYSTEM_CURRENT_STATE_busy, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_busyButAccepting, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_idle, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_idleButNotAccepting, [1341](#)
- ENUM_SYSTEM_CURRENT_STATE_noResponse, [1341](#)
- ENUM_TIMECATEGORY_input, [1340](#)
- ENUM_TIMECATEGORY_optimization, [1340](#)
- ENUM_TIMECATEGORY_other, [1340](#)
- ENUM_TIMECATEGORY_output, [1340](#)
- ENUM_TIMECATEGORY_postprocessing, [1340](#)
- ENUM_TIMECATEGORY_preprocessing, [1340](#)
- ENUM_TIMECATEGORY_total, [1340](#)
- ENUM_TIMETYPE_cpuTime, [1339](#)
- ENUM_TIMETYPE_elapsedTime, [1339](#)
- ENUM_TIMETYPE_other, [1339](#)
- ENUM_TIMEUNIT_day, [1339](#)
- ENUM_TIMEUNIT_hour, [1339](#)
- ENUM_TIMEUNIT_millisecond, [1339](#)
- ENUM_TIMEUNIT_minute, [1339](#)
- ENUM_TIMEUNIT_month, [1339](#)
- ENUM_TIMEUNIT_second, [1339](#)
- ENUM_TIMEUNIT_tick, [1339](#)
- ENUM_TIMEUNIT_week, [1339](#)
- ENUM_TIMEUNIT_year, [1339](#)
- ENUM_TRANSPORT_TYPE_ftp, [1340](#)
- ENUM_TRANSPORT_TYPE_http, [1340](#)
- ENUM_TRANSPORT_TYPE_osp, [1340](#)
- ENUM_TRANSPORT_TYPE_other, [1340](#)
- ENUM_TRANSPORT_TYPE_smtp, [1340](#)
- ENUM_VARTYPE_binary, [1342](#)
- ENUM_VARTYPE_continuous, [1342](#)
- ENUM_VARTYPE_integer, [1342](#)
- ENUM_VARTYPE_semicontinuous, [1342](#)
- ENUM_VARTYPE_semiinteger, [1342](#)
- ENUM_VARTYPE_string, [1342](#)
- OSParseosil.tab.hpp
- ABSEND, [1137](#), [1147](#), [1156](#)
- ABSSTART, [1137](#), [1147](#), [1156](#)
- ACTUALSTARTTIMEEND, [1150](#)
- ACTUALSTARTTimestart, [1150](#)
- ALLDIFFEND, [1137](#), [1147](#), [1156](#)
- ALLDIFFSTART, [1137](#), [1147](#), [1156](#)
- ATEQUALITYEND, [1142](#), [1150](#)
- ATEQUALITYSTART, [1142](#), [1150](#)
- ATLOWEREND, [1142](#), [1150](#)
- ATLOWERSTART, [1142](#), [1150](#)
- ATTRIBUTETEXT, [1130](#), [1138](#), [1148](#)
- ATUPPEREND, [1142](#), [1150](#)
- ATUPPERSTART, [1142](#), [1150](#)
- AVAILABLECPUNUMBEREND, [1150](#)
- AVAILABLECPUNUMBERSTART, [1150](#)
- AVAILABLECPUSPEEDEND, [1150](#)
- AVAILABLECPUSPEEDSTART, [1150](#)
- AVAILABLEDISKSPACEEND, [1150](#)
- AVAILABLEDISKSPACESTART, [1150](#)
- AVAILABLEMEMORYEND, [1150](#)
- AVAILABLEMEMORYSTART, [1150](#)
- AXISDIRECTIONATT, [1132](#)
- BASE64END, [1134](#), [1143](#), [1150](#)
- BASE64START, [1134](#), [1143](#), [1150](#)
- BASEMATRIXEND, [1134](#), [1144](#), [1154](#)
- BASEMATRIXENDCOLATT, [1135](#), [1145](#), [1154](#)
- BASEMATRIXENDROWATT, [1135](#), [1145](#), [1154](#)
- BASEMATRIXIDXATT, [1135](#), [1145](#), [1154](#)
- BASEMATRIXSTART, [1134](#), [1144](#), [1154](#)
- BASEMATRIXSTARTCOLATT, [1135](#), [1145](#), [1154](#)
- BASEMATRIXSTARTROWATT, [1135](#), [1145](#), [1154](#)
- BASETRANSPOSEATT, [1135](#), [1145](#), [1154](#)
- BASICEND, [1142](#), [1150](#)
- BASICSTART, [1142](#), [1150](#)
- BASISSTATUSEND, [1151](#)
- BASISSTATUSSTART, [1150](#)
- BASSTATUSEND, [1151](#)
- BASSTATUSSTART, [1151](#)
- BLOCKCOLIDXATT, [1136](#), [1146](#), [1155](#)
- BLOCKEND, [1134](#), [1144](#), [1154](#)
- BLOCKROWIDXATT, [1136](#), [1146](#), [1155](#)
- BLOCKSEND, [1134](#), [1144](#), [1154](#)
- BLOCKSSTART, [1134](#), [1144](#), [1154](#)
- BLOCKSTART, [1134](#), [1144](#), [1154](#)
- CATEGORYATT, [1139](#), [1149](#)
- COEFATT, [1135](#), [1145](#), [1154](#)
- COLOFFSETSEND, [1136](#), [1146](#), [1155](#)
- COLOFFSETSSTART, [1136](#), [1146](#), [1155](#)
- COMPONENTSEND, [1132](#)
- COMPONENTSSTART, [1132](#)
- CONEND, [1133](#), [1143](#), [1151](#)
- CONESSEND, [1131](#)
- CONESSTART, [1131](#)

CONREFERENCEELEMENTSEND, 1135, 1145, 1155
CONREFERENCEELEMENTSSTART, 1135, 1145, 1155
CONREFERENCEMATRIXIDXATT, 1133
CONSTANTATT, 1135, 1144, 1154
CONSTANTELEMENTSEND, 1135, 1145, 1154
CONSTANTELEMENTSSTART, 1135, 1145, 1154
CONSTANTMATRIXIDXATT, 1133
CONSTANT, 1133, 1143, 1151
CONSTRAINTSEND, 1133, 1143, 1151
CONSTRAINTSSTART, 1133, 1143, 1151
CONTACTEND, 1141
CONTACTSTART, 1141
CONTYPEATT, 1139, 1149
COSEND, 1137, 1147, 1156
COSSTART, 1137, 1147, 1156
CURRENTJOBCOUNTEND, 1151
CURRENTJOBCOUNTSTART, 1151
CURRENTSTATEEND, 1151
CURRENTSTATESTART, 1151
DEPENDENCIESEND, 1141
DEPENDENCIESSTART, 1141
DESCRIPTIONATT, 1139, 1149
DIRECTIONEND, 1132
DIRECTIONSTART, 1132
DIRECTORIESTODELETEEND, 1142
DIRECTORIESTODELETESTART, 1142
DIRECTORIESTOMAKEEND, 1141
DIRECTORIESTOMAKESTART, 1141
DISTORTIONMATRIXIDXATT, 1132
DIVIDEEND, 1136, 1146, 1156
DIVIDESTART, 1136, 1146, 1156
DOUBLE, 1131, 1138, 1148
DUALCONEEND, 1132
DUALCONESTART, 1132
DUALVALUESEND, 1151
DUALVALUESSTART, 1151
DUMMY, 1136, 1146, 1155
EEND, 1137, 1147, 1156
ELEMENTSEND, 1135, 1145, 1154
ELEMENTSSTART, 1135, 1145, 1154
ELEMENTTEXT, 1130, 1138, 1148
EEND, 1134, 1143, 1151
ELSTART, 1134, 1143, 1151
EMPTYBASETRANPOSEATT, 1135, 1145, 1154
EMPTYCATEGORYATT, 1140, 1149
EMPTYCONTYPEATT, 1139, 1149
EMPTYDESCRIPTIONATT, 1140, 1149
EMPTYENUMTYPEATT, 1139, 1149
EMPTYIDATT, 1131
EMPTYINCLUDEDIAGONALATT, 1138, 1148, 1158
EMPTYLBBDUALVALUEATT, 1140
EMPTYLBVALUEATT, 1140
EMPTYNAMEATT, 1134, 1140, 1149
EMPTYNEGATIVEPATTERNATT, 1135, 1144, 1154
EMPTYOBJTYPEATT, 1140, 1149
EMPTYROWMAJORATT, 1136, 1146, 1155
EMPTYSEMIDEFINITENESSATT, 1132
EMPTYSHAPEATT, 1134, 1144, 1154
EMPTYSOLVERATT, 1140
EMPTYSYMMETRYATT, 1134, 1144, 1154
EMPTYTARGETOBJECTIVENAMEATT, 1150
EMPTYTYPEATT, 1134, 1140, 1149
EMPTYUBDUALVALUEATT, 1140
EMPTYUBVALUEATT, 1140
EMPTYUNITATT, 1140, 1149
EMPTYVALUEATT, 1140, 1150
EMPTYVARTYPEATT, 1140, 1149
EMPTYWEIGHTATT, 1140
EMPTYWEIGHTEDOBJECTIVESATT, 1150
ENDOFELEMENT, 1131, 1139, 1148
ENDTIMEEND, 1151
ENDTimestart, 1151
ENUMERATIONEND, 1134, 1143, 1151
ENUMERATIONSTART, 1134, 1143, 1151
ENUMTYPEATT, 1139, 1149
ERFEND, 1137, 1147, 1156
ERFSTART, 1137, 1147, 1156
ESTART, 1137, 1147, 1156
EXPEND, 1137, 1146, 1156
EXPEND, 1136, 1146, 1155
EXPRSTART, 1136, 1146, 1155
EXPSTART, 1137, 1146, 1156
FACTORSEND, 1132
FACTORSTART, 1132
FILECREATOREMPTY, 1134, 1144, 1153
FILECREATOREND, 1134, 1144, 1153
FILECREATORSTART, 1134, 1144, 1153
FILECREATORSTARTANDEND, 1134, 1144, 1153
FILEDESCRIPTIONEMPTY, 1134, 1144, 1153
FILEDESCRIPTIONEND, 1134, 1144, 1153
FILEDESCRIPTIONSTART, 1134, 1144, 1153
FILEDESCRIPTIONSTARTANDEND, 1134, 1144, 1153
FILELICENCEEMPTY, 1134, 1144, 1154
FILELICENCEEND, 1134, 1144, 1154
FILELICENCESTART, 1134, 1144, 1153
FILELICENCESTARTANDEND, 1134, 1144, 1154
FILENAMEEMPTY, 1133, 1144, 1153
FILENAMEEND, 1133, 1144, 1153
FILENAMESTART, 1133, 1144, 1153
FILENAMESTARTANDEND, 1133, 1144, 1153
FILESOURCEEMPTY, 1133, 1144, 1153
FILESOURCEEND, 1133, 1144, 1153
FILESOURCESTART, 1133, 1144, 1153
FILESOURCESTARTANDEND, 1134, 1144, 1153
FILESTODELETEEND, 1142

FILESTODELETESTART, 1142
FILESTOMAKEEND, 1142
FILESTOMAKESTART, 1141
FIRSTAXISDIRECTIONATT, 1132
FROMATT, 1139
GENERALELEMENTSEND, 1135, 1145, 1155
GENERALELEMENTSSTART, 1135, 1145, 1155
GENERALEND, 1140, 1150
GENERALSTART, 1140, 1150
GENERALSTATUSEND, 1151
GENERALSTATUSSTART, 1151
GENERALSUBSTATUSEND, 1151
GENERALSUBSTATUSSTART, 1151
GREATERTHAN, 1131, 1139, 1148
GROUPWEIGHTATT, 1139
HEADEREND, 1133, 1144, 1150
HEADERSTART, 1133, 1144, 1150
HORIZONATT, 1133
IDATT, 1138, 1148, 1158
IDENTITYMATRIXEND, 1138, 1148, 1157
IDENTITYMATRIXSTART, 1138, 1148, 1157
IDXATT, 1135, 1139, 1149
IDXEND, 1151
IDXONEATT, 1131
IDXSTART, 1151
IDXTWOATT, 1131
IFEND, 1137, 1147, 1156
IFSTART, 1137, 1147, 1156
INCLUDEDIAGONALATT, 1138, 1148, 1158
INCRATT, 1134, 1143, 1149
INDEXESEND, 1135, 1145, 1155
INDEXESSTART, 1135, 1145, 1154
INITIALBASISSTATUSEND, 1142
INITIALBASISSTATUSSTART, 1142
INITIALCONSTRAINTVALUESEND, 1143
INITIALCONSTRAINTVALUESSTART, 1143
INITIALDUALVALUESEND, 1143
INITIALDUALVALUESSTART, 1143
INITIALOBJECTIVEBOUNDSEND, 1143
INITIALOBJECTIVEBOUNDSSTART, 1143
INITIALOBJECTIVEVALUESEND, 1143
INITIALOBJECTIVEVALUESSTART, 1143
INITIALVARIABLEVALUESEND, 1142
INITIALVARIABLEVALUESSTART, 1142
INITIALVARIABLEVALUESSTRINGEND, 1142
INITIALVARIABLEVALUESSTRINGSTART, 1142
INPUTDIRECTORIESTOMOVEEND, 1142
INPUTDIRECTORIESTOMOVESTART, 1142
INPUTFILESTOMOVEEND, 1142
INPUTFILESTOMOVESTART, 1142
INSTANCEDATAEND, 1131
INSTANCEDATASTARTEND, 1131
INSTANCELOCATIONEND, 1140
INSTANCELOCATIONSTART, 1140
INSTANCENAMEEND, 1140, 1151
INSTANCENAMESTART, 1140, 1151
INTEGER, 1131, 1138, 1148
INTEGERVARIABLEBRANCHINGWEIGHTSEND, 1143
INTEGERVARIABLEBRANCHINGWEIGHTSSTART, 1142
INTERSECTIONCONEEND, 1132
INTERSECTIONCONESTART, 1131
INTERVALEND, 1133
INTERVALSTART, 1133
ISFREEEND, 1142, 1151
ISFREESTART, 1142, 1151
ITEMEMPTY, 1134, 1143, 1150
ITEMEND, 1134, 1143, 1150
ITEMSTART, 1134, 1143, 1150
ITEMSTARTANDEND, 1134, 1143, 1150
ITEMTEXT, 1131, 1138, 1148
JOBEND, 1140, 1150
JOBIDEND, 1141, 1151
JOBIDSTART, 1140, 1151
JOBSTART, 1140, 1150
LBCONEIDXATT, 1132
LBDUALVALUEATT, 1140
LBMATRIXIDXATT, 1132
LBVALUEATT, 1140
LICENSEEND, 1141
LICENSESTART, 1141
LINEARELEMENTSEND, 1135, 1145, 1155
LINEARELEMENTSSTART, 1135, 1145, 1155
LNEND, 1136, 1146, 1156
LNSTART, 1136, 1146, 1156
LOCATIONTYPEATT, 1140
MAKECOPYATT, 1139
MATRICESEND, 1131
MATRICESSTART, 1131
MATRIXCONEND, 1132, 1144, 1153
MATRIXCONSTART, 1132, 1144, 1153
MATRIXCONSTRAINTSEND, 1132
MATRIXCONSTRAINTSSTART, 1132
MATRIXDETERMINANTEND, 1137, 1147, 1157
MATRIXDETERMINANTSTART, 1137, 1147, 1157
MATRIXDIAGONALEND, 1138, 1147, 1157
MATRIXDIAGONALSTART, 1138, 1147, 1157
MATRIXDOTTIMESEND, 1138, 1147, 1157
MATRIXDOTTIMESSTART, 1138, 1147, 1157
MATRIXEND, 1134, 1144, 1154
MATRIXEXPRESSIONSEND, 1136, 1146, 1155
MATRIXEXPRESSIONSSTART, 1136, 1146, 1155
MATRIXIDXATT, 1132
MATRIXINVERSEEND, 1138, 1148, 1157
MATRIXINVERSESTART, 1138, 1148, 1157
MATRIXLOWERTRIANGLEEND, 1138, 1147, 1157

- MATRIXLOWERTRIANGLESTART, 1138, 1147, 1157
- MATRIXMERGEEND, 1138, 1148, 1157
- MATRIXMERGESTART, 1138, 1148, 1157
- MATRIXMINUSEND, 1138, 1148, 1157
- MATRIXMINUSSTART, 1138, 1148, 1157
- MATRIXNEGATEEND, 1138, 1148, 1157
- MATRIXNEGATESTART, 1138, 1148, 1157
- MATRIXOBJECTIVESEND, 1132
- MATRIXOBJECTIVESSTART, 1132
- MATRIXOBJEND, 1132, 1143, 1153
- MATRIXOBJSTART, 1132, 1143, 1153
- MATRIXPLUSEND, 1138, 1148, 1157
- MATRIXPLUSSTART, 1138, 1148, 1157
- MATRIXPRODUCTEND, 1138, 1148, 1157
- MATRIXPRODUCTSTART, 1138, 1148, 1157
- MATRIXPROGRAMMINGEND, 1132
- MATRIXPROGRAMMINGSTART, 1132
- MATRIXREFERENCEEND, 1138, 1148, 1157
- MATRIXREFERENCESTART, 1138, 1148, 1157
- MATRIXSCALARTIMESEND, 1138, 1148, 1157
- MATRIXSCALARTIMESSTART, 1138, 1148, 1157
- MATRIXSTART, 1134, 1144, 1154
- MATRIXSUBMATRIXATEND, 1138, 1148, 1157
- MATRIXSUBMATRIXATSTART, 1138, 1148, 1157
- MATRIXTERMEND, 1136, 1146, 1155
- MATRIXTERMSTART, 1136, 1146, 1155
- MATRIXTIMESEND, 1138, 1148, 1157
- MATRIXTIMESSTART, 1138, 1148, 1157
- MATRIXTOSCALAREND, 1137, 1147, 1157
- MATRIXTOSCALARSTART, 1137, 1147, 1157
- MATRIXTRACEEND, 1137, 1147, 1157
- MATRIXTRACESTART, 1137, 1147, 1157
- MATRIXTRANSPOSEEND, 1138, 1148, 1157
- MATRIXTRANSPOSESTART, 1138, 1148, 1157
- MATRIXUPPERTRIANGLEEND, 1138, 1147, 1157
- MATRIXUPPERTRIANGLESTART, 1138, 1147, 1157
- MATRIXVAREND, 1132, 1143, 1153
- MATRIXVARIABLESEND, 1132
- MATRIXVARIABLESSTART, 1132
- MATRIXVARSTART, 1132, 1143, 1153
- MAXEND, 1137, 1147, 1156
- MAXSTART, 1137, 1147, 1156
- MAXTIMEEND, 1141
- MAXTimestart, 1141
- MESSAGEEND, 1151
- MESSAGESTART, 1151
- MINCPUNUMBEREND, 1141
- MINCPUNUMBERSTART, 1141
- MINCPUSPEEDEND, 1141
- MINCPUSPEEDSTART, 1141
- MINDISKSPACEEND, 1141
- MINDISKSPACESTART, 1141
- MINEND, 1137, 1147, 1156
- MINMEMORYEND, 1141
- MINMEMORYSTART, 1141
- MINSTART, 1137, 1147, 1156
- MINUSEND, 1136, 1146, 1156
- MINUSSTART, 1136, 1146, 1156
- MULTATT, 1134, 1143, 1149
- NAMEATT, 1134, 1139, 1149
- NEGATEEND, 1137, 1146, 1156
- NEGATESTART, 1137, 1146, 1156
- NEGATIVEPATTERNATT, 1135, 1144, 1154
- NLEND, 1136, 1146, 1155
- NLSTART, 1136, 1146, 1155
- NONLINEAREXPRESSIONSEND, 1136, 1146, 1155
- NONLINEAREXPRESSIONSSTART, 1136, 1146, 1155
- NONNEGATIVECONEEND, 1131
- NONNEGATIVECONESTART, 1131
- NONPOSITIVECONEEND, 1131
- NONPOSITIVECONESTART, 1131
- NONZEROSSEND, 1135, 1145, 1154
- NONZEROSSTART, 1135, 1145, 1154
- NORMSCALEFACTORATT, 1132
- NUMBEREND, 1137, 1147, 1157
- NUMBEROFBLOCKSATT, 1135, 1144, 1154
- NUMBEROFCOLUMNSATT, 1135, 1144, 1154
- NUMBEROFCONATT, 1139, 1148
- NUMBEROFCONESATT, 1131
- NUMBEROFCONSTRAINTSATT, 1133, 1139, 1148
- NUMBEROFELATT, 1134, 1139, 1149
- NUMBEROFENUMERATIONSATT, 1139, 1149
- NUMBEROFEXPR, 1136, 1146, 1155
- NUMBEROFIDXATT, 1149
- NUMBEROFITEMSATT, 1139, 1149
- NUMBEROFJOBIDSATT, 1139
- NUMBEROFMATRICESATT, 1131
- NUMBEROFMATRIXCONATT, 1132
- NUMBEROFMATRIXOBJATT, 1132
- NUMBEROFMATRIXTERMSATT, 1136, 1146, 1155
- NUMBEROFMATRIXVARATT, 1132
- NUMBEROFNONLINEAREXPRESSIONS, 1136, 1146, 1155
- NUMBEROFOBJATT, 1139, 1149
- NUMBEROFOBJECTIVESATT, 1133, 1139, 1149
- NUMBEROFOTHERCONSTRAINTOPTIONSATT, 1139
- NUMBEROFOTHERCONSTRAINTRESULTSATT, 1149
- NUMBEROFOTHEROBJECTIVEOPTIONSATT, 1139
- NUMBEROFOTHEROBJECTIVERESULTSATT, 1149
- NUMBEROFOTHEROPTIONSATT, 1139
- NUMBEROFOTHERRESULTSATT, 1149

NUMBEROFOTHERSOLUTIONRESULTSATT, 1149
NUMBEROFOTHERVARIABLEOPTIONSATT, 1139
NUMBEROFOTHERVARIABLERESULTSATT, 1149
NUMBEROFPATHPAIRSATT, 1139
NUMBEROFPATHSATT, 1139
NUMBEROFPROCESSESATT, 1139
NUMBEROFQTERMSATT, 1131
NUMBEROFROWSATT, 1135, 1144, 1154
NUMBEROFSOLUTIONSATT, 1149
NUMBEROFSOLVEROPTIONSATT, 1139
NUMBEROFSOLVEROUTPUTSATT, 1149
NUMBEROFSOSATT, 1139
NUMBEROFSTAGESATT, 1133
NUMBEROFSUBSTATUSESATT, 1149
NUMBEROFTIMESATT, 1149
NUMBEROFVALUESATT, 1135, 1145, 1154
NUMBEROFVARATT, 1139, 1149
NUMBEROFVARIABLESATT, 1133, 1139, 1149
NUMBEROFVARIDXATT, 1135, 1145, 1149
NUMBERSTART, 1137, 1147, 1157
OBJECTIVESEND, 1133, 1143, 1151
OBJECTIVESSTART, 1133, 1143, 1151
OBJEND, 1133, 1143, 1151
OBJREFERENCEELEMENTSEND, 1136, 1145, 1155
OBJREFERENCEELEMENTSSTART, 1136, 1145, 1155
OBJREFERENCEMATRIXIDXATT, 1133
OBJSTART, 1133, 1143, 1151
OBJTYPEATT, 1139, 1149
OPTIMIZATIONEND, 1140, 1150
OPTIMIZATIONSOLUTIONSTATUSSEND, 1151
OPTIMIZATIONSOLUTIONSTATUSSTART, 1151
OPTIMIZATIONSOLUTIONSUBSTATUSSEND, 1151
OPTIMIZATIONSOLUTIONSUBSTATUSSTART, 1151
OPTIMIZATIONSTART, 1140, 1150
ORDERCONEIDXATT, 1133
ORTHANTCONEEND, 1131
ORTHANTCONESTART, 1131
OSILEND, 1131
OSOLATTRIBUTETEXT, 1139
OSOLEND, 1139
OSOLSTART, 1139
OSOLSTARTEMPT, 1139
OSRLATTRIBUTETEXT, 1148
OSRLEND, 1148
OSRLSTART, 1148
OSRLSTARTEMPT, 1148
OTHEREND, 1141, 1152
OTHEROPTIONSSEND, 1141
OTHEROPTIONSSTART, 1141
OTHERRESULTSEND, 1152
OTHERRESULTSSTART, 1152
OTHERSOLUTIONRESULTEND, 1152
OTHERSOLUTIONRESULTSEND, 1152
OTHERSOLUTIONRESULTSSTART, 1152
OTHERSOLUTIONRESULTSTART, 1152
OTHERSOLVEROUTPUTEND, 1152
OTHERSOLVEROUTPUTSTART, 1152
OTHERSTART, 1141, 1151
OUTPUTDIRECTORIESTOMOVEEND, 1142
OUTPUTDIRECTORIESTOMOVESTART, 1142
OUTPUTFILESTOMOVEEND, 1142
OUTPUTFILESTOMOVESTART, 1142
PASSWORDEND, 1141
PASSWORDSTART, 1141
PATHEND, 1141
PATHPAIREND, 1141
PATHPAIRSTART, 1141
PATHSTART, 1141
PATTERNELEMENTSEND, 1136, 1145, 1155
PATTERNELEMENTSSTART, 1136, 1145, 1155
PIEND, 1137, 1147, 1156
PISTART, 1137, 1147, 1156
PLUSSEND, 1136, 1146, 1156
PLUSSTART, 1136, 1146, 1156
POLARCONEEND, 1132
POLARCONESTART, 1132
POLYHEDRALCONEEND, 1131
POLYHEDRALCONESTART, 1131
POWEREND, 1136, 1146, 1155
POWERSTART, 1136, 1146, 1155
PROCESSEND, 1142
PROCESSESTOKILLEND, 1142
PROCESSESTOKILLSTART, 1142
PROCESSSTART, 1142
PRODUCTCONEEND, 1131
PRODUCTCONESTART, 1131
PRODUCTEND, 1137, 1146, 1156
PRODUCTSTART, 1137, 1146, 1156
QTERMEND, 1131
QTERMSTART, 1131
QUADRATICCOEFFICIENTSEND, 1131
QUADRATICCOEFFICIENTSSTART, 1131
QUADRATICCONEEND, 1131
QUADRATICCONESTART, 1131
QUOTE, 1130, 1138, 1148
REFERENCEMATRIXIDXATT, 1132
REQUESTEDSTARTTIMEEND, 1141
REQUESTEDSTARTTimestart, 1141
REQUIREDDIRECTORIESEND, 1141
REQUIREDDIRECTORIESSTART, 1141
REQUIREDFILESSEND, 1141
REQUIREDFILESSTART, 1141
ROTATEDQUADRATICCONEEND, 1131
ROTATEDQUADRATICCONESTART, 1131

ROWMAJORATT, 1136, 1146, 1155
ROWOFFSETSEND, 1136, 1146, 1155
ROWOFFSETSSTART, 1136, 1146, 1155
SCALARMULTIPLIERATT, 1135, 1145, 1154
SCHEDULEDSTARTTIMEEND, 1152
SCHEDULEDSTARTTimestart, 1152
SECONDAXISDIRECTIONATT, 1132
SEMIDEFINITECONEEND, 1131
SEMIDEFINITECONESTART, 1131
SEMIDEFINITECESSATT, 1132
SERVICEEND, 1140, 1150
SERVICENAMEEND, 1140, 1152
SERVICENAMESTART, 1140, 1152
SERVICESTART, 1140, 1150
SERVICETYPEEND, 1141
SERVICETYPESTART, 1141
SERVICEURIEND, 1140, 1152
SERVICEURISTART, 1140, 1152
SERVICEUTILIZATIONEND, 1152
SERVICEUTILIZATIONSTART, 1152
SHAPEATT, 1134, 1144, 1154
SINEND, 1137, 1147, 1156
SINSTART, 1137, 1147, 1156
SIZEOFATT, 1134, 1143, 1149
SOLUTIONEND, 1152
SOLUTIONSTART, 1152
SOLVERATT, 1140
SOLVERINVOKEDEND, 1152
SOLVERINVOKEDSTART, 1152
SOLVEROPTIONEND, 1143
SOLVEROPTIONSSEND, 1143
SOLVEROPTIONSSTART, 1143
SOLVEROPTIONSTART, 1143
SOLVEROUTPUTEND, 1152
SOLVEROUTPUTSTART, 1152
SOLVERTOINVOKEEND, 1141
SOLVERTOINVOKESTART, 1141
SOSEND, 1143
SOSIDXATT, 1139
SOSSTART, 1143
SOSVARIABLEBRANCHINGWEIGHTSEND, 1143
SOSVARIABLEBRANCHINGWEIGHTSSTART, 1143
SQRTEND, 1136, 1146, 1156
SQRTSTART, 1136, 1146, 1156
SQUAREEND, 1137, 1147, 1156
SQUARESTART, 1137, 1147, 1156
STAGEEND, 1133
STAGESEND, 1133
STAGESSTART, 1133
STAGESTART, 1133
STARTATT, 1133
STARTIDXATT, 1133
STARTVECTOREND, 1135, 1145, 1154
STARTVECTORSTART, 1135, 1145, 1154
STATUSEND, 1152
STATUSSTART, 1152
SUBMITTIMEEND, 1152
SUBMITTimestart, 1152
SUBSTATUSEND, 1152
SUBSTATUSSTART, 1152
SUMEND, 1137, 1146, 1156
SUMSTART, 1137, 1146, 1156
SUPERBASICEND, 1142, 1152
SUPERBASICSTART, 1142, 1152
SYMMETRYATT, 1135, 1144, 1154
SYSTEMEND, 1140, 1150
SYSTEMINFORMATIONEND, 1152
SYSTEMINFORMATIONSTART, 1152
SYSTEMSTART, 1140, 1150
TARGETMATRIXFIRSTCOLATT, 1135, 1145, 1154
TARGETMATRIXFIRSTROWATT, 1135, 1145, 1154
TARGETOBJECTIVEIDXATT, 1149
TARGETOBJECTIVENAMEATT, 1150
TEMPLATMATRIXIDXATT, 1133
TIMEDOMAINEND, 1133
TIMEDOMAINSTART, 1133
TIMEEND, 1152
TIMESEND, 1137, 1147, 1157
TIMESERVICESTARTEDEND, 1152
TIMESERVICESTARTEDSTART, 1152
TIMESSTART, 1137, 1147, 1157
TIMESTAMPEND, 1152
TIMESTAMPSTART, 1152
Timestart, 1152
TIMINGINFORMATIONEND, 1152
TIMINGINFORMATIONSTART, 1152
TOATT, 1139
TOTALJOBSSOFAREND, 1153
TOTALJOBSSOFARSTART, 1152
TRANSFORMATIONEND, 1136, 1145, 1155
TRANSFORMATIONSTART, 1136, 1145, 1155
TRANSPORTTYPEATT, 1140
TWOQUOTES, 1131, 1138, 1148
TYPEATT, 1134, 1139, 1149
UBCONEIDXATT, 1132
UBDUALVALUEATT, 1140
UBMATRIXIDXATT, 1132
UBVALUEATT, 1140
UNITATT, 1139, 1149
UNKNOWNEND, 1142, 1153
UNKNOWNSTART, 1142, 1153
USEDPCUNUMBEREND, 1153
USEDPCUNUMBERSTART, 1153
USEDPCUSPEEDEND, 1153
USEDPCUSPEEDSTART, 1153
USEDISKSPACEEND, 1153
USEDISKSPACESTART, 1153

- USEDMEMORYEND, 1153
- USEDMEMORYSTART, 1153
- USERNAMEEND, 1141
- USERNAMESTART, 1141
- VALUEATT, 1131, 1139, 1149
- VALUESEND, 1135, 1145, 1155
- VALUESSTART, 1135, 1145, 1155
- VALUESSTRINGEND, 1153
- VALUESSTRINGSTART, 1153
- VALUETYPEATT, 1135, 1145, 1155
- VAREND, 1133, 1142, 1153
- VARIABLEEND, 1137, 1147, 1156
- VARIABLESEND, 1133, 1142, 1153
- VARIABLESSTART, 1133, 1142, 1153
- VARIABLESTART, 1137, 1147, 1156
- VARIDXEND, 1136, 1145, 1153
- VARIDXSTART, 1136, 1145, 1153
- VARREFERENCEELEMENTSEND, 1135, 1145, 1155
- VARREFERENCEELEMENTSSTART, 1135, 1145, 1155
- VARREFERENCEMATRIXIDXATT, 1133
- VARSTART, 1133, 1142, 1153
- VARTYPEATT, 1132, 1140, 1149
- WEIGHTATT, 1140
- WEIGHTEDOBJECTIVESATT, 1150
- OSParseosol.tab.hpp
 - ABSEND, 1212, 1222, 1231
 - ABSSTART, 1212, 1222, 1231
 - ACTUALSTARTTIMEEND, 1225
 - ACTUALSTARTTimestart, 1225
 - ALLDIFFEND, 1212, 1222, 1231
 - ALLDIFFSTART, 1212, 1222, 1231
 - ATEQUALITYEND, 1217, 1225
 - ATEQUALITYSTART, 1217, 1225
 - ATLOWEREND, 1217, 1225
 - ATLOWERSTART, 1217, 1225
 - ATTRIBUTETEXT, 1205, 1213, 1223
 - ATUPPEREND, 1217, 1225
 - ATUPPERSTART, 1217, 1225
 - AVAILABLECPUNUMBEREND, 1225
 - AVAILABLECPUNUMBERSTART, 1225
 - AVAILABLECPUSPEEDEND, 1225
 - AVAILABLECPUSPEEDSTART, 1225
 - AVAILABLEDISKSPACEEND, 1225
 - AVAILABLEDISKSPACESTART, 1225
 - AVAILABLEMEMORYEND, 1225
 - AVAILABLEMEMORYSTART, 1225
 - AXISDIRECTIONATT, 1207
 - BASE64END, 1209, 1218, 1225
 - BASE64START, 1209, 1218, 1225
 - BASEMATRIXEND, 1209, 1219, 1228
 - BASEMATRIXENDCOLATT, 1210, 1220, 1229
 - BASEMATRIXENDROWATT, 1210, 1219, 1229
 - BASEMATRIXIDXATT, 1210, 1219, 1229
 - BASEMATRIXSTART, 1209, 1219, 1228
 - BASEMATRIXSTARTCOLATT, 1210, 1219, 1229
 - BASEMATRIXSTARTROWATT, 1210, 1219, 1229
 - BASETRANPOSEATT, 1210, 1220, 1229
 - BASICEND, 1217, 1225
 - BASICSTART, 1217, 1225
 - BASISSTATUSEND, 1225
 - BASISSTATUSSTART, 1225
 - BASSTATUSEND, 1225
 - BASSTATUSSTART, 1225
 - BLOCKCOLIDXATT, 1211, 1220, 1230
 - BLOCKEND, 1209, 1219, 1228
 - BLOCKROWIDXATT, 1211, 1220, 1230
 - BLOCKSEND, 1209, 1219, 1229
 - BLOCKSSTART, 1209, 1219, 1229
 - BLOCKSTART, 1209, 1219, 1228
 - CATEGORYATT, 1214, 1224
 - COEFATT, 1210, 1219, 1229
 - COOFFSETSEND, 1211, 1220, 1230
 - COOFFSETSTART, 1210, 1220, 1230
 - COMPONENTSEND, 1207
 - COMPONENTSSTART, 1207
 - CONEND, 1208, 1218, 1225
 - CONESEND, 1206
 - CONESSTART, 1206
 - CONREFERENCEELEMENTSEND, 1210, 1220, 1230
 - CONREFERENCEELEMENTSSTART, 1210, 1220, 1230
 - CONREFERENCEMATRIXIDXATT, 1207
 - CONSTANTATT, 1209, 1219, 1229
 - CONSTANTELEMENTSEND, 1210, 1220, 1229
 - CONSTANTELEMENTSSTART, 1210, 1220, 1229
 - CONSTANTMATRIXIDXATT, 1207
 - CONSTANT, 1208, 1218, 1225
 - CONSTRAINTSEND, 1208, 1218, 1225
 - CONSTRAINTSSTART, 1208, 1218, 1225
 - CONTACTEND, 1216
 - CONTACTSTART, 1216
 - CONTYPEATT, 1214, 1224
 - COSEND, 1212, 1221, 1231
 - COSSTART, 1212, 1221, 1231
 - CURRENTJOBCOUNTEND, 1225
 - CURRENTJOBCOUNTSTART, 1225
 - CURRENTSTATEEND, 1226
 - CURRENTSTATESTART, 1226
 - DEPENDENCIESEND, 1216
 - DEPENDENCIESSTART, 1216
 - DESCRIPTIONATT, 1214, 1224
 - DIRECTIONEND, 1206
 - DIRECTIONSTART, 1206
 - DIRECTORIESTODELETEEND, 1216
 - DIRECTORIESTODELETESTART, 1216

DIRECTORiestomakeend, 1216
DIRECTORiestomakestart, 1216
DISTORTIONMATRIXIDXATT, 1207
DIVIDEEND, 1211, 1221, 1230
DIVIDESTART, 1211, 1221, 1230
DOUBLE, 1205, 1213, 1223
DUALCONEEND, 1206
DUALCONESTART, 1206
DUALVALUESEND, 1226
DUALVALUESSTART, 1226
DUMMY, 1211, 1220, 1230
EEND, 1212, 1222, 1231
ELEMENTSEND, 1210, 1220, 1229
ELEMENTSSTART, 1210, 1220, 1229
ELEMENTTEXT, 1205, 1213, 1223
ELEN, 1209, 1218, 1226
ELSTART, 1209, 1218, 1226
EMPTYBASETRANSPoseatt, 1210, 1220, 1229
EMPTYCATEGORYATT, 1214, 1224
EMPTYCONTYPEATT, 1214, 1224
EMPTYDESCRIPTIONATT, 1214, 1224
EMPTYENUMTYPEATT, 1214, 1224
EMPTYIDATT, 1206
EMPTYINCLUDEDIAGONALATT, 1213, 1223, 1232
EMPTYLBDUALVALUEATT, 1215
EMPTYLBVALUEATT, 1215
EMPTYNAMEATT, 1209, 1214, 1224
EMPTYNEGATIVEPATTERNATT, 1209, 1219, 1229
EMPTYOBJTYPEATT, 1214, 1224
EMPTYROWMAJORATT, 1211, 1220, 1230
EMPTYSEMIDEFINITENESSATT, 1207
EMPTYSHAPEATT, 1209, 1219, 1229
EMPTYSOLVERATT, 1215
EMPTYSYMMETRYATT, 1209, 1219, 1229
EMPTYTARGETOBJECTIVENAMEATT, 1224
EMPTYTYPEATT, 1209, 1214, 1224
EMPTYUBDUALVALUEATT, 1215
EMPTYUBVALUEATT, 1215
EMPTYUNITATT, 1214, 1224
EMPTYVALUEATT, 1214, 1224
EMPTYVARTYPEATT, 1214, 1224
EMPTYWEIGHTATT, 1215
EMPTYWEIGHTEDOBJECTIVESATT, 1224
ENDOFELEMENT, 1205, 1213, 1223
ENDTIMEEND, 1226
ENDTimestart, 1226
ENUMERATIONEND, 1209, 1218, 1226
ENUMERATIONSTART, 1209, 1218, 1226
ENUMTYPEATT, 1214, 1224
ERFEND, 1212, 1222, 1231
ERFSTART, 1212, 1222, 1231
ESTART, 1212, 1222, 1231
EXPEND, 1211, 1221, 1231
EXPREND, 1211, 1221, 1230
EXPRSTART, 1211, 1221, 1230
EXPSTART, 1211, 1221, 1231
FACTORSEND, 1206
FACTORSSTART, 1206
FILECREATOREMPTY, 1208, 1219, 1228
FILECREATOREND, 1208, 1219, 1228
FILECREATORSTART, 1208, 1219, 1228
FILECREATORSTARTANDEND, 1208, 1219, 1228
FILEDESCRIPTIONEMPT, 1208, 1219, 1228
FILEDESCRIPTIONEND, 1208, 1219, 1228
FILEDESCRIPTIONSTART, 1208, 1219, 1228
FILEDESCRIPTIONSTARTANDEND, 1208, 1219, 1228
FILELICENCEEMPT, 1209, 1219, 1228
FILELICENCEEND, 1209, 1219, 1228
FILELICENCESTART, 1209, 1219, 1228
FILELICENCESTARTANDEND, 1209, 1219, 1228
FILENAMEEMPT, 1208, 1218, 1228
FILENAMEEND, 1208, 1218, 1228
FILENAMESTART, 1208, 1218, 1228
FILENAMESTARTANDEND, 1208, 1218, 1228
FILESOURCEEMPT, 1208, 1219, 1228
FILESOURCEEND, 1208, 1219, 1228
FILESOURCESTART, 1208, 1218, 1228
FILESOURCESTARTANDEND, 1208, 1219, 1228
FILESTODELETEEND, 1216
FILESTODELETESTART, 1216
FILESTOMAKEEND, 1216
FILESTOMAKESTART, 1216
FIRSTAXISDIRECTIONATT, 1207
FROMATT, 1214
GENERALELEMENTSEND, 1210, 1220, 1229
GENERALELEMENTSSTART, 1210, 1220, 1229
GENERALEND, 1215, 1224
GENERALSTART, 1215, 1224
GENERALSTATUSEND, 1226
GENERALSTATUSSTART, 1226
GENERALSUBSTATUSEND, 1226
GENERALSUBSTATUSSTART, 1226
GREATERTHAN, 1205, 1213, 1223
GROUPWEIGHTATT, 1214
HEADEREND, 1208, 1218, 1224
HEADERSTART, 1208, 1218, 1224
HORIZONATT, 1208
IDATT, 1213, 1223, 1232
IDENTITYMATRIXEND, 1213, 1223, 1232
IDENTITYMATRIXSTART, 1213, 1223, 1232
IDXATT, 1210, 1214, 1224
IDXEND, 1226
IDXONEATT, 1206
IDXSTART, 1226
IDXTWOATT, 1206
IFEND, 1212, 1221, 1231
IFSTART, 1211, 1221, 1231

INCLUDEDIAGONALATT, 1213, 1223, 1232
INCRATT, 1209, 1218, 1224
INDEXESEND, 1210, 1220, 1229
INDEXESSTART, 1210, 1220, 1229
INITIALBASISSTATUSSEND, 1217
INITIALBASISSTATUSSTART, 1217
INITIALCONSTRAINTVALUESSEND, 1218
INITIALCONSTRAINTVALUESSTART, 1218
INITIALDUALVALUESSEND, 1218
INITIALDUALVALUESSTART, 1218
INITIALOBJECTIVEBOUNDSSEND, 1218
INITIALOBJECTIVEBOUNDSSTART, 1218
INITIALOBJECTIVEVALUESSEND, 1217
INITIALOBJECTIVEVALUESSTART, 1217
INITIALVARIABLEVALUESSEND, 1217
INITIALVARIABLEVALUESSTART, 1217
INITIALVARIABLEVALUESSTRINGEND, 1217
INITIALVARIABLEVALUESSTRINGSTART, 1217
INPUTDIRECTORIESTOMOVEEND, 1216
INPUTDIRECTORIESTOMOVESTART, 1216
INPUTFILESTOMOVEEND, 1216
INPUTFILESTOMOVESTART, 1216
INSTANCEDATAEND, 1205
INSTANCEDATASTARTEND, 1206
INSTANCELOCATIONEND, 1215
INSTANCELOCATIONSTART, 1215
INSTANCENAMEEND, 1215, 1226
INSTANCENAMESTART, 1215, 1226
INTEGER, 1205, 1213, 1223
INTEGERVARIABLEBRANCHINGWEIGHTSEND, 1217
INTEGERVARIABLEBRANCHINGWEIGHTSSTART, 1217
INTERSECTIONCONEEND, 1206
INTERSECTIONCONESTART, 1206
INTERVALEND, 1208
INTERVALSTART, 1208
ISFREEEND, 1217, 1226
ISFREESTART, 1217, 1226
ITEMEMPTY, 1209, 1218, 1225
ITEMEND, 1209, 1218, 1225
ITEMSTART, 1209, 1218, 1225
ITEMSTARTANDEND, 1209, 1218, 1225
ITEMTEXT, 1205, 1213, 1223
JOBEND, 1215, 1225
JOBIDEND, 1215, 1226
JOBIDSTART, 1215, 1226
JOBSTART, 1215, 1225
LBCONEIDXATT, 1207
LBDUALVALUEATT, 1215
LBMATRIXIDXATT, 1207
LBVALUEATT, 1215
LICENSEEND, 1215
LICENSESTART, 1215
LINEARELEMENTSEND, 1210, 1220, 1229
LINEARELEMENTSSTART, 1210, 1220, 1229
LNEND, 1211, 1221, 1230
LNSTART, 1211, 1221, 1230
LOCATIONTYPEATT, 1215
MAKECOPYATT, 1214
MATRICESEND, 1206
MATRICESSTART, 1206
MATRIXCONEND, 1207, 1218, 1228
MATRIXCONSTART, 1207, 1218, 1228
MATRIXCONSTRAINTSEND, 1207
MATRIXCONSTRAINTSSTART, 1207
MATRIXDETERMINANTEND, 1212, 1222, 1231
MATRIXDETERMINANTSTART, 1212, 1222, 1231
MATRIXDIAGONALEND, 1212, 1222, 1232
MATRIXDIAGONALSTART, 1212, 1222, 1232
MATRIXDOTTIMESEND, 1212, 1222, 1232
MATRIXDOTTIMESSTART, 1212, 1222, 1232
MATRIXEND, 1209, 1219, 1228
MATRIXEXPRESSIONSEND, 1211, 1221, 1230
MATRIXEXPRESSIONSSTART, 1211, 1221, 1230
MATRIXIDXATT, 1207
MATRIXINVERSEEND, 1213, 1223, 1232
MATRIXINVERSESTART, 1213, 1223, 1232
MATRIXLOWERTRIANGLEEND, 1212, 1222, 1232
MATRIXLOWERTRIANGLESTART, 1212, 1222, 1232
MATRIXMERGEEND, 1213, 1222, 1232
MATRIXMERGESTART, 1212, 1222, 1232
MATRIXMINUSEND, 1213, 1222, 1232
MATRIXMINUSSTART, 1213, 1222, 1232
MATRIXNEGATEEND, 1213, 1222, 1232
MATRIXNEGATESTART, 1213, 1222, 1232
MATRIXOBJECTIVESEND, 1207
MATRIXOBJECTIVESSTART, 1207
MATRIXOBJEND, 1207, 1218, 1228
MATRIXOBJSTART, 1207, 1218, 1228
MATRIXPLUSEND, 1213, 1222, 1232
MATRIXPLUSSTART, 1213, 1222, 1232
MATRIXPRODUCTEND, 1213, 1223, 1232
MATRIXPRODUCTSTART, 1213, 1223, 1232
MATRIXPROGRAMMINGEND, 1207
MATRIXPROGRAMMINGSTART, 1207
MATRIXREFERENCEEND, 1213, 1223, 1232
MATRIXREFERENCESTART, 1213, 1223, 1232
MATRIXSCALARTIMESEND, 1213, 1223, 1232
MATRIXSCALARTIMESSTART, 1213, 1223, 1232
MATRIXSTART, 1209, 1219, 1228
MATRIXSUBMATRIXATEND, 1213, 1223, 1232
MATRIXSUBMATRIXATSTART, 1213, 1223, 1232
MATRIXTERMEND, 1211, 1221, 1230
MATRIXTERMSTART, 1211, 1221, 1230
MATRIXTIMESEND, 1213, 1223, 1232
MATRIXTIMESSTART, 1213, 1222, 1232

MATRIXTOSCALAREND, [1212](#), [1222](#), [1232](#)
MATRIXTOSCALARSTART, [1212](#), [1222](#), [1231](#)
MATRIXTRACEEND, [1212](#), [1222](#), [1231](#)
MATRIXTRACESTART, [1212](#), [1222](#), [1231](#)
MATRIXTRANPOSEEND, [1213](#), [1223](#), [1232](#)
MATRIXTRANPOSESTART, [1213](#), [1223](#), [1232](#)
MATRIXUPPERTRIANGLEEND, [1212](#), [1222](#), [1232](#)
MATRIXUPPERTRIANGLESTART, [1212](#), [1222](#), [1232](#)
MATRIXVAREND, [1207](#), [1218](#), [1228](#)
MATRIXVARIABLESEND, [1207](#)
MATRIXVARIABLESSTART, [1207](#)
MATRIXVARSTART, [1207](#), [1218](#), [1228](#)
MAXEND, [1212](#), [1222](#), [1231](#)
MAXSTART, [1212](#), [1222](#), [1231](#)
MAXTIMEEND, [1216](#)
MAXTimestart, [1216](#)
MESSAGEEND, [1226](#)
MESSAGESTART, [1226](#)
MINCPUNUMBEREND, [1216](#)
MINCPUNUMBERSTART, [1216](#)
MINCPUSPEEDEND, [1216](#)
MINCPUSPEEDSTART, [1216](#)
MINDISKSPACEEND, [1216](#)
MINDISKSPACESTART, [1216](#)
MINEND, [1212](#), [1222](#), [1231](#)
MINMEMORYEND, [1216](#)
MINMEMORYSTART, [1216](#)
MINSTART, [1212](#), [1222](#), [1231](#)
MINUSEND, [1211](#), [1221](#), [1230](#)
MINUSSTART, [1211](#), [1221](#), [1230](#)
MULTATT, [1209](#), [1218](#), [1224](#)
NAMEATT, [1209](#), [1214](#), [1224](#)
NEGATEEND, [1211](#), [1221](#), [1231](#)
NEGATESTART, [1211](#), [1221](#), [1231](#)
NEGATIVEPATTERNATT, [1209](#), [1219](#), [1229](#)
NLEND, [1211](#), [1221](#), [1230](#)
NLSTART, [1211](#), [1221](#), [1230](#)
NONLINEAREXPRESSIONSEND, [1211](#), [1221](#), [1230](#)
NONLINEAREXPRESSIONSSTART, [1211](#), [1221](#), [1230](#)
NONNEGATIVECONEEND, [1206](#)
NONNEGATIVECONESTART, [1206](#)
NONPOSITIVECONEEND, [1206](#)
NONPOSITIVECONESTART, [1206](#)
NONZEROSSEND, [1210](#), [1220](#), [1229](#)
NONZEROSSTART, [1210](#), [1220](#), [1229](#)
NORMSCALEFACTORATT, [1207](#)
NUMBEREND, [1212](#), [1222](#), [1231](#)
NUMBEROFBLOCKSATT, [1209](#), [1219](#), [1229](#)
NUMBEROFCOLUMNSATT, [1209](#), [1219](#), [1229](#)
NUMBEROFCONATT, [1214](#), [1223](#)
NUMBEROFCONESATT, [1206](#)
NUMBEROFCONSTRAINTSATT, [1208](#), [1214](#), [1223](#)
NUMBEROFELATT, [1209](#), [1214](#), [1223](#)
NUMBEROFENUMERATIONSATT, [1213](#), [1223](#)
NUMBEROFEXPR, [1211](#), [1221](#), [1230](#)
NUMBEROFIDXATT, [1223](#)
NUMBEROFITEMSATT, [1214](#), [1223](#)
NUMBEROFJOBIDSATT, [1213](#)
NUMBEROFMATRICESATT, [1206](#)
NUMBEROFMATRIXCONATT, [1207](#)
NUMBEROFMATRIXOBJATT, [1207](#)
NUMBEROFMATRIXTERMSATT, [1211](#), [1221](#), [1230](#)
NUMBEROFMATRIXVARATT, [1207](#)
NUMBEROFNONLINEAREXPRESSIONS, [1211](#), [1221](#), [1230](#)
NUMBEROFOBJATT, [1214](#), [1223](#)
NUMBEROFOBJECTIVESATT, [1208](#), [1214](#), [1223](#)
NUMBEROFOTHERCONSTRAINTOPTIONSATT, [1214](#)
NUMBEROFOTHERCONSTRAINTRESULTSATT, [1223](#)
NUMBEROFOTHEROBJECTIVEOPTIONSATT, [1214](#)
NUMBEROFOTHEROBJECTIVERESULTSATT, [1223](#)
NUMBEROFOTHEROPTIONSATT, [1213](#)
NUMBEROFOTHERRESULTSATT, [1223](#)
NUMBEROFOTHERSOLUTIONRESULTSATT, [1224](#)
NUMBEROFOTHERVARIABLEOPTIONSATT, [1214](#)
NUMBEROFOTHERVARIABLERESULTSATT, [1224](#)
NUMBEROFPATHPAIRSATT, [1214](#)
NUMBEROFPATHSATT, [1214](#)
NUMBEROFPROCESSESATT, [1214](#)
NUMBEROFQTERMSATT, [1206](#)
NUMBEROFROWSATT, [1209](#), [1219](#), [1229](#)
NUMBEROFSOLUTIONSATT, [1224](#)
NUMBEROFSOLVEROPTIONSATT, [1214](#)
NUMBEROFSOLVEROUTPUTSATT, [1224](#)
NUMBEROFSOSATT, [1214](#)
NUMBEROFSTAGESATT, [1208](#)
NUMBEROFSUBSTATUSESATT, [1224](#)
NUMBEROFTIMESATT, [1224](#)
NUMBEROFVALUESATT, [1209](#), [1219](#), [1229](#)
NUMBEROFVARATT, [1214](#), [1224](#)
NUMBEROFVARIABLESATT, [1208](#), [1214](#), [1224](#)
NUMBEROFVARIDXATT, [1209](#), [1219](#), [1224](#)
NUMBERSTART, [1212](#), [1222](#), [1231](#)
OBJECTIVESEND, [1208](#), [1217](#), [1226](#)
OBJECTIVESSTART, [1208](#), [1217](#), [1226](#)
OBJEND, [1208](#), [1218](#), [1226](#)
OBJREFERENCEELEMENTSEND, [1210](#), [1220](#), [1230](#)
OBJREFERENCEELEMENTSSTART, [1210](#), [1220](#), [1230](#)
OBJREFERENCEMATRIXIDXATT, [1207](#)

OBJSTART, [1208](#), [1218](#), [1226](#)
OBJTYPEATT, [1214](#), [1224](#)
OPTIMIZATIONEND, [1215](#), [1225](#)
OPTIMIZATIONSOLUTIONSTATUSSEND, [1226](#)
OPTIMIZATIONSOLUTIONSTATUSSTART, [1226](#)
OPTIMIZATIONSOLUTIONSUBSTATUSSEND, [1226](#)
OPTIMIZATIONSOLUTIONSUBSTATUSSTART, [1226](#)
OPTIMIZATIONSTART, [1215](#), [1225](#)
ORDERCONEIDXATT, [1207](#)
ORTHANTCONEEND, [1206](#)
ORTHANTCONESTART, [1206](#)
OSILEND, [1205](#)
OSOLATTRIBUTETEXT, [1213](#)
OSOLEND, [1213](#)
OSOLSTART, [1213](#)
OSOLSTARTEMPT, [1213](#)
OSRLATTRIBUTETEXT, [1223](#)
OSRLEND, [1223](#)
OSRLSTART, [1223](#)
OSRLSTARTEMPT, [1223](#)
OTHEREND, [1216](#), [1226](#)
OTHEROPTIONSSEND, [1216](#)
OTHEROPTIONSSTART, [1216](#)
OTHERRESULTSEND, [1226](#)
OTHERRESULTSSTART, [1226](#)
OTHERSOLUTIONRESULTEND, [1226](#)
OTHERSOLUTIONRESULTSEND, [1226](#)
OTHERSOLUTIONRESULTSSTART, [1226](#)
OTHERSOLUTIONRESULTSTART, [1226](#)
OTHERSOLVEROUTPUTEND, [1226](#)
OTHERSOLVEROUTPUTSTART, [1226](#)
OTHERSTART, [1216](#), [1226](#)
OUTPUTDIRECTORIESTOMOVEEND, [1217](#)
OUTPUTDIRECTORIESTOMOVESTART, [1217](#)
OUTPUTFILESTOMOVEEND, [1217](#)
OUTPUTFILESTOMOVESTART, [1217](#)
PASSWORDEND, [1215](#)
PASSWORDSTART, [1215](#)
PATHEND, [1216](#)
PATHPAIREND, [1216](#)
PATHPAIRSTART, [1216](#)
PATHSTART, [1216](#)
PATTERNELEMENTSEND, [1210](#), [1220](#), [1230](#)
PATTERNELEMENTSSTART, [1210](#), [1220](#), [1230](#)
PIEND, [1212](#), [1222](#), [1231](#)
PISTART, [1212](#), [1222](#), [1231](#)
PLUSEND, [1211](#), [1221](#), [1230](#)
PLUSSTART, [1211](#), [1221](#), [1230](#)
POLARCONEEND, [1206](#)
POLARCONESTART, [1206](#)
POLYHEDRALCONEEND, [1206](#)
POLYHEDRALCONESTART, [1206](#)
POWEREND, [1211](#), [1221](#), [1230](#)
POWERSTART, [1211](#), [1221](#), [1230](#)
PROCESSEND, [1217](#)
PROCESSESTOKILLEND, [1217](#)
PROCESSESTOKILLSTART, [1217](#)
PROCESSSTART, [1217](#)
PRODUCTCONEEND, [1206](#)
PRODUCTCONESTART, [1206](#)
PRODUCTEND, [1211](#), [1221](#), [1231](#)
PRODUCTSTART, [1211](#), [1221](#), [1231](#)
QTERMEND, [1206](#)
QTERMSTART, [1206](#)
QUADRATICCOEFFICIENTSEND, [1206](#)
QUADRATICCOEFFICIENTSSTART, [1206](#)
QUADRATICCONEEND, [1206](#)
QUADRATICCONESTART, [1206](#)
QUOTE, [1205](#), [1213](#), [1223](#)
REFERENCEMATRIXIDXATT, [1207](#)
REQUESTEDSTARTTIMEEND, [1216](#)
REQUESTEDSTARTTIMESTART, [1216](#)
REQUIREDDIRECTORIESEND, [1216](#)
REQUIREDDIRECTORIESSTART, [1216](#)
REQUIREDFILESSEND, [1216](#)
REQUIREDFILESSTART, [1216](#)
ROTATEDQUADRATICCONEEND, [1206](#)
ROTATEDQUADRATICCONESTART, [1206](#)
ROWMAJORATT, [1211](#), [1220](#), [1230](#)
ROWOFFSETSEND, [1211](#), [1220](#), [1230](#)
ROWOFFSETSTART, [1211](#), [1220](#), [1230](#)
SCALARMULTIPLIERATT, [1210](#), [1220](#), [1229](#)
SCHEDULEDSTARTTIMEEND, [1227](#)
SCHEDULEDSTARTTIMESTART, [1227](#)
SECONDAXISDIRECTIONATT, [1207](#)
SEMIDEFINITECONEEND, [1206](#)
SEMIDEFINITECONESTART, [1206](#)
SEMIDEFINITENESSATT, [1207](#)
SERVICEEND, [1215](#), [1225](#)
SERVICENAMEEND, [1215](#), [1227](#)
SERVICENAMESTART, [1215](#), [1227](#)
SERVICESTART, [1215](#), [1225](#)
SERVICETYPEEND, [1216](#)
SERVICETYPESTART, [1216](#)
SERVICEURIEND, [1215](#), [1227](#)
SERVICEURISTART, [1215](#), [1227](#)
SERVICEUTILIZATIONEND, [1227](#)
SERVICEUTILIZATIONSTART, [1227](#)
SHAPEATT, [1209](#), [1219](#), [1229](#)
SINEND, [1212](#), [1221](#), [1231](#)
SINSTART, [1212](#), [1221](#), [1231](#)
SIZEOFATT, [1209](#), [1218](#), [1224](#)
SOLUTIONEND, [1227](#)
SOLUTIONSTART, [1227](#)
SOLVERATT, [1215](#)
SOLVERINVOKEDEND, [1227](#)
SOLVERINVOKEDSTART, [1227](#)

SOLVEROPTIONEND, 1218
SOLVEROPTIONSSEND, 1218
SOLVEROPTIONSSTART, 1218
SOLVEROPTIONSTART, 1218
SOLVEROUTPUTEND, 1227
SOLVEROUTPUTSTART, 1227
SOLVERTOINVOKEEND, 1215
SOLVERTOINVOKESTART, 1215
SOSEND, 1217
SOSIDXATT, 1214
SOSSTART, 1217
SOSVARIABLEBRANCHINGWEIGHTSEND, 1217
SOSVARIABLEBRANCHINGWEIGHTSSTART, 1217
SQRTEND, 1211, 1221, 1231
SQRTSTART, 1211, 1221, 1230
SQUAREEND, 1212, 1221, 1231
SQUARESTART, 1212, 1221, 1231
STAGEEND, 1208
STAGESEND, 1208
STAGESSTART, 1207
STAGESTART, 1208
STARTATT, 1208
STARTIDXATT, 1208
STARTVECTOREND, 1210, 1220, 1229
STARTVECTORSTART, 1210, 1220, 1229
STATUSEND, 1227
STATUSSTART, 1227
SUBMITTIMEEND, 1227
SUBMITTIMESTART, 1227
SUBSTATUSEND, 1227
SUBSTATUSSTART, 1227
SUMEND, 1211, 1221, 1231
SUMSTART, 1211, 1221, 1231
SUPERBASICEND, 1217, 1227
SUPERBASICSTART, 1217, 1227
SYMMETRYATT, 1209, 1219, 1229
SYSTEMEND, 1215, 1225
SYSTEMINFORMATIONEND, 1227
SYSTEMINFORMATIONSTART, 1227
SYSTEMSTART, 1215, 1225
TARGETMATRIXFIRSTCOLATT, 1210, 1219, 1229
TARGETMATRIXFIRSTROWATT, 1210, 1219, 1229
TARGETOBJECTIVEIDXATT, 1224
TARGETOBJECTIVENAMEATT, 1224
TEMPLATMATRIXIDXATT, 1207
TIMEDOMAINEND, 1207
TIMEDOMAINSTART, 1207
TIMEEND, 1227
TIMESEND, 1212, 1222, 1231
TIMESERVICESTARTEDEND, 1227
TIMESERVICESTARTEDSTART, 1227
TIMESSTART, 1212, 1222, 1231
TIMESTAMPEND, 1227

TIMESTAMPSTART, 1227
TIMESTART, 1227
TIMINGINFORMATIONEND, 1227
TIMINGINFORMATIONSTART, 1227
TOATT, 1214
TOTALJOBSSOFAREND, 1227
TOTALJOBSSOFARSTART, 1227
TRANSFORMATIONEND, 1210, 1220, 1230
TRANSFORMATIONSTART, 1210, 1220, 1230
TRANSPORTTYPEATT, 1215
TWOQUOTES, 1205, 1213, 1223
TYPEATT, 1209, 1214, 1224
UBCONEIDXATT, 1207
UBDUALVALUEATT, 1215
UBMATRIXIDXATT, 1207
UBVALUEATT, 1215
UNITATT, 1214, 1224
UNKNOWNEND, 1217, 1227
UNKNOWNSTART, 1217, 1227
USEDPCPUNUMBEREND, 1227
USEDPCPUNUMBERSTART, 1227
USEDPCPUSPEEDEND, 1227
USEDPCPUSPEEDSTART, 1227
USEDISKSPACEEND, 1227
USEDISKSPACESTART, 1227
USEDMEMORYEND, 1228
USEDMEMORYSTART, 1228
USERNAMEEND, 1215
USERNAMESTART, 1215
VALUEATT, 1206, 1214, 1224
VALUESEND, 1210, 1220, 1229
VALUESSTART, 1210, 1220, 1229
VALUESSTRINGEND, 1228
VALUESSTRINGSTART, 1228
VALUETYPEATT, 1210, 1220, 1230
VAREND, 1208, 1217, 1228
VARIABLEEND, 1212, 1222, 1231
VARIABLESEND, 1208, 1217, 1228
VARIABLESSTART, 1208, 1217, 1228
VARIABLESTART, 1212, 1221, 1231
VARIDXEND, 1210, 1220, 1228
VARIDXSTART, 1210, 1220, 1228
VARREFERENCEELEMENTSEND, 1210, 1220, 1229
VARREFERENCEELEMENTSSTART, 1210, 1220, 1229
VARREFERENCEMATRIXIDXATT, 1207
VARSTART, 1208, 1217, 1228
VARTYPEATT, 1207, 1214, 1224
WEIGHTATT, 1215
WEIGHTEDOBJECTIVESATT, 1224
OSParseosrl.tab.hpp
ABSEND, 1285, 1294, 1304
ABSSTART, 1285, 1294, 1304

ACTUALSTARTTIMEEND, 1298
ACTUALSTARTTIMESTART, 1298
ALLDIFFEND, 1285, 1295, 1304
ALLDIFFSTART, 1285, 1295, 1304
ATEQUALITYEND, 1290, 1298
ATEQUALITYSTART, 1290, 1298
ATLOWEREND, 1290, 1298
ATLOWERSTART, 1290, 1298
ATTRIBUTETEXT, 1278, 1286, 1296
ATUPPEREND, 1290, 1298
ATUPPERSTART, 1290, 1298
AVAILABLECPUNUMBEREND, 1298
AVAILABLECPUNUMBERSTART, 1298
AVAILABLECPUSPEEDEND, 1298
AVAILABLECPUSPEEDSTART, 1298
AVAILABLEDISKSPACEEND, 1298
AVAILABLEDISKSPACESTART, 1298
AVAILABLEMEMORYEND, 1298
AVAILABLEMEMORYSTART, 1298
AXISDIRECTIONATT, 1279
BASE64END, 1282, 1291, 1298
BASE64START, 1282, 1291, 1298
BASEMATRIXEND, 1282, 1292, 1301
BASEMATRIXENDCOLATT, 1283, 1292, 1302
BASEMATRIXENDROWATT, 1283, 1292, 1302
BASEMATRIXIDXATT, 1282, 1292, 1302
BASEMATRIXSTART, 1282, 1292, 1301
BASEMATRIXSTARTCOLATT, 1283, 1292, 1302
BASEMATRIXSTARTROWATT, 1283, 1292, 1302
BASETRANSPOSEATT, 1283, 1292, 1302
BASICEND, 1290, 1298
BASICSTART, 1290, 1298
BASISSTATUSEND, 1298
BASISSTATUSSTART, 1298
BASSTATUSEND, 1298
BASSTATUSSTART, 1298
BLOCKCOLIDXATT, 1284, 1293, 1303
BLOCKEND, 1282, 1292, 1301
BLOCKROWIDXATT, 1284, 1293, 1303
BLOCKSEND, 1282, 1292, 1301
BLOCKSSTART, 1282, 1292, 1301
BLOCKSTART, 1282, 1292, 1301
CATEGORYATT, 1287, 1297
COEFATT, 1282, 1292, 1302
COLOFFSETSEND, 1283, 1293, 1303
COLOFFSETSTART, 1283, 1293, 1303
COMPONENTSEND, 1279
COMPONENTSSTART, 1279
CONEND, 1281, 1291, 1298
CONESEND, 1279
CONESSTART, 1279
CONREFERENCEELEMENTSEND, 1283, 1293, 1302
CONREFERENCEELEMENTSSTART, 1283, 1293, 1302
CONREFERENCEMATRIXIDXATT, 1280
CONSTANTATT, 1282, 1292, 1302
CONSTANTELEMENTSEND, 1283, 1293, 1302
CONSTANTELEMENTSSTART, 1283, 1293, 1302
CONSTANTMATRIXIDXATT, 1280
CONSTART, 1281, 1291, 1298
CONSTRAINTSEND, 1281, 1291, 1298
CONSTRAINTSSTART, 1281, 1290, 1298
CONTACTEND, 1288
CONTACTSTART, 1288
CONTYPEATT, 1287, 1297
COSEND, 1284, 1294, 1304
COSSTART, 1284, 1294, 1304
CURRENTJOBCOUNTEND, 1298
CURRENTJOBCOUNTSTART, 1298
CURRENTSTATEEND, 1298
CURRENTSTATESTART, 1298
DEPENDENCIESEND, 1289
DEPENDENCIESSTART, 1289
DESCRIPTIONATT, 1287, 1297
DIRECTIONEND, 1279
DIRECTIONSTART, 1279
DIRECTORIESTODELETEEND, 1289
DIRECTORIESTODELETESTART, 1289
DIRECTORIESTOMAKEEND, 1289
DIRECTORIESTOMAKESTART, 1289
DISTORTIONMATRIXIDXATT, 1279
DIVIDEEND, 1284, 1294, 1303
DIVIDESTART, 1284, 1294, 1303
DOUBLE, 1278, 1286, 1296
DUALCONEEND, 1279
DUALCONESTART, 1279
DUALVALUESEND, 1298
DUALVALUESSTART, 1298
DUMMY, 1284, 1293, 1303
EEND, 1285, 1295, 1304
ELEMENTSEND, 1283, 1293, 1302
ELEMENTSSTART, 1283, 1292, 1302
ELEMENTTEXT, 1278, 1286, 1296
EEND, 1282, 1291, 1299
ELSTART, 1282, 1291, 1298
EMPTYBASETRANSPOSEATT, 1283, 1292, 1302
EMPTYCATEGORYATT, 1287, 1297
EMPTYCONTYPEATT, 1287, 1297
EMPTYDESCRIPTIONATT, 1287, 1297
EMPTYENUMTYPEATT, 1287, 1297
EMPTYIDATT, 1278
EMPTYINCLUDEDIAGONALATT, 1286, 1296, 1305
EMPTYLB DUALVALUEATT, 1287
EMPTYLBVALUEATT, 1287
EMPTYNAMEATT, 1282, 1287, 1297
EMPTYNEGATIVEPATTERNATT, 1282, 1292, 1302

EMPTYOBJTYPEATT, 1287, 1297
EMPTYROWMAJORATT, 1283, 1293, 1303
EMPTYSEMIDEFINITENESSATT, 1280
EMPTYSHAPEATT, 1282, 1292, 1301
EMPTYSOLVERATT, 1288
EMPTYSYMMETRYATT, 1282, 1292, 1301
EMPTYTARGETOBJECTIVENAMEATT, 1297
EMPTYTYPEATT, 1282, 1287, 1297
EMPTYUBDUALVALUEATT, 1288
EMPTYUBVALUEATT, 1287
EMPTYUNITATT, 1287, 1297
EMPTYVALUEATT, 1287, 1297
EMPTYVVARTYPEATT, 1287, 1297
EMPTYWEIGHTATT, 1288
EMPTYWEIGHTEDOBJECTIVESATT, 1297
ENDOFELEMENT, 1278, 1286, 1296
ENDTIMEEND, 1299
ENDTIMESTART, 1299
ENUMERATIONEND, 1282, 1291, 1299
ENUMERATIONSTART, 1281, 1291, 1299
ENUMTYPEATT, 1287, 1297
ERFEND, 1285, 1294, 1304
ERFSTART, 1285, 1294, 1304
ESTART, 1285, 1295, 1304
EXPEND, 1284, 1294, 1304
EXPEND, 1284, 1294, 1303
EXPRSTART, 1284, 1294, 1303
EXPSTART, 1284, 1294, 1304
FACTORSEND, 1279
FACTORSTART, 1279
FILECREATOREMPTY, 1281, 1292, 1301
FILECREATOREND, 1281, 1292, 1301
FILECREATORSTART, 1281, 1292, 1301
FILECREATORSTARTANDEND, 1281, 1292, 1301
FILEDESCRIPTIONEMPTY, 1281, 1292, 1301
FILEDESCRIPTIONEND, 1281, 1291, 1301
FILEDESCRIPTIONSTART, 1281, 1291, 1301
FILEDESCRIPTIONSTARTANDEND, 1281, 1292, 1301
FILELICENCEEMPTY, 1281, 1292, 1301
FILELICENCEEND, 1281, 1292, 1301
FILELICENCESTART, 1281, 1292, 1301
FILELICENCESTARTANDEND, 1281, 1292, 1301
FILENAMEEMPTY, 1281, 1291, 1301
FILENAMEEND, 1281, 1291, 1301
FILENAMESTART, 1281, 1291, 1301
FILENAMESTARTANDEND, 1281, 1291, 1301
FILESOURCEEMPTY, 1281, 1291, 1301
FILESOURCEEND, 1281, 1291, 1301
FILESOURCESTART, 1281, 1291, 1301
FILESOURCESTARTANDEND, 1281, 1291, 1301
FILESTODELETEEND, 1289
FILESTODELETESTART, 1289
FILESTOMAKEEND, 1289
FILESTOMAKESTART, 1289
FIRSTAXISDIRECTIONATT, 1280
FROMATT, 1286
GENERALELEMENTSEND, 1283, 1293, 1302
GENERALELEMENTSSTART, 1283, 1293, 1302
GENERALEND, 1288, 1297
GENERALSTART, 1288, 1297
GENERALSTATUSEND, 1299
GENERALSTATUSSTART, 1299
GENERALSUBSTATUSEND, 1299
GENERALSUBSTATUSSTART, 1299
GREATERTHAN, 1278, 1286, 1296
GROUPWEIGHTATT, 1287
HEADEREND, 1281, 1291, 1297
HEADERSTART, 1281, 1291, 1297
HORIZONATT, 1280
IDATT, 1286, 1296, 1305
IDENTITYMATRIXEND, 1286, 1296, 1305
IDENTITYMATRIXSTART, 1286, 1296, 1305
IDXATT, 1282, 1287, 1297
IDXEND, 1299
IDXONEATT, 1278
IDXSTART, 1299
IDXTWOATT, 1278
IFEND, 1284, 1294, 1304
IFSTART, 1284, 1294, 1304
INCLUDEDIAGONALATT, 1286, 1296, 1305
INCRATT, 1282, 1291, 1297
INDEXESEND, 1283, 1293, 1302
INDEXESSTART, 1283, 1293, 1302
INITIALBASISSTATUSEND, 1290
INITIALBASISSTATUSSTART, 1290
INITIALCONSTRAINTVALUESEND, 1291
INITIALCONSTRAINTVALUESSTART, 1291
INITIALDUALVALUESEND, 1291
INITIALDUALVALUESSTART, 1291
INITIALOBJECTIVEBOUNSEND, 1290
INITIALOBJECTIVEBOUNDSSTART, 1290
INITIALOBJECTIVEVALUESEND, 1290
INITIALOBJECTIVEVALUESSTART, 1290
INITIALVARIABLEVALUESEND, 1290
INITIALVARIABLEVALUESSTART, 1290
INITIALVARIABLEVALUESSTRINGEND, 1290
INITIALVARIABLEVALUESSTRINGSTART, 1290
INPUTDIRECTORIESTOMOVEEND, 1289
INPUTDIRECTORIESTOMOVESTART, 1289
INPUTFILESTOMOVEEND, 1289
INPUTFILESTOMOVESTART, 1289
INSTANCEDATAEND, 1278
INSTANCEDATASTARTEND, 1278
INSTANCELOCATIONEND, 1288
INSTANCELOCATIONSTART, 1288
INSTANCENAMEEND, 1288, 1299
INSTANCENAMESTART, 1288, 1299

- INTEGER, 1278, 1286, 1296
INTEGervARIABLEBRANCHINGWEIGHTSEND, 1290
INTEGervARIABLEBRANCHINGWEIGHTSSTART, 1290
INTERSECTIONCONEEND, 1279
INTERSECTIONCONESTART, 1279
INTERVALEND, 1281
INTERVALSTART, 1281
ISFREEEND, 1290, 1299
ISFREESTART, 1290, 1299
ITEMEMPTY, 1282, 1291, 1298
ITEMEND, 1282, 1291, 1298
ITEMSTART, 1282, 1291, 1298
ITEMSTARTANDEND, 1282, 1291, 1298
ITEMTEXT, 1278, 1286, 1296
JOBEND, 1288, 1298
JOBIDEND, 1288, 1299
JOBIDSTART, 1288, 1299
JOBSTART, 1288, 1297
LBCONEIDXATT, 1280
LBDUALVALUEATT, 1288
LBMATRIXIDXATT, 1280
LBVALUEATT, 1287
LICENSEEND, 1288
LICENSESTART, 1288
LINEARELEMENTSEND, 1283, 1293, 1302
LINEARELEMENTSSTART, 1283, 1293, 1302
LNEND, 1284, 1294, 1303
LNSTART, 1284, 1294, 1303
LOCATIONTYPEATT, 1288
MAKECOPYATT, 1286
MATRICESEND, 1279
MATRICESSTART, 1279
MATRIXCONEND, 1280, 1291, 1301
MATRIXCONSTART, 1280, 1291, 1301
MATRIXCONSTRAINTSEND, 1280
MATRIXCONSTRAINTSSTART, 1280
MATRIXDETERMINANTEND, 1285, 1295, 1304
MATRIXDETERMINANTSTART, 1285, 1295, 1304
MATRIXDIAGONALEND, 1285, 1295, 1304
MATRIXDIAGONALSTART, 1285, 1295, 1304
MATRIXDOTTIMESEND, 1285, 1295, 1304
MATRIXDOTTIMESSTART, 1285, 1295, 1304
MATRIXEND, 1282, 1292, 1301
MATRIXEXPRESSIONSEND, 1284, 1294, 1303
MATRIXEXPRESSIONSSTART, 1284, 1294, 1303
MATRIXIDXATT, 1280
MATRIXINVERSEEND, 1286, 1296, 1305
MATRIXINVERSESTART, 1286, 1296, 1305
MATRIXLOWERTRIANGLEEND, 1285, 1295, 1305
MATRIXLOWERTRIANGLESTART, 1285, 1295, 1305
MATRIXMERGEEND, 1285, 1295, 1305
MATRIXMERGESTART, 1285, 1295, 1305
MATRIXMINUSEND, 1285, 1295, 1305
MATRIXMINUSSTART, 1285, 1295, 1305
MATRIXNEGATEEND, 1285, 1295, 1305
MATRIXNEGATESTART, 1285, 1295, 1305
MATRIXOBJECTIVESEND, 1280
MATRIXOBJECTIVESSTART, 1280
MATRIXOBJEND, 1280, 1291, 1301
MATRIXOBJSTART, 1280, 1291, 1301
MATRIXPLUSEND, 1286, 1295, 1305
MATRIXPLUSSTART, 1286, 1295, 1305
MATRIXPRODUCTEND, 1286, 1295, 1305
MATRIXPRODUCTSTART, 1286, 1295, 1305
MATRIXPROGRAMMINGEND, 1280
MATRIXPROGRAMMINGSTART, 1280
MATRIXREFERENCEEND, 1286, 1296, 1305
MATRIXREFERENCESTART, 1286, 1296, 1305
MATRIXSCALARTIMESEND, 1286, 1295, 1305
MATRIXSCALARTIMESSTART, 1286, 1295, 1305
MATRIXSTART, 1282, 1292, 1301
MATRIXSUBMATRIXATEND, 1286, 1296, 1305
MATRIXSUBMATRIXATSTART, 1286, 1296, 1305
MATRIXTERMEND, 1284, 1294, 1303
MATRIXTERMSTART, 1284, 1294, 1303
MATRIXTIMESEND, 1286, 1295, 1305
MATRIXTIMESSTART, 1286, 1295, 1305
MATRIXTOSCALAREND, 1285, 1295, 1304
MATRIXTOSCALARSTART, 1285, 1295, 1304
MATRIXTRACEEND, 1285, 1295, 1304
MATRIXTRACESTART, 1285, 1295, 1304
MATRIXTRANSPOSEEND, 1286, 1296, 1305
MATRIXTRANSPOSESTART, 1286, 1296, 1305
MATRIXUPPERTRIANGLEEND, 1285, 1295, 1305
MATRIXUPPERTRIANGLESTART, 1285, 1295, 1305
MATRIXVAREND, 1280, 1291, 1301
MATRIXVARIABLESEND, 1280
MATRIXVARIABLESSTART, 1280
MATRIXVARSTART, 1280, 1291, 1301
MAXEND, 1285, 1295, 1304
MAXSTART, 1285, 1295, 1304
MAXTIMEEND, 1289
MAXTIMESTART, 1289
MESSAGEEND, 1299
MESSAGESTART, 1299
MINCPUNUMBEREND, 1289
MINCPUNUMBERSTART, 1289
MINCPUSPEEDEND, 1289
MINCPUSPEEDSTART, 1289
MINDISKSPACEEND, 1289
MINDISKSPACESTART, 1289
MINEND, 1285, 1295, 1304
MINMEMORYEND, 1289
MINMEMORYSTART, 1289

MINSTART, 1285, 1295, 1304
MINUSEND, 1284, 1294, 1303
MINUSSTART, 1284, 1294, 1303
MULTATT, 1282, 1291, 1297
NAMEATT, 1282, 1287, 1297
NEGATEEND, 1284, 1294, 1304
NEGATESTART, 1284, 1294, 1304
NEGATIVEPATTERNATT, 1282, 1292, 1302
NLEND, 1284, 1293, 1303
NLSTART, 1284, 1293, 1303
NONLINEAREXPRESSIONSSEND, 1284, 1293, 1303
NONLINEAREXPRESSIONSSTART, 1284, 1293, 1303
NONNEGATIVECONEEND, 1279
NONNEGATIVECONESTART, 1279
NONPOSITIVECONEEND, 1279
NONPOSITIVECONESTART, 1279
NONZEROSSEND, 1283, 1293, 1302
NONZEROSSTART, 1283, 1293, 1302
NORMSCALEFACTORATT, 1279
NUMBEREND, 1285, 1295, 1304
NUMBEROFBLOCKSATT, 1282, 1292, 1302
NUMBEROFCOLUMNSATT, 1282, 1292, 1302
NUMBEROFCONATT, 1287, 1296
NUMBEROFCONESATT, 1279
NUMBEROFCONSTRAINTSATT, 1281, 1287, 1296
NUMBEROFELATT, 1282, 1287, 1296
NUMBEROFENUMERATIONSATT, 1286, 1296
NUMBEROFEXPR, 1284, 1294, 1303
NUMBEROFIDXATT, 1296
NUMBEROFITEMSATT, 1287, 1296
NUMBEROFJOBIDSATT, 1286
NUMBEROFMATRICESATT, 1279
NUMBEROFMATRIXCONATT, 1280
NUMBEROFMATRIXOBJATT, 1280
NUMBEROFMATRIXTERMSATT, 1284, 1294, 1303
NUMBEROFMATRIXVARATT, 1280
NUMBEROFNONLINEAREXPRESSIONS, 1284, 1293, 1303
NUMBEROFOBJATT, 1287, 1296
NUMBEROFOBJECTIVESATT, 1281, 1287, 1296
NUMBEROFOTHERCONSTRAINTOPTIONSATT, 1287
NUMBEROFOTHERCONSTRAINTRESULTSATT, 1296
NUMBEROFOTHEROBJECTIVEOPTIONSATT, 1287
NUMBEROFOTHEROBJECTIVERESULTSATT, 1296
NUMBEROFOTHEROPTIONSATT, 1286
NUMBEROFOTHERRESULTSATT, 1296
NUMBEROFOTHERSOLUTIONRESULTSATT, 1296
NUMBEROFOTHERVARIABLEOPTIONSATT, 1287
NUMBEROFOTHERVARIABLERESULTSATT, 1296
NUMBEROFPATHPAIRSATT, 1286
NUMBEROFPATHSATT, 1286
NUMBEROFPROCESSESATT, 1287
NUMBEROFQTERMSATT, 1279
NUMBEROFROWSATT, 1282, 1292, 1302
NUMBEROFSOLUTIONSATT, 1296
NUMBEROFSOLVEROPTIONSATT, 1287
NUMBEROFSOLVEROUTPUTSATT, 1296
NUMBEROFSOSATT, 1287
NUMBEROFSTAGESATT, 1280
NUMBEROFSUBSTATUSESATT, 1296
NUMBEROFTIMESATT, 1297
NUMBEROFVALUESATT, 1282, 1292, 1302
NUMBEROFVARATT, 1287, 1297
NUMBEROFVARIABLESATT, 1281, 1287, 1297
NUMBEROFVARIDXATT, 1282, 1292, 1297
NUMBERSTART, 1285, 1295, 1304
OBJECTIVESEND, 1281, 1290, 1299
OBJECTIVESSTART, 1281, 1290, 1299
OBJEND, 1281, 1290, 1299
OBJREFERENCEELEMENTSEND, 1283, 1293, 1302
OBJREFERENCEELEMENTSSTART, 1283, 1293, 1302
OBJREFERENCEMATRIXIDXATT, 1280
OBJSTART, 1281, 1290, 1299
OBJTYPEATT, 1287, 1297
OPTIMIZATIONEND, 1288, 1298
OPTIMIZATIONSOLUTIONSTATUSSEND, 1299
OPTIMIZATIONSOLUTIONSTATUSSTART, 1299
OPTIMIZATIONSOLUTIONSUBSTATUSSEND, 1299
OPTIMIZATIONSOLUTIONSUBSTATUSSTART, 1299
OPTIMIZATIONSTART, 1288, 1298
ORDERCONEIDXATT, 1280
ORTHANTCONEEND, 1279
ORTHANTCONESTART, 1279
OSILEND, 1278
OSOLATTRIBUTETEXT, 1286
OSOLEND, 1286
OSOLSTART, 1286
OSOLSTARTEMPT, 1286
OSRLATTRIBUTETEXT, 1296
OSRLEND, 1296
OSRLSTART, 1296
OSRLSTARTEMPT, 1296
OTHEREND, 1289, 1299
OTHEROPTIONSSEND, 1288
OTHEROPTIONSSTART, 1288
OTHERRESULTSSEND, 1299
OTHERRESULTSSTART, 1299
OTHERSOLUTIONRESULTEND, 1299
OTHERSOLUTIONRESULTSSEND, 1299

OTHERSOLUTIONRESULTSSTART, 1299
OTHERSOLUTIONRESULTSTART, 1299
OTHERSOLVEROUTPUTEND, 1299
OTHERSOLVEROUTPUTSTART, 1299
OTHERSTART, 1288, 1299
OUTPUTDIRECTORIESTOMOVEEND, 1289
OUTPUTDIRECTORIESTOMOVESTART, 1289
OUTPUTFILESTOMOVEEND, 1289
OUTPUTFILESTOMOVESTART, 1289
PASSWORDEND, 1288
PASSWORDSTART, 1288
PATHEND, 1289
PATHPAIREND, 1289
PATHPAIRSTART, 1289
PATHSTART, 1289
PATTERNELEMENTSEND, 1283, 1293, 1303
PATTERNELEMENTSSTART, 1283, 1293, 1303
PIEND, 1285, 1295, 1304
PISTART, 1285, 1295, 1304
PLUSEND, 1284, 1294, 1303
PLUSSTART, 1284, 1294, 1303
POLARCONEEND, 1279
POLARCONESTART, 1279
POLYHEDRALCONEEND, 1279
POLYHEDRALCONESTART, 1279
POWEREND, 1284, 1294, 1303
POWERSTART, 1284, 1294, 1303
PROCESSEND, 1290
PROCESSESTOKILLEND, 1290
PROCESSESTOKILLSTART, 1289
PROCESSSTART, 1290
PRODUCTCONEEND, 1279
PRODUCTCONESTART, 1279
PRODUCTEND, 1284, 1294, 1303
PRODUCTSTART, 1284, 1294, 1303
QTERMEND, 1279
QTERMSTART, 1279
QUADRATICCOEFFICIENTSEND, 1279
QUADRATICCOEFFICIENTSSTART, 1279
QUADRATICCONEEND, 1279
QUADRATICCONESTART, 1279
QUOTE, 1278, 1286, 1296
REFERENCEMATRIIXIDXATT, 1280
REQUESTEDSTARTTIMEEND, 1289
REQUESTEDSTARTTimestart, 1289
REQUIREDDIRECTORIESEND, 1289
REQUIREDDIRECTORIESSTART, 1289
REQUIREDFILESSEND, 1289
REQUIREDFILESSTART, 1289
ROTATEDQUADRATICCONEEND, 1279
ROTATEDQUADRATICCONESTART, 1279
ROWMAJORATT, 1283, 1293, 1303
ROWOFFSETSEND, 1283, 1293, 1303
ROWOFFSETSSTART, 1283, 1293, 1303
SCALARMULTIPLIERATT, 1283, 1292, 1302
SCHEDULEDSTARTTIMEEND, 1299
SCHEDULEDSTARTTimestart, 1299
SECONDAXISDIRECTIONATT, 1280
SEMIDEFINITECONEEND, 1279
SEMIDEFINITECONESTART, 1279
SEMIDEFINITENESSATT, 1280
SERVICEEND, 1288, 1297
SERVICENAMEEND, 1288, 1299
SERVICENAMESTART, 1288, 1299
SERVICESTART, 1288, 1297
SERVICETYPEEND, 1289
SERVICETYPESTART, 1289
SERVICEURIEND, 1288, 1300
SERVICEURISTART, 1288, 1299
SERVICEUTILIZATIONEND, 1300
SERVICEUTILIZATIONSTART, 1300
SHAPEATT, 1282, 1292, 1301
SINEND, 1285, 1294, 1304
SINSTART, 1285, 1294, 1304
SIZEOFATT, 1282, 1291, 1297
SOLUTIONEND, 1300
SOLUTIONSTART, 1300
SOLVERATT, 1288
SOLVERINVOKEDEND, 1300
SOLVERINVOKEDSTART, 1300
SOLVEROPTIONEND, 1291
SOLVEROPTIONSEND, 1291
SOLVEROPTIONSSTART, 1291
SOLVEROPTIONSTART, 1291
SOLVEROUTPUTEND, 1300
SOLVEROUTPUTSTART, 1300
SOLVERTOINVOKEEND, 1288
SOLVERTOINVOKESTART, 1288
SOSEND, 1290
SOSIDXATT, 1287
SOSSTART, 1290
SOSVARIABLEBRANCHINGWEIGHTSEND, 1290
SOSVARIABLEBRANCHINGWEIGHTSSTART, 1290
SQRTEND, 1284, 1294, 1303
SQRTSTART, 1284, 1294, 1303
SQUAREEND, 1284, 1294, 1304
SQUARESTART, 1284, 1294, 1304
STAGEEND, 1280
STAGESEND, 1280
STAGESSTART, 1280
STAGESTART, 1280
STARTATT, 1281
STARTIDXATT, 1281
STARTVECTOREND, 1283, 1293, 1302
STARTVECTORSTART, 1283, 1293, 1302
STATUSEND, 1300
STATUSSTART, 1300

- SUBMITTIMEEND, [1300](#)
- SUBMITTimestart, [1300](#)
- SUBSTATUSEND, [1300](#)
- SUBSTATUSSTART, [1300](#)
- SUMEND, [1284](#), [1294](#), [1303](#)
- SUMSTART, [1284](#), [1294](#), [1303](#)
- SUPERBASICEND, [1290](#), [1300](#)
- SUPERBASICSTART, [1290](#), [1300](#)
- SYMMETRYATT, [1282](#), [1292](#), [1302](#)
- SYSTEMEND, [1288](#), [1297](#)
- SYSTEMINFORMATIONEND, [1300](#)
- SYSTEMINFORMATIONSTART, [1300](#)
- SYSTEMSTART, [1288](#), [1297](#)
- TARGETMATRIXFIRSTCOLATT, [1282](#), [1292](#), [1302](#)
- TARGETMATRIXFIRSTROWATT, [1282](#), [1292](#), [1302](#)
- TARGETOBJECTIVEIDXATT, [1297](#)
- TARGETOBJECTIVENAMEATT, [1297](#)
- TEMPLATMATRIXIDXATT, [1280](#)
- TIMEDOMAINEND, [1280](#)
- TIMEDOMAINSTART, [1280](#)
- TIMEEND, [1300](#)
- TIMESEND, [1285](#), [1295](#), [1304](#)
- TIMESERVICESTARTEDEND, [1300](#)
- TIMESERVICESTARTEDSTART, [1300](#)
- TIMESSTART, [1285](#), [1295](#), [1304](#)
- TIMESTAMPEND, [1300](#)
- TIMESTAMPSTART, [1300](#)
- TIMESTART, [1300](#)
- TIMINGINFORMATIONEND, [1300](#)
- TIMINGINFORMATIONSTART, [1300](#)
- TOATT, [1286](#)
- TOTALJOBSSOFAREND, [1300](#)
- TOTALJOBSSOFARSTART, [1300](#)
- TRANSFORMATIONEND, [1283](#), [1293](#), [1303](#)
- TRANSFORMATIONSTART, [1283](#), [1293](#), [1303](#)
- TRANSPORTTYPEATT, [1288](#)
- TWOQUOTES, [1278](#), [1286](#), [1296](#)
- TYPEATT, [1282](#), [1287](#), [1297](#)
- UBCONEIDXATT, [1280](#)
- UBDUALVALUEATT, [1288](#)
- UBMATRIXIDXATT, [1280](#)
- UBVALUEATT, [1287](#)
- UNITATT, [1287](#), [1297](#)
- UNKNOWNEND, [1290](#), [1300](#)
- UNKNOWNSTART, [1290](#), [1300](#)
- USEDPCUNUMBEREND, [1300](#)
- USEDPCUNUMBERSTART, [1300](#)
- USEDPCUSPEEDEND, [1300](#)
- USEDPCUSPEEDSTART, [1300](#)
- USEDISKSPACEEND, [1300](#)
- USEDISKSPACESTART, [1300](#)
- USEDMEMORYEND, [1300](#)
- USEDMEMORYSTART, [1300](#)
- USERNAMEEND, [1288](#)
- USERNAMESTART, [1288](#)
- VALUEATT, [1278](#), [1287](#), [1297](#)
- VALUESEND, [1283](#), [1293](#), [1302](#)
- VALUESSTART, [1283](#), [1293](#), [1302](#)
- VALUESSTRINGEND, [1300](#)
- VALUESSTRINGSTART, [1300](#)
- VALUETYPEATT, [1283](#), [1293](#), [1302](#)
- VAREND, [1281](#), [1290](#), [1301](#)
- VARIABLEEND, [1285](#), [1294](#), [1304](#)
- VARIABLESEND, [1281](#), [1290](#), [1301](#)
- VARIABLESSTART, [1281](#), [1290](#), [1301](#)
- VARIABLESTART, [1285](#), [1294](#), [1304](#)
- VARIDXEND, [1283](#), [1293](#), [1301](#)
- VARIDXSTART, [1283](#), [1293](#), [1301](#)
- VARREFERENCEELEMENTSEND, [1283](#), [1293](#), [1302](#)
- VARREFERENCEELEMENTSSTART, [1283](#), [1293](#), [1302](#)
- VARREFERENCEMATRIXIDXATT, [1280](#)
- VARSTART, [1281](#), [1290](#), [1300](#)
- VARTYPEATT, [1280](#), [1287](#), [1297](#)
- WEIGHTATT, [1288](#)
- WEIGHTEDOBJECTIVESATT, [1297](#)
- OSRLATTRIBUTETEXT
 - OSParseosil.tab.hpp, [1148](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- OSRLEND
 - OSParseosil.tab.hpp, [1148](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- OSRLSTART
 - OSParseosil.tab.hpp, [1148](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- OSRLSTARTEMPTY
 - OSParseosil.tab.hpp, [1148](#)
 - OSParseosol.tab.hpp, [1223](#)
 - OSParseosrl.tab.hpp, [1296](#)
- OSnl2OS.h
 - OS_AMPL_SUFFIX_DIRECTION_both, [1082](#)
 - OS_AMPL_SUFFIX_DIRECTION_local, [1082](#)
 - OS_AMPL_SUFFIX_DIRECTION_toAMPL, [1082](#)
 - OS_AMPL_SUFFIX_DIRECTION_toSolver, [1082](#)
 - OS_AMPL_SUFFIX_SCOPE_constraints, [1081](#)
 - OS_AMPL_SUFFIX_SCOPE_objectives, [1081](#)
 - OS_AMPL_SUFFIX_SCOPE_problems, [1081](#)
 - OS_AMPL_SUFFIX_SCOPE_variables, [1081](#)
 - OS_AMPL_SUFFIX_TYPE_double, [1081](#)
 - OS_AMPL_SUFFIX_TYPE_integer, [1081](#)
- OTHEREND
 - OSParseosil.tab.hpp, [1141](#), [1152](#)
 - OSParseosol.tab.hpp, [1216](#), [1226](#)
 - OSParseosrl.tab.hpp, [1289](#), [1299](#)

- OTHEROPTIONSSEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1288](#)
- OTHEROPTIONSSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1288](#)
- OTHERRESULTSEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERRESULTSSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLUTIONRESULTEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLUTIONRESULTSEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLUTIONRESULTSSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLUTIONRESULTSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLVEROUTPUTEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSOLVEROUTPUTSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1226](#)
 - OSParseosrl.tab.hpp, [1299](#)
- OTHERSTART
 - OSParseosil.tab.hpp, [1141](#), [1151](#)
 - OSParseosol.tab.hpp, [1216](#), [1226](#)
 - OSParseosrl.tab.hpp, [1288](#), [1299](#)
- OUTPUTDIRECTORIESTOMOVEEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1289](#)
- OUTPUTDIRECTORIESTOMOVESTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1289](#)
- OUTPUTFILESTOMOVEEND
 - OSParseosil.tab.hpp, [1142](#)
- OSParseosol.tab.hpp, [1217](#)
- OSParseosrl.tab.hpp, [1289](#)
- OUTPUTFILESTOMOVESTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1289](#)
- OBJECTIVESEND
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1188](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OBJECTIVESSTART
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1188](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OBJEND
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1188](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OBJSTART
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1188](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OBJTYPEATT
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- OPTIMIZATIONEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1254](#)
- OPTIMIZATIONSTART
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1254](#)
- ORDERCONEIDXATT
 - OSParseosil.tab.hpp, [1113](#)
- ORTHANTCONEEND
 - OSParseosil.tab.hpp, [1108](#)
- ORTHANTCONESTART
 - OSParseosil.tab.hpp, [1108](#)
- OS_ABS
 - OSParameters.h, [1334](#)
- OS_ALLDIFF
 - OSParameters.h, [1335](#)
- OS_AMPL_SUFFIX, [391](#)
 - direction, [391](#)
 - name, [391](#)
 - scope, [391](#)
 - type, [391](#)
- OS_COS
 - OSParameters.h, [1335](#)
- OS_DIVIDE
 - OSParameters.h, [1334](#)
- OS_E
 - OSParameters.h, [1335](#)
- OS_E_VALUE
 - OSParameters.h, [1337](#)

- OS_EPS
 - OSParameters.h, [1337](#)
- OS_ERF
 - OSParameters.h, [1334](#)
- OS_EXP
 - OSParameters.h, [1334](#)
- OS_HAS_AS_L
 - config_os_default.h, [1041](#)
- OS_HAS_BONMIN
 - config_os_default.h, [1041](#)
- OS_HAS_COUENNE
 - config_os_default.h, [1041](#)
- OS_HAS_CPPAD
 - config_os_default.h, [1041](#)
- OS_HAS_DYLP
 - config_os_default.h, [1041](#)
- OS_HAS_GLPK
 - config_os_default.h, [1041](#)
- OS_HAS_IPOPT
 - config_os_default.h, [1041](#)
- OS_HAS_SYMPHONY
 - config_os_default.h, [1041](#)
- OS_HAS_VOL
 - config_os_default.h, [1041](#)
- OS_IDENTITY_MATRIX
 - OSParameters.h, [1336](#)
- OS_IF
 - OSParameters.h, [1335](#)
- OS_LN
 - OSParameters.h, [1334](#)
- OS_MATRIX_CON
 - OSParameters.h, [1337](#)
- OS_MATRIX_DIAGONAL
 - OSParameters.h, [1336](#)
- OS_MATRIX_DOTTIMES
 - OSParameters.h, [1336](#)
- OS_MATRIX_INVERSE
 - OSParameters.h, [1336](#)
- OS_MATRIX_MINUS
 - OSParameters.h, [1336](#)
- OS_MATRIX_NEGATE
 - OSParameters.h, [1336](#)
- OS_MATRIX_OBJ
 - OSParameters.h, [1337](#)
- OS_MATRIX_PLUS
 - OSParameters.h, [1335](#)
- OS_MATRIX_PRODUCT
 - OSParameters.h, [1336](#)
- OS_MATRIX_SUM
 - OSParameters.h, [1335](#)
- OS_MATRIX_TIMES
 - OSParameters.h, [1336](#)
- OS_MATRIX_TRACE
 - OSParameters.h, [1335](#)
- OS_MATRIX_VAR
 - OSParameters.h, [1337](#)
- OS_MAX
 - OSParameters.h, [1335](#)
- OS_MIN
 - OSParameters.h, [1335](#)
- OS_MINUS
 - OSParameters.h, [1334](#)
- OS_NEAR_EQUAL
 - OSParameters.h, [1337](#)
- OS_NEGATE
 - OSParameters.h, [1334](#)
- OS_NUMBER
 - OSParameters.h, [1335](#)
- OS_PI
 - OSParameters.h, [1335](#)
- OS_PI_VALUE
 - OSParameters.h, [1337](#)
- OS_PLUS
 - OSParameters.h, [1334](#)
- OS_POWER
 - OSParameters.h, [1334](#)
- OS_PRODUCT
 - OSParameters.h, [1334](#)
- OS_SCHEMA_VERSION
 - OSParameters.h, [1337](#)
- OS_SIN
 - OSParameters.h, [1334](#)
- OS_SQRT
 - OSParameters.h, [1334](#)
- OS_SQUARE
 - OSParameters.h, [1334](#)
- OS_SUM
 - OSParameters.h, [1334](#)
- OS_TIMES
 - OSParameters.h, [1334](#)
- OS_VARIABLE
 - OSParameters.h, [1335](#)
- OS_VERSION
 - config_os_default.h, [1041](#)
- OS_VERSION_MAJOR
 - config_os_default.h, [1041](#)
- OS_VERSION_MINOR
 - config_os_default.h, [1041](#)
- OSCommandLine, [392](#)
 - ~OSCommandLine, [395](#)
 - browser, [398](#)
 - configFile, [396](#)
 - convertSolverNameToLowerCase, [395](#)
 - convertSolverNameToUpperCase, [395](#)
 - dat, [398](#)
 - datFile, [398](#)
 - filePrintLevel, [398](#)
 - gamsControlFile, [398](#)

- insList, [397](#)
- insListFile, [397](#)
- invokeHelp, [399](#)
- jobID, [398](#)
- list_options, [395](#)
- listOptions, [399](#)
- logFile, [398](#)
- mps, [397](#)
- mpsFile, [397](#)
- nl, [398](#)
- nlFile, [398](#)
- OSCommandLine, [395](#)
- OSCommandLine, [395](#)
- osil, [396](#)
- osilFile, [396](#)
- osilOutputFile, [396](#)
- osinstance, [395](#)
- osol, [396](#)
- osolFile, [396](#)
- osolOutputFile, [397](#)
- osoption, [395](#)
- osplInput, [397](#)
- osplInputFile, [397](#)
- osplOutputFile, [397](#)
- osrlFile, [397](#)
- printLevel, [398](#)
- printModel, [399](#)
- printRowNumberAsString, [399](#)
- quit, [399](#)
- reset_options, [395](#)
- serviceLocation, [396](#)
- serviceMethod, [396](#)
- solverName, [396](#)
- writeVersion, [399](#)
- OSCommandLineReader, [399](#)
 - ~OSCommandLineReader, [400](#)
 - OSCommandLineReader, [400](#)
 - OSCommandLineReader, [400](#)
 - parseString, [401](#)
 - readCommandLine, [400](#)
- OSDBL_MAX
 - OSParameters.h, [1349](#)
- OSExpressionTree, [401](#)
 - ~OSExpressionTree, [403](#)
 - bADMustReTape, [403](#)
 - bDestroyNINodes, [403](#)
 - IsEqual, [403](#)
 - m_bIndexMapGenerated, [403](#)
 - mapVarIdx, [403](#)
 - OSExpressionTree, [403](#)
 - OSExpressionTree, [403](#)
- OSGeneral, [405](#)
- OSGeneral.h
 - OSIsEqual, [1052](#)
- OSILEND
 - OSParseosil.tab.hpp, [1107](#)
- OSINT_MAX
 - OSParameters.h, [1349](#)
- OSInstance, [434](#)
 - ~OSInstance, [443](#)
 - addCone, [468–473](#)
 - addConstraint, [464](#)
 - addMatrix, [468](#)
 - addObjective, [463](#)
 - addQTermsToExpressionTree, [479](#)
 - addQTermsToExressionTree, [478](#)
 - addVariable, [462](#)
 - bAMatrixModified, [483](#)
 - bConstraintsModified, [483](#)
 - bObjectivesModified, [483](#)
 - bUseExpTreeForFunEval, [483](#)
 - bVariablesModified, [483](#)
 - calculateAllConstraintFunctionGradients, [476](#)
 - calculateAllConstraintFunctionValues, [474, 475](#)
 - calculateAllObjectiveFunctionGradients, [476](#)
 - calculateAllObjectiveFunctionValues, [475](#)
 - calculateConstraintFunctionGradient, [476](#)
 - calculateFunctionValue, [474](#)
 - calculateHessian, [478](#)
 - calculateLagrangianHessian, [477](#)
 - calculateObjectiveFunctionGradient, [477](#)
 - copyLinearConstraintCoefficients, [466](#)
 - createOSADFun, [479](#)
 - duplicateExpressionTreesMap, [479](#)
 - forwardAD, [479](#)
 - getADSparsityHessian, [480](#)
 - getAllMatrixExpressionTrees, [459](#)
 - getAllMatrixExpressionTreesMod, [459](#)
 - getAllNonlinearExpressionTrees, [453](#)
 - getAllNonlinearExpressionTreesMod, [453](#)
 - getAllNonlinearVariablesIndexMap, [478](#)
 - getConstraintConstants, [449](#)
 - getConstraintLowerBounds, [449](#)
 - getConstraintNames, [448](#)
 - getConstraintNumber, [448](#)
 - getConstraintTypes, [449](#)
 - getConstraintUpperBounds, [449](#)
 - getDenseObjectiveCoefficients, [448](#)
 - getFirstOrderResults, [481](#)
 - getInstanceCreator, [444](#)
 - getInstanceDescription, [443](#)
 - getInstanceLicence, [444](#)
 - getInstanceName, [443](#)
 - getInstanceSource, [443](#)
 - getIterateResults, [480](#)
 - getJacobianSparsityPattern, [479](#)
 - getLagrangianExpTree, [478](#)
 - getLagrangianHessianSparsityPattern, [478](#)

getLinearConstraintCoefficientMajor, 450
getLinearConstraintCoefficientNumber, 450
getLinearConstraintCoefficientsInColumnMajor, 450
getLinearConstraintCoefficientsInRowMajor, 450
getMatrix, 456
getMatrixBlockInColumnMajorForm, 457
getMatrixCoefficientsInColumnMajor, 456
getMatrixCoefficientsInRowMajor, 457
getMatrixExpressionTree, 458
getMatrixExpressionTreeInInfix, 459
getMatrixExpressionTreeInPostfix, 458
getMatrixExpressionTreeInPrefix, 458
getMatrixExpressionTreeIndexes, 459
getMatrixExpressionTreeModInPostfix, 458
getMatrixExpressions, 458
getMatrixName, 455
getMatrixNumber, 454
getMatrixSymmetry, 455
getMatrixType, 454
getNonlinearExpressionTree, 452
getNonlinearExpressionTreeInInfix, 452
getNonlinearExpressionTreeInPostfix, 452
getNonlinearExpressionTreeInPrefix, 452
getNonlinearExpressionTreeIndexes, 453
getNonlinearExpressionTreeMod, 452
getNonlinearExpressionTreeModInPostfix, 452
getNonlinearExpressionTreeModInPrefix, 453
getNonlinearExpressionTreeModIndexes, 454
getNonlinearExpressions, 452
getNumberOfBinaryVariables, 445
getNumberOfBlocksConstructors, 456
getNumberOfColumnsForMatrix, 455
getNumberOfElementConstructors, 456
getNumberOfIntegerVariables, 445
getNumberOfMatrixConstraints, 458
getNumberOfMatrixExpressionTreeIndexes, 459
getNumberOfMatrixExpressions, 458
getNumberOfMatrixObjectives, 457
getNumberOfMatrixVariables, 457
getNumberOfNonlinearConstraints, 453
getNumberOfNonlinearExpressionTreeIndexes, 453
getNumberOfNonlinearExpressionTreeModIndexes, 454
getNumberOfNonlinearExpressions, 451
getNumberOfNonlinearObjectives, 453
getNumberOfQuadraticRowIndexes, 451
getNumberOfQuadraticTerms, 451
getNumberOfRowsForMatrix, 455
getNumberOfSemiContinuousVariables, 445
getNumberOfSemiIntegerVariables, 445
getNumberOfStringVariables, 446
getNumberOfTransformationConstructors, 456
getObjectiveCoefficientNumbers, 447
getObjectiveCoefficients, 448
getObjectiveConstants, 447
getObjectiveMaxOrMins, 447
getObjectiveNames, 446
getObjectiveNumber, 446
getObjectiveWeights, 447
getQuadraticRowIndexes, 451
getQuadraticTerms, 451
getSecondOrderResults, 481
getSparseJacobianFromColumnMajor, 478
getSparseJacobianFromRowMajor, 478
getTimeDomainFormat, 459
getTimeDomainIntervalHorizon, 461
getTimeDomainIntervalStart, 461
getTimeDomainStageConList, 460
getTimeDomainStageNames, 460
getTimeDomainStageNumber, 459
getTimeDomainStageNumberOfConstraints, 460
getTimeDomainStageNumberOfObjectives, 460
getTimeDomainStageNumberOfVariables, 460
getTimeDomainStageObjList, 460
getTimeDomainStageVarList, 460
getVariableLowerBounds, 446
getVariableNames, 444
getVariableNumber, 444
getVariableTypes, 444
getVariableUpperBounds, 446
getZeroOrderResults, 480
initForAlgDiff, 481
initObjGradients, 481
initializeNonLinearStructures, 474
instanceData, 483
instanceHeader, 482
isEqual, 443
matrixHasBase, 456
matrixHasBlocks, 456
matrixHasElements, 456
matrixHasTransformations, 456
OSInstance, 443
OSInstance, 443
printModel, 474
reverseAD, 480
setConeNumber, 468
setConstraintNumber, 464
setConstraints, 465
setInstanceCreator, 462
setInstanceDescription, 461
setInstanceLicence, 462
setInstanceName, 461
setInstanceSource, 461
setLinearConstraintCoefficients, 465
setMatrixNumber, 467
setNonlinearExpressions, 467
setNumberOfQuadraticTerms, 466
setObjectiveNumber, 463

- setObjectives, 464
- setQuadraticCoefficients, 466
- setQuadraticTermsInNonlinearExpressions, 467
- setTimeDomain, 481
- setTimeDomainInterval, 482
- setTimeDomainStageConstraintsOrdered, 482
- setTimeDomainStageConstraintsUnordered, 482
- setTimeDomainStageObjectivesOrdered, 482
- setTimeDomainStageObjectivesUnordered, 482
- setTimeDomainStageVariablesOrdered, 482
- setTimeDomainStageVariablesUnordered, 482
- setTimeDomainStages, 481
- setVariableNumber, 462
- setVariables, 463
- OSIsEqual
 - OSGeneral.h, 1052
- OSIsnan
 - OSParameters.h, 1345
- OSMathUtil.h
 - getMult, 1325
 - getMultIncr, 1325
 - OSRand, 1326
 - OSIRand, 1326
 - os_strtod_wrap, 1325
- OSMatlab, 483
 - ~OSMatlab, 486
 - bl, 486
 - bu, 487
 - createOSInstance, 486
 - instanceName, 488
 - numCon, 487
 - numQTerms, 487
 - numVar, 487
 - OSMatlab, 486
 - obj, 487
 - objType, 487
 - osil, 488
 - osinstance, 488
 - OSMatlab, 486
 - qIndex1, 488
 - qIndex2, 488
 - qRows, 487
 - qVal, 488
 - sAgentAddress, 488
 - sSolverName, 488
 - solve, 486
 - solverType, 488
 - sparseMat, 486
 - varType, 487
 - vl, 487
 - vu, 487
- OSMatrix, 489
 - ~OSMatrix, 491
 - alignsOnBlockBoundary, 492
 - cloneMatrixNode, 493
 - createConstructorTreeFromPrefix, 491
 - deepCopyFrom, 494
 - getColumnPartition, 492
 - getColumnPartitionSize, 492
 - getMatrixNodeInXML, 493
 - getMatrixType, 491
 - getNodeName, 491
 - getNodeType, 491
 - getRowPartition, 491
 - getRowPartitionSize, 491
 - idx, 494
 - isBlockDiagonal, 493
 - IsEqual, 493
 - name, 494
 - OSMatrix, 491
 - OSMatrix, 491
 - processBlocks, 492
 - setMatrix, 493
 - setRandom, 493
- OSNaN
 - OSParameters.h, 1345
- OSOLATTRIBUTETEXT
 - OSParseosol.tab.hpp, 1176
- OSOLEND
 - OSParseosol.tab.hpp, 1176
- OSOLSTART
 - OSParseosol.tab.hpp, 1176
- OSOLSTARTEMPTY
 - OSParseosol.tab.hpp, 1176
- OSOption, 675
 - ~OSOption, 686
 - deepCopyFrom, 686
 - general, 713
 - getAllOtherConstraintOptions, 703
 - getAllOtherObjectiveOptions, 701
 - getAllOtherOptions, 693
 - getAllOtherVariableOptions, 698
 - getAllSolverOptions, 704
 - getContact, 687
 - getContactTransportType, 688
 - getDirectoriesToDelete, 694
 - getDirectoriesToMake, 693
 - getFileCreator, 687
 - getFileDescription, 687
 - getFileLicence, 687
 - getFileName, 686
 - getFileSource, 686
 - getFilesToDelete, 694
 - getFilesToMake, 694
 - getInitBasisStatusDense, 696
 - getInitBasisStatusSparse, 696
 - getInitConValuesDense, 701
 - getInitConValuesSparse, 701

getInitDualVarLowerBoundsDense, 701
getInitDualVarUpperBoundsDense, 702
getInitDualVarValuesSparse, 701
getInitObjBoundsSparse, 699
getInitObjLowerBoundsDense, 699
getInitObjUpperBoundsDense, 699, 700
getInitObjValuesDense, 698, 699
getInitObjValuesSparse, 698
getInitVarValuesDense, 695
getInitVarValuesSparse, 695
getInitVarValuesStringDense, 695, 696
getInitVarValuesStringSparse, 695
getInitialBasisElements, 697
getInputDirectoriesToMove, 694
getInputFilesToMove, 694
getInstanceLocation, 687
getInstanceLocationType, 687
getInstanceName, 687
getIntegerVariableBranchingWeightsDense, 697
getIntegerVariableBranchingWeightsSparse, 697
getJobDependencies, 693
getJobID, 687
getLicense, 687
getMaxTime, 689
getMaxTimeUnit, 688
getMinCPUNumber, 689
getMinCPUNumberDescription, 688
getMinCPUSpeed, 689
getMinCPUSpeedDescription, 688
getMinCPUSpeedUnit, 688
getMinDiskSpace, 688
getMinDiskSpaceDescription, 688
getMinDiskSpaceUnit, 688
getMinMemoryDescription, 688
getMinMemorySize, 688
getMinMemoryUnit, 688
getNumberOfConstraints, 690
getNumberOfDirectoriesToDelete, 690
getNumberOfDirectoriesToMake, 689
getNumberOfFilesToDelete, 690
getNumberOfFilesToMake, 689
getNumberOfInitConValues, 692
getNumberOfInitDualVarValues, 692
getNumberOfInitObjBounds, 691
getNumberOfInitObjValues, 691
getNumberOfInitVarValues, 690
getNumberOfInitVarValuesString, 690
getNumberOfInitialBasisElements, 696
getNumberOfInputDirectoriesToMove, 689
getNumberOfInputFilesToMove, 690
getNumberOfIntegerVariableBranchingWeights, 691
getNumberOfJobDependencies, 689
getNumberOfObjectives, 690
getNumberOfOtherConstraintOptions, 692
getNumberOfOtherGeneralOptions, 689
getNumberOfOtherJobOptions, 689
getNumberOfOtherObjectiveOptions, 691
getNumberOfOtherServiceOptions, 689
getNumberOfOtherSystemOptions, 689
getNumberOfOtherVariableOptions, 691
getNumberOfOutputDirectoriesToMove, 690
getNumberOfOutputFilesToMove, 690
getNumberOfProcessesToKill, 690
getNumberOfRequiredDirectories, 689
getNumberOfRequiredFiles, 689
getNumberOfSOS, 691
getNumberOfSOSVarBranchingWeights, 691
getNumberOfSolverOptions, 692
getNumberOfVariables, 690
getObjectiveInitialBasisStatusDense, 700
getOptionDbl, 689
getOptionInt, 692
getOptionStr, 688
getOtherConstraintOption, 703
getOtherConstraintOptions, 702
getOtherGeneralOptions, 692
getOtherJobOptions, 693
getOtherObjectiveOption, 700
getOtherObjectiveOptions, 700
getOtherOptions, 693
getOtherServiceOptions, 692
getOtherSystemOptions, 692
getOtherVariableOption, 698
getOtherVariableOptions, 698
getOutputDirectoriesToMove, 694
getOutputFilesToMove, 694
getPassword, 687
getProcessesToKill, 695
getRequestedStartTime, 688
getRequiredDirectories, 693
getRequiredFiles, 693
getSOSVariableBranchingWeightsSparse, 698
getServiceName, 687
getServiceType, 688
getServiceURI, 687
getSlackVariableInitialBasisStatusDense, 702
getSolverOptions, 703
getSolverToInvoke, 687
getUserName, 687
getVariableInitialBasisStatusDense, 696
isEqual, 686
job, 713
OSOption, 686
optimization, 713
optionHeader, 713
OSOption, 686
service, 713
setAnotherConstraintOption, 712

setAnotherGeneralOption, 705
setAnotherJobOption, 708
setAnotherObjectiveOption, 710
setAnotherServiceOption, 706
setAnotherSystemOption, 706
setAnotherVariableOption, 709
setAnotherDirectoryToDelete, 707
setAnotherDirectoryToMake, 707
setAnotherFileToDelete, 707
setAnotherFileToMake, 707
setAnotherInitBasisStatus, 708
setAnotherInitConValue, 711
setAnotherInitDualVarValue, 711
setAnotherInitObjBound, 710
setAnotherInitObjValue, 710
setAnotherInitVarValue, 708
setAnotherInitVarValueString, 708
setAnotherInputDirectoryToMove, 707
setAnotherInputFileToMove, 707
setAnotherIntegerVariableBranchingWeight, 709
setAnotherJobDependency, 707
setAnotherOutputDirectoryToMove, 707
setAnotherOutputFileToMove, 707
setAnotherProcessToKill, 708
setAnotherRequiredDirectory, 707
setAnotherRequiredFile, 707
setAnotherSOSVariableBranchingWeight, 709
setAnotherSolverOption, 713
setContact, 704, 705
setContactTransportType, 705
setDirectoriesToDelete, 707
setDirectoriesToMake, 707
setFilesToDelete, 707
setFilesToMake, 707
setHeader, 686
setInitBasisStatus, 708
setInitBasisStatusDense, 708
setInitBasisStatusSparse, 708
setInitConValues, 711
setInitConValuesDense, 711
setInitConValuesSparse, 711
setInitDualValues, 711
setInitDualVarValuesDense, 711
setInitDualVarValuesSparse, 711
setInitObjBounds, 710
setInitObjBoundsDense, 710
setInitObjBoundsSparse, 710
setInitObjValues, 710
setInitObjValuesDense, 710
setInitObjValuesSparse, 710
setInitVarValues, 708
setInitVarValuesDense, 708
setInitVarValuesSparse, 708
setInitVarValuesString, 708
setInitVarValuesStringDense, 708
setInitVarValuesStringSparse, 708
setInputDirectoriesToMove, 707
setInputFilesToMove, 707
setInstanceLocation, 704
setInstanceLocationType, 704
setInstanceName, 704
setIntegerVariableBranchingWeights, 708
setIntegerVariableBranchingWeightsDense, 709
setIntegerVariableBranchingWeightsSparse, 709
setJobDependencies, 706
setJobID, 704
setLicense, 704
setMaxTime, 706
setMaxTimeUnit, 706
setMinCPUNumber, 706
setMinCPUSpeed, 706
setMinCPUSpeedUnit, 706
setMinDiskSpace, 705
setMinDiskSpaceUnit, 705
setMinMemorySize, 705, 706
setMinMemoryUnit, 706
setNumberOfConstraints, 708
setNumberOfObjectives, 708
setNumberOfOtherConstraintOptions, 711
setNumberOfOtherObjectiveOptions, 710
setNumberOfOtherVariableOptions, 709
setNumberOfSolverOptions, 712
setNumberOfVariables, 708
setOptionDbl, 713
setOptionInt, 713
setOptionStr, 713
setOtherConstraintOptionAttributes, 712
setOtherConstraintOptionCon, 712
setOtherConstraintOptions, 711
setOtherGeneralOptions, 705
setOtherJobOptions, 708
setOtherObjectiveOptionAttributes, 710
setOtherObjectiveOptionObj, 711
setOtherObjectiveOptions, 710
setOtherOptionEnumeration, 709
setOtherServiceOptions, 706
setOtherSystemOptions, 706
setOtherVariableOptionAttributes, 709
setOtherVariableOptionVar, 710
setOtherVariableOptions, 709
setOutputDirectoriesToMove, 707
setOutputFilesToMove, 707
setPassword, 704
setPathPairs, 707
setProcessesToKill, 708
setRandom, 686
setRequestedStartTime, 706
setRequiredDirectories, 707

- setRequiredFiles, 707
- setSOSVariableBranchingWeights, 709
- setServiceName, 704
- setServiceType, 706
- setServiceURI, 704
- setSolverOptionContent, 712
- setSolverOptions, 713
- setSolverToInvoke, 704
- setUserName, 704
- system, 713
- OSOutput, 721
 - ~OSOutput, 723
 - AddChannel, 724
 - DeleteChannel, 724
 - FindChannel, 724
 - FlushAllBuffers, 723
 - OSOutput, 723
 - OSPrint, 723
 - OSOutput, 723
 - SetPrintLevel, 723, 724
- OSOutput.h
 - osoutput, 1327
- OSOutputChannel, 725
 - ~OSOutputChannel, 726
 - flushBuffer, 727
 - isAccepted, 727
 - Name, 726
 - OSOutputChannel, 726
 - OSPrintf, 727
 - Open, 727
 - OSOutputChannel, 726
 - setAllPrintLevels, 726
 - setPrintLevel, 726
- OSParameters.h
 - ENUM_BASIS_STATUS, 1341
 - ENUM_CONE_TYPE, 1344
 - ENUM_CPUSPEEDUNIT, 1338
 - ENUM_JOB_STATUS, 1341
 - ENUM_LOCATIONTYPE, 1340
 - ENUM_MATRIX_TYPE, 1343
 - ENUM_OUTPUT_AREA, 1338
 - ENUM_OUTPUT_LEVEL, 1337
 - ENUM_PATHPAIR, 1342
 - ENUM_SERVICE_TYPE, 1340
 - ENUM_STORAGEUNIT, 1339
 - ENUM_TIMECATEGORY, 1339
 - ENUM_TIMETYPE, 1339
 - ENUM_TIMEUNIT, 1339
 - ENUM_VARTYPE, 1342
 - mergeMatrixType, 1348
 - OS_ABS, 1334
 - OS_ALLDIFF, 1335
 - OS_COS, 1335
 - OS_DIVIDE, 1334
 - OS_E, 1335
 - OS_E_VALUE, 1337
 - OS_EPS, 1337
 - OS_ERF, 1334
 - OS_EXP, 1334
 - OS_IF, 1335
 - OS_LN, 1334
 - OS_MATRIX_CON, 1337
 - OS_MATRIX_INVERSE, 1336
 - OS_MATRIX_MINUS, 1336
 - OS_MATRIX_NEGATE, 1336
 - OS_MATRIX_OBJ, 1337
 - OS_MATRIX_PLUS, 1335
 - OS_MATRIX_PRODUCT, 1336
 - OS_MATRIX_SUM, 1335
 - OS_MATRIX_TIMES, 1336
 - OS_MATRIX_TRACE, 1335
 - OS_MATRIX_VAR, 1337
 - OS_MAX, 1335
 - OS_MIN, 1335
 - OS_MINUS, 1334
 - OS_NEAR_EQUAL, 1337
 - OS_NEGATE, 1334
 - OS_NUMBER, 1335
 - OS_PI, 1335
 - OS_PI_VALUE, 1337
 - OS_PLUS, 1334
 - OS_POWER, 1334
 - OS_PRODUCT, 1334
 - OS_SCHEMA_VERSION, 1337
 - OS_SIN, 1334
 - OS_SQRT, 1334
 - OS_SQUARE, 1334
 - OS_SUM, 1334
 - OS_TIMES, 1334
 - OS_VARIABLE, 1335
 - OSDBL_MAX, 1349
 - OSINT_MAX, 1349
 - OSIsnan, 1345
 - OSNaN, 1345
 - OSgetVersionInfo, 1345
 - returnBasisStatus, 1347
 - returnBasisStatusString, 1347
 - returnCPUSpeedUnit, 1345
 - returnConReferenceValueType, 1348
 - returnConReferenceValueTypeString, 1348
 - returnConeType, 1349
 - returnExprShapeString, 1348
 - returnGeneralResultStatus, 1346
 - returnJobStatus, 1347
 - returnLocationType, 1346
 - returnMatrixConstructorType, 1348
 - returnMatrixSymmetry, 1348
 - returnMatrixSymmetryString, 1348

- returnMatrixType, 1347
- returnMatrixTypeString, 1348
- returnNIExprShape, 1348
- returnServiceType, 1346
- returnSolutionStatus, 1347
- returnSolutionSubstatusType, 1347
- returnStorageUnit, 1345
- returnSystemCurrentState, 1347
- returnTimeCategory, 1346
- returnTimeType, 1346
- returnTimeUnit, 1346
- returnTransportType, 1346
- returnVarType, 1347
- verifyBasisStatus, 1347
- verifyCPUSpeedUnit, 1345
- verifyConReferenceValueType, 1348
- verifyConeType, 1349
- verifyGeneralResultStatus, 1346
- verifyJobStatus, 1347
- verifyLocationType, 1346
- verifyMatrixConstructorType, 1348
- verifyMatrixSymmetry, 1348
- verifyMatrixType, 1348
- verifyNIExprShape, 1349
- verifyServiceType, 1346
- verifySolutionStatus, 1347
- verifySolutionSubstatusType, 1347
- verifyStorageUnit, 1345
- verifySystemCurrentState, 1347
- verifyTimeCategory, 1346
- verifyTimeType, 1346
- verifyTimeUnit, 1346
- verifyTransportType, 1346
- verifyVarType, 1347
- OSParseosil.tab.hpp
 - ABSEND, 1126
 - ABSSTART, 1126
 - ALLDIFFEND, 1126
 - ALLDIFFSTART, 1126
 - ATTRIBUTETEXT, 1106
 - AXISDIRECTIONATT, 1110
 - BASE64END, 1117
 - BASE64START, 1117
 - BASEMATRIXEND, 1117
 - BASEMATRIXIDXATT, 1119
 - BASEMATRIXSTART, 1117
 - BASETRANPOSEATT, 1120
 - BLOCKCOLIDXATT, 1122
 - BLOCKEND, 1118
 - BLOCKROWIDXATT, 1122
 - BLOCKSEND, 1118
 - BLOCKSSTART, 1118
 - BLOCKSTART, 1118
 - COEFATT, 1119
 - COLOFFSETSEND, 1122
 - COLOFFSETSTART, 1122
 - COMPONENTSEND, 1110
 - COMPONENTSSTART, 1110
 - CONEND, 1114
 - CONESEND, 1108
 - CONESSTART, 1108
 - CONSTANTATT, 1119
 - CONSTART, 1114
 - CONSTRAINTSEND, 1114
 - CONSTRAINTSSTART, 1113
 - COSEND, 1125
 - COSSTART, 1125
 - DIRECTIONEND, 1110
 - DIRECTIONSTART, 1110
 - DIVIDEEND, 1124
 - DIVIDESTART, 1124
 - DOUBLE, 1107
 - DUALCONEEND, 1109
 - DUALCONESTART, 1109
 - DUMMY, 1122
 - EEND, 1126
 - ELEMENTSEND, 1120
 - ELEMENTSSTART, 1120
 - ELEMENTTEXT, 1106
 - ELEND, 1117
 - ELSTART, 1117
 - EMPTYIDATT, 1107
 - EMPTYNAMEATT, 1118
 - EMPTYROWMAJORATT, 1122
 - EMPTYSHAPEATT, 1118
 - EMPTYSYMMETRYATT, 1118
 - EMPTYTYPEATT, 1118
 - ENDOFELEMENT, 1107
 - ENUMERATIONEND, 1116
 - ENUMERATIONSTART, 1116
 - ERFEND, 1126
 - ERFSTART, 1126
 - ESTART, 1126
 - EXPEND, 1125
 - EXPEND, 1123
 - EXPRSTART, 1123
 - EXPSTART, 1125
 - FACTORSEND, 1110
 - FACTORSTART, 1110
 - FILECREATOREMPTY, 1116
 - FILECREATOREND, 1116
 - FILECREATORSTART, 1116
 - FILEDESCRIPTIONEND, 1115
 - FILELICENCEEMPTY, 1116
 - FILELICENCEEND, 1116
 - FILELICENCESTART, 1116
 - FILENAMEEMPTY, 1115
 - FILENAMEEND, 1115

FILENAMESTART, 1115
FILESOURCEEMPTY, 1115
FILESOURCEEND, 1115
FILESOURCESTART, 1115
GENERALELEMENTSEND, 1121
GREATERTHAN, 1107
HEADEREND, 1115
HEADERSTART, 1115
HORIZONATT, 1113
IDATT, 1130
IDENTITYMATRIXEND, 1129
IDXATT, 1119
IDXONEATT, 1107
IDXTWOATT, 1107
IFEND, 1125
IFSTART, 1125
INCLUDEDIAGONALATT, 1130
INCRATT, 1117
INDEXESEND, 1120
INDEXESSTART, 1120
INSTANCEDATAEND, 1107
INTEGER, 1106
INTERVALEND, 1115
INTERVALSTART, 1115
ITEMEMPTY, 1116
ITEMEND, 1117
ITEMSTART, 1117
ITEMSTARTANDEND, 1117
ITEMTEXT, 1106
LBCONEIDXATT, 1112
LBMATRIXIDXATT, 1112
LINEARELEMENTSEND, 1121
LNEND, 1124
LNSTART, 1124
MATRICESEND, 1108
MATRICESSTART, 1108
MATRIXCONEND, 1112
MATRIXCONSTART, 1112
MATRIXDIAGONALEND, 1127
MATRIXDOTTIMESEND, 1128
MATRIXEND, 1117
MATRIXIDXATT, 1112
MATRIXINVERSEEND, 1130
MATRIXINVERSESTART, 1129
MATRIXMERGEEND, 1128
MATRIXMERGESTART, 1128
MATRIXMINUSEND, 1128
MATRIXMINUSSTART, 1128
MATRIXNEGATEEND, 1128
MATRIXNEGATESTART, 1128
MATRIXOBJEND, 1112
MATRIXOBJSTART, 1111
MATRIXPLUSEND, 1128
MATRIXPLUSSTART, 1128
MATRIXPRODUCTEND, 1129
MATRIXPRODUCTSTART, 1129
MATRIXREFERENCEEND, 1129
MATRIXSTART, 1117
MATRIXTERMEND, 1123
MATRIXTERMSTART, 1123
MATRIXTIMESEND, 1129
MATRIXTIMESSTART, 1128
MATRIXTOSCALAREND, 1127
MATRIXTRACEEND, 1127
MATRIXTRACESTART, 1127
MATRIXTRANSPOSEEND, 1129
MATRIXVAREND, 1111
MATRIXVARIABLESEND, 1111
MATRIXVARSTART, 1111
MAXEND, 1126
MAXSTART, 1126
MINEND, 1126
MINSTART, 1126
MINUSEND, 1124
MINUSSTART, 1124
MULTATT, 1117
NAMEATT, 1118
NEGATEEND, 1125
NEGATESTART, 1125
NEGATIVEPATTERNATT, 1118
NLEND, 1123
NLSTART, 1123
NONNEGATIVECONEEND, 1108
NONPOSITIVECONEEND, 1108
NONZEROSSEND, 1120
NONZEROSSTART, 1120
NORMSCALEFACTORATT, 1110
NUMBEREND, 1127
NUMBEROFBLOCKSATT, 1119
NUMBEROFCOLUMNSATT, 1119
NUMBEROFCONESATT, 1108
NUMBEROFELATT, 1116
NUMBEROFEXPR, 1123
NUMBEROFQTERMSATT, 1107
NUMBEROFROWSATT, 1119
NUMBEROFSTAGESATT, 1113
NUMBEROFVALUESATT, 1119
NUMBEROFVARIDXATT, 1119
NUMBERSTART, 1127
OBJECTIVESEND, 1114
OBJECTIVESSTART, 1114
OBJEND, 1114
OBJSTART, 1114
ORDERCONEIDXATT, 1113
ORTHANTCONEEND, 1108
ORTHANTCONESTART, 1108
OSILEND, 1107
PATTERNELEMENTSEND, 1122

PIEND, 1127
PISTART, 1126
PLUSEND, 1124
PLUSSTART, 1124
POLARCONEND, 1110
POLARCONESTART, 1110
POLYHEDRALCONEND, 1109
POWEREND, 1124
POWERSTART, 1123
PRODUCTCONEND, 1109
PRODUCTCONESTART, 1109
PRODUCTEND, 1125
PRODUCTSTART, 1124
QTERMEND, 1108
QTERMSTART, 1108
QUADRATICCONEND, 1109
QUADRATICCONESTART, 1109
QUOTE, 1106
ROWMAJORATT, 1122
ROWOFFSETSEND, 1122
ROWOFFSETSSTART, 1122
SHAPEATT, 1118
SINEND, 1125
SINSTART, 1125
SIZEOFATT, 1117
SQRTEND, 1124
SQRTSTART, 1124
SQUAREEND, 1125
SQUARESTART, 1125
STAGEEND, 1113
STAGESEND, 1113
STAGESSTART, 1113
STAGESTART, 1113
STARTATT, 1113
STARTIDXATT, 1114
STARTVECTOREND, 1120
STARTVECTORSTART, 1120
SUMEND, 1124
SUMSTART, 1124
SYMMETRYATT, 1118
TIMEDOMAINEND, 1113
TIMEDOMAINSTART, 1113
TIMESEND, 1127
TIMESSTART, 1127
TRANSFORMATIONEND, 1122
TWOQUOTES, 1107
TYPEATT, 1118
UBCONEIDXATT, 1112
UBMATRIXIDXATT, 1112
VALUEATT, 1107
VALUESEND, 1121
VALUESSTART, 1121
VALUETYPEATT, 1121
VAREND, 1114
VARIABLEEND, 1126
VARIABLESEND, 1114
VARIABLESSTART, 1113
VARIABLESTART, 1125
VARIDXEND, 1122
VARIDXSTART, 1122
VARSTART, 1114
VARTYPEATT, 1111
YYLTYPE, 1130
YYSTYPE, 1130
yyltype, 1130
yystype, 1130
yytokentype, 1130
OSParseosol.tab.hpp
ABSEND, 1200
ABSSTART, 1200
ALLDIFFEND, 1201
ALLDIFFSTART, 1201
ATEQUALITYEND, 1187
ATEQUALITYSTART, 1187
ATLOWEREND, 1187
ATLOWERSTART, 1187
ATTRIBUTETEXT, 1175
ATUPPEREND, 1187
ATUPPERSTART, 1187
BASE64END, 1190
BASE64START, 1190
BASEMATRIXEND, 1193
BASEMATRIXIDXATT, 1194
BASEMATRIXSTART, 1193
BASETRANPOSEATT, 1195
BASICEND, 1187
BASICSTART, 1187
BLOCKCOLIDXATT, 1197
BLOCKEND, 1193
BLOCKROWIDXATT, 1197
BLOCKSEND, 1193
BLOCKSSTART, 1193
BLOCKSTART, 1193
CATEGORYATT, 1177
COEFATT, 1194
COLOFFSETSEND, 1197
COLOFFSETSSTART, 1197
CONEND, 1189
CONSTANTATT, 1193
CONSTART, 1189
CONSTRAINTSEND, 1189
CONSTRAINTSSTART, 1189
CONTACTEND, 1182
CONTACTSTART, 1182
CONTYPEATT, 1178
COSEND, 1200
COSSTART, 1200
DEPENDENCIESEND, 1184

DEPENDENCIESSTART, 1184
DESCRIPTIONATT, 1178
DIVIDEEND, 1199
DIVIDESTART, 1199
DOUBLE, 1175
DUMMY, 1197
EEND, 1201
ELEMENTSEND, 1195
ELEMENTSSTART, 1195
ELEMENTTEXT, 1175
EEND, 1190
ELSTART, 1190
EMPTYCATEGORYATT, 1179
EMPTYCONTYPEATT, 1178
EMPTYENUMTYPEATT, 1179
EMPTYLBVALUEATT, 1179
EMPTYNAMEATT, 1179
EMPTYOBJTYPEATT, 1179
EMPTYROWMAJORATT, 1197
EMPTYSHAPEATT, 1193
EMPTYSolverATT, 1180
EMPTYSYMMETRYATT, 1193
EMPTYTYPEATT, 1179
EMPTYUBVALUEATT, 1179
EMPTYUNITATT, 1179
EMPTYVALUEATT, 1179
EMPTYVARTYPEATT, 1179
EMPTYWEIGHTATT, 1180
ENDOFELEMENT, 1176
ENUMERATIONEND, 1190
ENUMERATIONSTART, 1189
ENUMTYPEATT, 1178
ERFEND, 1201
ERFSTART, 1201
ESTART, 1201
EXPEND, 1199
EXPEND, 1198
EXPRSTART, 1198
EXPSTART, 1199
FILECREATOREMPTY, 1192
FILECREATOREND, 1192
FILECREATORSTART, 1192
FILEDESCRIPTIONEND, 1192
FILELICENCEEMPTY, 1192
FILELICENCEEND, 1192
FILELICENCESTART, 1192
FILENAMEEMPTY, 1191
FILENAMEEND, 1191
FILENAMESTART, 1191
FILESOURCEEMPTY, 1191
FILESOURCEEND, 1191
FILESOURCESTART, 1191
FILESTODELETEEND, 1185
FILESTODELETESTART, 1185
FILESTOMAKEEND, 1185
FILESTOMAKESTART, 1185
FROMATT, 1176
GENERALELEMENTSEND, 1196
GENERALEND, 1180
GENERALSTART, 1180
GREATERTHAN, 1176
GROUPWEIGHTATT, 1177
HEADEREND, 1191
HEADERSTART, 1191
IDATT, 1205
IDENTITYMATRIXEND, 1204
IDXATT, 1178
IFEND, 1200
IFSTART, 1200
INCLUDEDIAGONALATT, 1204
INCRATT, 1190
INDEXESEND, 1195
INDEXESSTART, 1195
INSTANCENAMEEND, 1181
INSTANCENAMESTART, 1181
INTEGER, 1175
ISFREEEND, 1187
ISFREESTART, 1187
ITEMEMPTY, 1190
ITEMEND, 1190
ITEMSTART, 1190
ITEMSTARTANDEND, 1190
ITEMTEXT, 1175
JOBEND, 1181
JOBIDEND, 1182
JOBIDSTART, 1182
JOBSTART, 1181
LBDUALVALUEATT, 1180
LBVALUEATT, 1179
LICENSEEND, 1182
LICENSESTART, 1182
LINEARELEMENTSEND, 1196
LNEND, 1199
LNSTART, 1199
LOCATIONTYPEATT, 1180
MAKECOPYATT, 1177
MATRIXCONEND, 1191
MATRIXCONSTART, 1191
MATRIXDIAGONALEND, 1202
MATRIXDOTTIMESEND, 1202
MATRIXEND, 1192
MATRIXINVERSEEND, 1204
MATRIXINVERSESTART, 1204
MATRIXMERGEEND, 1203
MATRIXMERGESTART, 1203
MATRIXMINUSEND, 1203
MATRIXMINUSSTART, 1203
MATRIXNEGATEEND, 1203

MATRIXNEGATESTART, 1203
MATRIXOBJEND, 1191
MATRIXOBJSTART, 1191
MATRIXPLUSEND, 1203
MATRIXPLUSSTART, 1203
MATRIXPRODUCTEND, 1203
MATRIXPRODUCTSTART, 1203
MATRIXREFERENCEEND, 1204
MATRIXSTART, 1192
MATRIXTERMEND, 1198
MATRIXTERMSTART, 1198
MATRIXTIMESEND, 1203
MATRIXTIMESSTART, 1203
MATRIXTOSCALAREND, 1202
MATRIXTRACEEND, 1202
MATRIXTRACESTART, 1202
MATRIXTRANPOSEEND, 1204
MATRIXVAREND, 1190
MATRIXVARSTART, 1190
MAXEND, 1201
MAXSTART, 1201
MAXTIMEEND, 1184
MAXTIMESTART, 1183
MINCPUNUMBEREND, 1183
MINCPUNUMBERSTART, 1183
MINCPUSPEEDEND, 1183
MINCPUSPEEDSTART, 1183
MINDISKSPACEEND, 1183
MINDISKSPACESTART, 1183
MINEND, 1201
MINMEMORYEND, 1183
MINMEMORYSTART, 1183
MINSTART, 1201
MINUSEND, 1199
MINUSSTART, 1199
MULTATT, 1190
NAMEATT, 1178
NEGATEEND, 1200
NEGATESTART, 1200
NEGATIVEPATTERNATT, 1193
NLEND, 1198
NLSTART, 1198
NONZEROSEND, 1195
NONZEROSTART, 1195
NUMBEREND, 1202
NUMBEROFBLOCKSATT, 1193
NUMBEROFCOLUMNSATT, 1194
NUMBEROFCONATT, 1178
NUMBEROFELATT, 1178
NUMBEROFEXPR, 1198
NUMBEROFITEMSATT, 1178
NUMBEROFJOBIDSATT, 1176
NUMBEROFJOBATT, 1178
NUMBEROFPATHSATT, 1176
NUMBEROFROWSATT, 1194
NUMBEROFSOSATT, 1177
NUMBEROFVALUESATT, 1194
NUMBEROFVARATT, 1178
NUMBEROFVARIDXATT, 1194
NUMBERSTART, 1202
OBJECTIVESEND, 1188
OBJECTIVESSTART, 1188
OBJEND, 1188
OBJSTART, 1188
OBJTYPEATT, 1179
OPTIMIZATIONEND, 1181
OPTIMIZATIONSTART, 1181
OSOLATTRIBUTETEXT, 1176
OSOLEND, 1176
OSOLSTART, 1176
OSOLSTARTEMPT, 1176
OTHEREND, 1183
OTHEROPTIONSSEND, 1183
OTHEROPTIONSSTART, 1182
OTHERSTART, 1183
PASSWORDEND, 1182
PASSWORDSTART, 1182
PATHEND, 1184
PATHPAIREND, 1184
PATHPAIRSTART, 1184
PATHSTART, 1184
PATTERNELEMENTSEND, 1196
PIEND, 1201
PISTART, 1201
PLUSEND, 1198
PLUSSTART, 1198
POWEREND, 1198
POWERSTART, 1198
PROCESSEND, 1186
PROCESSESTOKILLEND, 1186
PROCESSSTART, 1186
PRODUCTEND, 1199
PRODUCTSTART, 1199
QUOTE, 1176
REQUIREDFILESSEND, 1184
REQUIREDFILESSTART, 1184
ROWMAJORATT, 1197
ROWOFFSETSEND, 1197
ROWOFFSETSTART, 1197
SERVICEEND, 1181
SERVICENAMEEND, 1181
SERVICENAMESTART, 1181
SERVICESTART, 1181
SERVICETYPEEND, 1183
SERVICETYPESTART, 1183
SERVICEURIEND, 1181
SERVICEURISTART, 1181
SHAPEATT, 1193

SINEND, 1200
SINSTART, 1200
SIZEOFATT, 1190
SOLVERATT, 1180
SOLVEROPTIONEND, 1189
SOLVEROPTIONSSEND, 1189
SOLVEROPTIONSSTART, 1189
SOLVEROPTIONSTART, 1189
SOLVERTOINVOKEEND, 1182
SOSEND, 1188
SOSIDXATT, 1178
SOSSTART, 1188
SQRTEND, 1199
SQRTSTART, 1199
SQUAREEND, 1200
SQUARESTART, 1200
STARTVECTOREND, 1195
STARTVECTORSTART, 1195
SUMEND, 1199
SUMSTART, 1199
SUPERBASICEND, 1187
SUPERBASICSTART, 1187
SYMMETRYATT, 1193
SYSTEMEND, 1181
SYSTEMSTART, 1180
TIMESEND, 1201
TIMESSTART, 1201
TOATT, 1177
TRANSFORMATIONEND, 1197
TRANSPORTTYPEATT, 1180
TWOQUOTES, 1176
TYPEATT, 1177
UBDUALVALUEATT, 1180
UBVALUEATT, 1180
UNITATT, 1178
UNKNOWNEND, 1188
UNKNOWNSTART, 1187
USERNAMEEND, 1182
USERNAMESTART, 1182
VALUEATT, 1178
VALUESEND, 1195
VALUESSTART, 1195
VALUETYPEATT, 1196
VAREND, 1186
VARIABLEEND, 1200
VARIABLESEND, 1186
VARIABLESSTART, 1186
VARIABLESTART, 1200
VARIDXEND, 1196
VARIDXSTART, 1196
VARSTART, 1186
VARTYPEATT, 1179
WEIGHTATT, 1180
YYLTYPE, 1205
YYSTYPE, 1205
yyltype, 1205
yystype, 1205
yytokentype, 1205
OSParseosrl.tab.hpp
ABSEND, 1273
ABSSTART, 1273
ACTUALSTARTTIMEEND, 1255
ALLDIFFEND, 1274
ALLDIFFSTART, 1274
ATEQUALITYEND, 1255
ATEQUALITYSTART, 1255
ATLOWEREND, 1255
ATLOWERSTART, 1255
ATTRIBUTETEXT, 1249
ATUPPEREND, 1255
ATUPPERSTART, 1255
AVAILABLEMEMORYEND, 1256
BASE64END, 1256
BASE64START, 1256
BASEMATRIXEND, 1266
BASEMATRIXIDXATT, 1267
BASEMATRIXSTART, 1266
BASETRANPOSEATT, 1268
BASICEND, 1256
BASICSTART, 1256
BASISSTATUSEND, 1256
BASISSTATUSSTART, 1256
BASSTATUSEND, 1256
BASSTATUSSTART, 1256
BLOCKCOLIDXATT, 1270
BLOCKEND, 1266
BLOCKROWIDXATT, 1270
BLOCKSEND, 1266
BLOCKSSTART, 1266
BLOCKSTART, 1266
CATEGORYATT, 1252
COEFATT, 1267
COLOFFSETSEND, 1270
COLOFFSETSTART, 1270
CONEND, 1257
CONSTANTATT, 1266
CONSTANT, 1256
CONSTRAINTSEND, 1257
CONSTRAINTSSTART, 1257
CONTYPEATT, 1252
COSEND, 1273
COSSTART, 1273
CURRENTJOBCOUNTEND, 1257
CURRENTSTATEEND, 1257
CURRENTSTATESTART, 1257
DESCRIPTIONATT, 1252
DIVIDEEND, 1272
DIVIDESTART, 1272

DOUBLE, 1249
DUALVALUESEND, 1257
DUALVALUESSTART, 1257
DUMMY, 1270
EEND, 1274
ELEMENTSEND, 1268
ELEMENTSSTART, 1268
ELEMENTTEXT, 1249
EEND, 1257
ELSTART, 1257
EMPTYCATEGORYATT, 1252
EMPTYCONTYPEATT, 1253
EMPTYENUMTYPEATT, 1253
EMPTYNAMEATT, 1252
EMPTYOBJTYPEATT, 1253
EMPTYROWMAJORATT, 1270
EMPTYSHAPEATT, 1266
EMPTYSYMMETRYATT, 1266
EMPTYTYPEATT, 1252
EMPTYUNITATT, 1253
EMPTYVALUEATT, 1253
EMPTYVARTYPEATT, 1253
ENDOFELEMENT, 1250
ENDTIMEEND, 1258
ENDTimestart, 1257
ENUMERATIONEND, 1257
ENUMERATIONSTART, 1257
ENUMTYPEATT, 1253
ERFEND, 1273
ERFSTART, 1273
ESTART, 1274
EXPEND, 1272
EXPEND, 1271
EXPRSTART, 1271
EXPSTART, 1272
FILECREATOREMPTY, 1265
FILECREATOREND, 1265
FILECREATORSTART, 1265
FILEDESCRIPTIONEND, 1265
FILELICENCEEMPTY, 1265
FILELICENCEEND, 1265
FILELICENCESTART, 1265
FILENAMEEMPTY, 1264
FILENAMEEND, 1264
FILENAMESTART, 1264
FILESOURCEEMPTY, 1264
FILESOURCEEND, 1264
FILESOURCESTART, 1264
GENERALELEMENTSEND, 1269
GENERALEND, 1254
GENERALSTART, 1254
GENERALSTATUSEND, 1258
GENERALSTATUSSTART, 1258
GREATERTHAN, 1250
HEADEREND, 1254
HEADERSTART, 1254
IDATT, 1277
IDENTITYMATRIXEND, 1277
IDXATT, 1252
IDXEND, 1258
IDXSTART, 1258
IFEND, 1273
IFSTART, 1273
INCLUDEDIAGONALATT, 1277
INCRATT, 1252
INDEXESEND, 1268
INDEXESSTART, 1268
INSTANCENAMEEND, 1258
INSTANCENAMESTART, 1258
INTEGER, 1249
ISFREEEND, 1258
ISFREESTART, 1258
ITEMEMPTY, 1255
ITEMEND, 1255
ITEMSTART, 1254
ITEMSTARTANDEND, 1255
ITEMTEXT, 1249
JOBEND, 1254
JOBIDEND, 1258
JOBIDSTART, 1258
JOBSTART, 1254
LINEARELEMENTSEND, 1269
LNEND, 1272
LNSTART, 1272
MATRIXCONEND, 1264
MATRIXCONSTART, 1264
MATRIXDIAGONALEND, 1275
MATRIXDOTTIMESEND, 1275
MATRIXEND, 1266
MATRIXINVERSEEND, 1277
MATRIXINVERSESTART, 1277
MATRIXMERGEEND, 1276
MATRIXMERGESTART, 1276
MATRIXMINUSEND, 1276
MATRIXMINUSSTART, 1276
MATRIXNEGATEEND, 1276
MATRIXNEGATESTART, 1276
MATRIXOBJEND, 1264
MATRIXOBJSTART, 1264
MATRIXPLUSEND, 1276
MATRIXPLUSSTART, 1276
MATRIXPRODUCTEND, 1276
MATRIXPRODUCTSTART, 1276
MATRIXREFERENCEEND, 1277
MATRIXSTART, 1265
MATRIXTERMEND, 1271
MATRIXTERMSTART, 1271
MATRIXTIMESEND, 1276

MATRIXTIMESSTART, 1276
MATRIXTOSCALAREND, 1275
MATRIXTRACEEND, 1275
MATRIXTRACESTART, 1275
MATRIXTRANPOSEEND, 1277
MATRIXVAREND, 1264
MATRIXVARSTART, 1264
MAXEND, 1274
MAXSTART, 1274
MESSAGEEND, 1259
MESSAGESTART, 1258
MINEND, 1274
MINSTART, 1274
MINUSEND, 1271
MINUSSTART, 1271
MULTATT, 1252
NAMEATT, 1252
NEGATEEND, 1272
NEGATESTART, 1272
NEGATIVEPATTERNATT, 1266
NLEND, 1270
NLSTART, 1270
NONZEROSSEND, 1268
NONZEROSSTART, 1268
NUMBEREND, 1274
NUMBEROFBLOCKSATT, 1267
NUMBEROFCOLUMNSATT, 1267
NUMBEROFCONATT, 1250
NUMBEROFELATT, 1250
NUMBEROFEXPR, 1271
NUMBEROFIDXATT, 1250
NUMBEROFITEMSATT, 1250
NUMBEROFOBJATT, 1251
NUMBEROFROWSATT, 1267
NUMBEROFTIMESATT, 1251
NUMBEROFVALUESATT, 1267
NUMBEROFVARATT, 1251
NUMBEROFVARIDXATT, 1251
NUMBERSTART, 1274
OBJECTIVESEND, 1259
OBJECTIVESSTART, 1259
OBJEND, 1259
OBJSTART, 1259
OBJTYPEATT, 1253
OPTIMIZATIONEND, 1254
OPTIMIZATIONSTART, 1254
OSRLATTRIBUTETEXT, 1250
OSRLEND, 1250
OSRLSTART, 1250
OSRLSTARTEMPTY, 1250
OTHEREND, 1259
OTHERRESULTSEND, 1259
OTHERRESULTSSTART, 1259
OTHERSTART, 1259
PATTERNELEMENTSEND, 1269
PIEND, 1274
PISTART, 1274
PLUSEND, 1271
PLUSSTART, 1271
POWEREND, 1271
POWERSTART, 1271
PRODUCTEND, 1272
PRODUCTSTART, 1272
QUOTE, 1250
ROWMAJORATT, 1270
ROWOFFSETSEND, 1270
ROWOFFSETSSTART, 1270
SERVICEEND, 1254
SERVICENAMEEND, 1260
SERVICENAMESTART, 1260
SERVICESTART, 1254
SERVICEURIEND, 1260
SERVICEURISTART, 1260
SHAPEATT, 1266
SINEND, 1273
SINSTART, 1273
SIZEOFATT, 1252
SOLUTIONEND, 1261
SOLUTIONSTART, 1260
SOLVERINVOKEDEND, 1261
SOLVERINVOKEDSTART, 1261
SOLVEROUTPUTEND, 1261
SOLVEROUTPUTSTART, 1261
SQRTEND, 1272
SQRTSTART, 1272
SQUAREEND, 1273
SQUARESTART, 1273
STARTVECTOREND, 1268
STARTVECTORSTART, 1268
STATUSEND, 1261
STATUSSTART, 1261
SUBMITTIMEEND, 1261
SUBMITTIMESTART, 1261
SUBSTATUSEND, 1261
SUBSTATUSSTART, 1261
SUMEND, 1272
SUMSTART, 1272
SUPERBASICEND, 1261
SUPERBASICSTART, 1261
SYMMETRYATT, 1266
SYSTEMEND, 1254
SYSTEMSTART, 1254
TIMEEND, 1262
TIMESEND, 1274
TIMESSTART, 1274
TIMESTAMPEND, 1262
TIMESTAMPSTART, 1262
TIMESTART, 1262

- TOTALJOBSSOFAREND, [1262](#)
- TRANSFORMATIONEND, [1269](#)
- TWOQUOTES, [1250](#)
- TYPEATT, [1252](#)
- UNITATT, [1253](#)
- UNKNOWNEND, [1262](#)
- UNKNOWNSTART, [1262](#)
- USEDPCPUNUMBEREND, [1263](#)
- USEDPCPUNUMBERSTART, [1262](#)
- USEDPCPUSPEEDEND, [1263](#)
- USEDPCPUSPEEDSTART, [1263](#)
- USEDDISKSPACEEND, [1263](#)
- USEDDISKSPACESTART, [1263](#)
- USEDMEMORYEND, [1263](#)
- USEDMEMORYSTART, [1263](#)
- VALUEATT, [1253](#)
- VALUESEND, [1268](#)
- VALUESSTART, [1268](#)
- VALUESSTRINGEND, [1263](#)
- VALUESSTRINGSTART, [1263](#)
- VALUETYPEATT, [1269](#)
- VAREND, [1263](#)
- VARIABLEEND, [1273](#)
- VARIABLESEND, [1263](#)
- VARIABLESSTART, [1263](#)
- VARIABLESTART, [1273](#)
- VARIDXEND, [1264](#)
- VARIDXSTART, [1263](#)
- VARSTART, [1263](#)
- VARTYPEATT, [1253](#)
- YYLTYPE, [1278](#)
- YYSTYPE, [1278](#)
- yyltype, [1278](#)
- yystype, [1278](#)
- yytokentype, [1278](#)
- OSPrint
 - OSOutput, [723](#)
- OSPrintf
 - OSOutputChannel, [727](#)
- OSRLATTRIBUTETEXT
 - OSParseosrl.tab.hpp, [1250](#)
- OSRLEND
 - OSParseosrl.tab.hpp, [1250](#)
- OSRLSTART
 - OSParseosrl.tab.hpp, [1250](#)
- OSRLSTARTEMPTY
 - OSParseosrl.tab.hpp, [1250](#)
- OSRand
 - OSMathUtil.h, [1326](#)
- OSReferencedObject, [727](#)
 - ~OSReferencedObject, [729](#)
 - AddRef, [730](#)
 - OSReferencedObject, [729](#)
 - OSReferencedObject, [729](#)
- ReferenceCount, [729](#)
- ReleaseRef, [730](#)
- OSReferencer, [730](#)
- OSResult, [730](#)
 - ~OSResult, [744](#)
 - addTimingInformation, [771](#)
 - dualVals, [811](#)
 - general, [810](#)
 - getActualStartTime, [749](#)
 - getAnotherVariableResultNumberOfVar, [753](#)
 - getAvailableCPUNumberDescription, [748](#)
 - getAvailableCPUNumberValue, [748](#)
 - getAvailableCPUSpeedDescription, [748](#)
 - getAvailableCPUSpeedUnit, [748](#)
 - getAvailableCPUSpeedValue, [748](#)
 - getAvailableDiskSpaceDescription, [748](#)
 - getAvailableDiskSpaceUnit, [748](#)
 - getAvailableDiskSpaceValue, [748](#)
 - getAvailableMemoryDescription, [748](#)
 - getAvailableMemoryUnit, [748](#)
 - getAvailableMemoryValue, [748](#)
 - getBasisInformationDense, [753](#)
 - getBasisStatusEI, [753](#)
 - getBasisStatusNumberOfEI, [752](#)
 - getConstraintNumber, [750](#)
 - getCurrentJobCount, [748](#)
 - getCurrentState, [748](#)
 - getDualValue, [757](#)
 - getDualValueIdx, [757](#)
 - getDualValueName, [757](#)
 - getGeneralMessage, [746](#)
 - getGeneralStatus, [745](#)
 - getGeneralStatusDescription, [745](#)
 - getGeneralStatusType, [745](#)
 - getGeneralSubstatusDescription, [746](#)
 - getGeneralSubstatusName, [746](#)
 - getInstanceName, [747](#)
 - getJobEndTime, [749](#)
 - getJobID, [747](#)
 - getJobStatus, [748](#)
 - getJobSubmitTime, [749](#)
 - getNumberOfDualValues, [757](#)
 - getNumberOfGeneralSubstatuses, [746](#)
 - getNumberOfObjValues, [755](#)
 - getNumberOfOtherConstraintResults, [758](#)
 - getNumberOfOtherGeneralResults, [747](#)
 - getNumberOfOtherJobResults, [750](#)
 - getNumberOfOtherObjectiveResults, [756](#)
 - getNumberOfOtherServiceResults, [748](#)
 - getNumberOfOtherSolutionResults, [760](#)
 - getNumberOfOtherSystemResults, [748](#)
 - getNumberOfOtherVariableResults, [753](#)
 - getNumberOfPrimalVariableValues, [752](#)
 - getNumberOfSolutionSubstatuses, [751](#)

getNumberOfSolverOutputs, 760
getNumberOfTimes, 749
getNumberOfVarValues, 752
getNumberOfVarValuesString, 752
getObjValue, 755
getObjValueIdx, 755
getObjValueName, 755
getObjectiveNumber, 750
getOptimalDualVariableValues, 757
getOptimalObjValue, 755
getOptimalPrimalVariableValues, 752
getOtherConstraintResultArrayDense, 759
getOtherConstraintResultArrayType, 758
getOtherConstraintResultCon, 758
getOtherConstraintResultConIdx, 758
getOtherConstraintResultDescription, 758
getOtherConstraintResultEnumerationDescription, 759
getOtherConstraintResultEnumerationEI, 759
getOtherConstraintResultEnumerationNumberOfEI, 759
getOtherConstraintResultEnumerationValue, 758
getOtherConstraintResultName, 758
getOtherConstraintResultNumberOfCon, 758
getOtherConstraintResultNumberOfEnumerations, 758
getOtherConstraintResultType, 758
getOtherConstraintResultValue, 758
getOtherGeneralResultDescription, 748
getOtherGeneralResultName, 747
getOtherGeneralResultValue, 748
getOtherJobResultDescription, 750
getOtherJobResultName, 750
getOtherJobResultValue, 750
getOtherObjectiveResultArrayDense, 757
getOtherObjectiveResultArrayType, 756
getOtherObjectiveResultDescription, 756
getOtherObjectiveResultEnumerationDescription, 756
getOtherObjectiveResultEnumerationEI, 757
getOtherObjectiveResultEnumerationNumberOfEI, 757
getOtherObjectiveResultEnumerationValue, 756
getOtherObjectiveResultName, 756
getOtherObjectiveResultNumberOfEnumerations, 756
getOtherObjectiveResultNumberOfObj, 756
getOtherObjectiveResultObj, 756
getOtherObjectiveResultObjIdx, 756
getOtherObjectiveResultType, 756
getOtherObjectiveResultValue, 756
getOtherServiceResultDescription, 748
getOtherServiceResultName, 748
getOtherServiceResultValue, 748
getOtherSolutionResultCategory, 760
getOtherSolutionResultDescription, 760
getOtherSolutionResultItem, 760
getOtherSolutionResultName, 760
getOtherSolutionResultNumberOfItems, 760
getOtherSolutionResultValue, 760
getOtherSystemResultDescription, 748
getOtherSystemResultName, 748
getOtherSystemResultValue, 748
getOtherVariableResultArrayDense, 755
getOtherVariableResultArrayType, 754
getOtherVariableResultDescription, 753
getOtherVariableResultEnumerationDescription, 754
getOtherVariableResultEnumerationEI, 755
getOtherVariableResultEnumerationNumberOfEI, 754
getOtherVariableResultEnumerationValue, 754
getOtherVariableResultName, 753
getOtherVariableResultNumberOfEnumerations, 753
getOtherVariableResultNumberOfVar, 753
getOtherVariableResultType, 753
getOtherVariableResultValue, 753
getOtherVariableResultVar, 754
getOtherVariableResultVarIdx, 754
getScheduledStartTime, 749
getServiceName, 746
getServiceURI, 746
getServiceUtilization, 748
getSolutionMessage, 752
getSolutionNumber, 750
getSolutionStatus, 750
getSolutionStatusDescription, 751
getSolutionStatusType, 751
getSolutionSubstatusDescription, 751
getSolutionSubstatusType, 751
getSolutionTargetObjectiveIdx, 751
getSolutionTargetObjectiveName, 751
getSolutionWeightedObjectives, 751
getSolverInvoked, 747
getSolverOutputCategory, 760
getSolverOutputDescription, 760
getSolverOutputItem, 760
getSolverOutputName, 760
getSolverOutputNumberOfItems, 760
getSystemInformation, 748
getTimeNumber, 749
getTimeServiceStarted, 748
getTimeStamp, 747
getTimeValue, 749
getTimingInfoCategory, 749
getTimingInfoDescription, 749
getTimingInfoType, 749
getTimingInfoUnit, 749
getTimingInfoValue, 749

getTotalJobsSoFar, 748
getUsedCPUNumberDescription, 749
getUsedCPUNumberValue, 750
getUsedCPUSpeedDescription, 749
getUsedCPUSpeedUnit, 749
getUsedCPUSpeedValue, 749
getUsedDiskSpaceDescription, 749
getUsedDiskSpaceUnit, 749
getUsedDiskSpaceValue, 749
getUsedMemoryDescription, 749
getUsedMemoryUnit, 749
getUsedMemoryValue, 749
getVarValue, 752
getVarValueIdx, 752
getVarValueName, 752
getVarValueString, 752
getVarValueStringIdx, 752
getVarValueStringName, 752
getVariableNumber, 750
isEqual, 745
job, 810
m_iConstraintNumber, 810
m_iNumberOfOtherVariableResults, 810
m_iObjectiveNumber, 810
m_iVariableNumber, 810
m_mdDualValues, 811
m_mdPrimalValues, 811
OSResult, 744
optimization, 810
OSResult, 744
primalVals, 811
resultHeader, 810
service, 810
setActualStartTime, 770
setAnOtherSolutionResult, 808
setAnOtherVariableResultDense, 784, 785
setAnOtherVariableResultSparse, 783, 784
setAvailableCPUNumberDescription, 766
setAvailableCPUNumberValue, 767
setAvailableCPUSpeedDescription, 766
setAvailableCPUSpeedUnit, 766
setAvailableCPUSpeedValue, 766
setAvailableDiskSpaceDescription, 764
setAvailableDiskSpaceUnit, 764
setAvailableDiskSpaceValue, 765
setAvailableMemoryDescription, 765
setAvailableMemoryUnit, 765
setAvailableMemoryValue, 765
setBasisStatus, 782
setConstraintNumber, 776
setConstraintValuesDense, 800
setCurrentJobCount, 768
setCurrentState, 768
setDualValue, 800
setDualVariableValuesDense, 800
setDualVariableValuesSparse, 799
setGeneralMessage, 761
setGeneralStatus, 760
setGeneralStatusDescription, 761
setGeneralStatusType, 760
setGeneralSubstatusDescription, 761
setGeneralSubstatusName, 761
setHeader, 745
setInstanceName, 762
setJobEndTime, 771
setJobID, 762
setJobStatus, 770
setJobSubmitTime, 770
setNumberOfDualValues, 798
setNumberOfDualVariableValues, 799
setNumberOfGeneralSubstatuses, 760
setNumberOfObjValues, 791
setNumberOfObjectiveValues, 792
setNumberOfOtherConstraintResults, 798
setNumberOfOtherGeneralResults, 763
setNumberOfOtherJobResults, 775
setNumberOfOtherObjectiveResults, 791
setNumberOfOtherServiceResults, 769
setNumberOfOtherSolutionResults, 805
setNumberOfOtherSystemResults, 767
setNumberOfOtherVariableResults, 783
setNumberOfPrimalVariableValues, 780
setNumberOfSolutionSubstatuses, 777
setNumberOfSolverOutputs, 808
setNumberOfTimes, 772
setNumberOfVarValues, 781
setNumberOfVarValuesString, 782
setObjValue, 793
setObjectiveNumber, 776
setObjectiveValuesDense, 792
setObjectiveValuesSparse, 792
setOtherConstraintResultCon, 805
setOtherConstraintResultConIdx, 804
setOtherConstraintResultConName, 805
setOtherConstraintResultConType, 802
setOtherConstraintResultDescription, 804
setOtherConstraintResultEnumType, 803
setOtherConstraintResultName, 801
setOtherConstraintResultNumberOfCon, 801
setOtherConstraintResultNumberOfEnumerations, 801
setOtherConstraintResultType, 802
setOtherConstraintResultValue, 803
setOtherGeneralResultDescription, 764
setOtherGeneralResultName, 763
setOtherGeneralResultValue, 763
setOtherJobResultDescription, 775
setOtherJobResultName, 775

- setOtherJobResultValue, [775](#)
- setOtherObjectiveResultDescription, [796](#)
- setOtherObjectiveResultEnumType, [795](#)
- setOtherObjectiveResultName, [794](#)
- setOtherObjectiveResultNumberOfEnumerations, [794](#)
- setOtherObjectiveResultNumberOfObj, [793](#)
- setOtherObjectiveResultObj, [798](#)
- setOtherObjectiveResultObjIdx, [797](#)
- setOtherObjectiveResultObjName, [797](#)
- setOtherObjectiveResultObjType, [795](#)
- setOtherObjectiveResultType, [794](#)
- setOtherObjectiveResultValue, [796](#)
- setOtherOptionEnumeration, [790](#)
- setOtherServiceResultDescription, [770](#)
- setOtherServiceResultName, [769](#)
- setOtherServiceResultValue, [769](#)
- setOtherSolutionResultCategory, [807](#)
- setOtherSolutionResultDescription, [807](#)
- setOtherSolutionResultItem, [807](#)
- setOtherSolutionResultName, [806](#)
- setOtherSolutionResultNumberOfItems, [807](#)
- setOtherSolutionResultValue, [806](#)
- setOtherSystemResultDescription, [767](#)
- setOtherSystemResultName, [767](#)
- setOtherSystemResultValue, [767](#)
- setOtherVariableResultDescription, [788](#)
- setOtherVariableResultEnumType, [788](#)
- setOtherVariableResultName, [786](#)
- setOtherVariableResultNumberOfEnumerations, [786](#)
- setOtherVariableResultNumberOfVar, [786](#)
- setOtherVariableResultType, [787](#)
- setOtherVariableResultValue, [788](#)
- setOtherVariableResultVar, [790](#)
- setOtherVariableResultVarIdx, [789](#)
- setOtherVariableResultVarName, [789](#)
- setOtherVariableResultVarType, [787](#)
- setPrimalVariableValuesDense, [780](#)
- setPrimalVariableValuesSparse, [780](#)
- setRandom, [745](#)
- setScheduledStartTime, [770](#)
- setServiceName, [762](#)
- setServiceURI, [762](#)
- setServiceUtilization, [769](#)
- setSolutionMessage, [779](#)
- setSolutionNumber, [776](#)
- setSolutionStatus, [777](#)
- setSolutionStatusDescription, [778](#)
- setSolutionStatusType, [777](#)
- setSolutionSubstatusDescription, [778](#)
- setSolutionSubstatusType, [778](#)
- setSolutionTargetObjectiveIdx, [778](#)
- setSolutionTargetObjectiveName, [779](#)
- setSolutionWeightedObjectives, [779](#)
- setSolverInvoked, [763](#)
- setSolverOutputCategory, [809](#)
- setSolverOutputDescription, [809](#)
- setSolverOutputItem, [809](#)
- setSolverOutputName, [808](#)
- setSolverOutputNumberOfItems, [809](#)
- setSystemInformation, [764](#)
- setTime, [771](#)
- setTimeNumber, [772](#)
- setTimeServiceStarted, [768](#)
- setTimeStamp, [763](#)
- setTimingInformation, [771](#)
- setTotalJobsSoFar, [768](#)
- setUsedCPUNumberDescription, [774](#)
- setUsedCPUNumberValue, [775](#)
- setUsedCPUSpeedDescription, [774](#)
- setUsedCPUSpeedUnit, [774](#)
- setUsedCPUSpeedValue, [774](#)
- setUsedDiskSpaceDescription, [772](#)
- setUsedDiskSpaceUnit, [772](#)
- setUsedDiskSpaceValue, [773](#)
- setUsedMemoryDescription, [773](#)
- setUsedMemoryUnit, [773](#)
- setUsedMemoryValue, [773](#)
- setVarValue, [781](#)
- setVarValueString, [782](#)
- setVariableNumber, [776](#)
- system, [810](#)
- OSRunSolver.h
 - runSolver, [1318](#)
 - selectSolver, [1319](#)
- OSSmartPtr
 - ~OSSmartPtr, [834](#)
 - ConstPtr, [836](#)
 - GetRawPtr, [836](#)
 - IsNull, [836](#)
 - IsValid, [836](#)
 - OSSmartPtr, [834](#)
 - operator*, [835](#)
 - operator->, [835](#)
 - operator=, [835](#)
 - operator==, [835](#)
 - OSSmartPtr, [834](#)
- OSSmartPtr< T >, [830](#)
- OSSmartPtr.hpp
 - ComparePointers, [1353](#)
 - ConstPtr, [1352](#)
 - GetRawPtr, [1352](#)
 - IsNull, [1352](#)
 - IsValid, [1352](#)
 - operator==, [1352](#)
- OSSolverAgent, [836](#)
 - ~OSSolverAgent, [838](#)
 - fileUpload, [840](#)

- getJobID, 838
- kill, 839
- knock, 839
- OSSolverAgent, 838
- OSSolverAgent, 838
- retrieve, 839
- send, 839
- solve, 838
- OSStringUtil.h
 - makeStringFromInt, 1354
 - writeStringData, 1354
- OSWSUtil.h
 - RCVBUFSIZE, 1045
- OSdtoa.h
 - os_dtoa, 1321
 - os_freedtoa, 1321
 - os_strtod, 1321
- OSgLParseData, 406
 - ~OSgLParseData, 409
 - baseMatrixEndCol, 413
 - baseMatrixEndColPresent, 413
 - baseMatrixEndRow, 413
 - baseMatrixEndRowPresent, 413
 - baseMatrixIdx, 412
 - baseMatrixIdxPresent, 413
 - baseMatrixStartCol, 413
 - baseMatrixStartColPresent, 413
 - baseMatrixStartRow, 413
 - baseMatrixStartRowPresent, 413
 - baseTranspose, 413
 - baseTransposePresent, 413
 - blockColIdx, 414
 - blockColIdxPresent, 414
 - blockRowIdx, 414
 - blockRowIdxPresent, 414
 - colOffsets, 411
 - description, 410
 - descriptionPresent, 410
 - errorText, 410
 - fileCreator, 410
 - fileCreatorPresent, 410
 - fileName, 410
 - fileNamePresent, 410
 - idx, 412
 - idxPresent, 412
 - ignoreDataAfterErrors, 411
 - licence, 410
 - licencePresent, 410
 - matrix, 411
 - matrixBlockNumberOfCols, 415
 - matrixBlockNumberOfRows, 415
 - matrixCounter, 411
 - mtxBlkVec, 411
 - mtxBlocksVec, 411
 - mtxConstructorVec, 411
 - name, 412
 - namePresent, 412
 - numberOfBlocks, 412
 - numberOfBlocksPresent, 414
 - numberOfColumns, 412
 - numberOfColumnsPresent, 414
 - numberOfEI, 415
 - numberOfEIPresent, 415
 - numberOfMatrices, 411
 - numberOfRows, 412
 - numberOfRowsPresent, 414
 - numberOfValues, 415
 - numberOfValuesPresent, 414
 - numberOfVarIdx, 415
 - numberOfVarIdxPresent, 415
 - OSgLParseData, 409
 - osglCoef, 414
 - osglCoefPresent, 414
 - osglConstantPresent, 414
 - osglCounter, 409
 - osglDblArray, 409
 - osglIncr, 409
 - osglIncrPresent, 409
 - osglIntArray, 409
 - osglMult, 409
 - osglMultPresent, 409
 - osglNonzeroCounter, 415
 - osglNumberOfEI, 409
 - osglNumberOfEIPresent, 409
 - osglNumberOfNonzeros, 415
 - osglSize, 409
 - osglTempint, 409
 - osglValArray, 409
 - OSgLParseData, 409
 - parser_errors, 410
 - rowMajor, 414
 - rowMajorPresent, 414
 - rowOffsets, 411
 - scalarMultiplier, 413
 - scalarMultiplierPresent, 414
 - scanner, 410
 - shape, 415
 - shapePresent, 415
 - source, 410
 - sourcePresent, 410
 - suppressFurtherErrorMessages, 411
 - symmetry, 412
 - symmetryPresent, 412
 - targetMatrixFirstCol, 412
 - targetMatrixFirstColPresent, 413
 - targetMatrixFirstRow, 412
 - targetMatrixFirstRowPresent, 413
 - tempC, 411

- type, 412
- typePresent, 412
- valueType, 415
- valueTypePresent, 415
- OSglParserData.h
 - osgl_empty_vectors, 1085
- OSglWriter.h
 - writeBasisStatus, 1054
 - writeDbfVectorData, 1054
 - writeGeneralFileHeader, 1054
 - writeIntVectorData, 1053
 - writeOtherOptionEnumeration, 1054
- OSgams2osil, 404
 - ~OSgams2osil, 405
 - createOSInstance, 405
 - getOSInstance, 405
 - initGMO, 405
 - OSgams2osil, 405
 - OSgams2osil, 405
 - osinstance, 405
 - takeOverOSInstance, 405
- OSgetVersionInfo
 - OSParameters.h, 1345
- OShL, 416
 - ~OShL, 417
 - getJobID, 417
 - kill, 418
 - knock, 418
 - OShL, 417
 - OShL, 417
 - retrieve, 418
 - send, 417
 - solve, 417
- OSiLParserData, 419
 - ~OSiLParserData, 423
 - axisDirection, 427
 - axisDirectionPresent, 427
 - conReferenceMatrixIdx, 430
 - conReferenceMatrixIdxPresent, 429
 - coneCounter, 426
 - constantMatrixIdx, 430
 - constantMatrixIdxPresent, 429
 - distortionMatrix, 427
 - distortionMatrixPresent, 427
 - elCounter, 427
 - errorText, 431
 - firstAxisDirection, 427
 - firstAxisDirectionPresent, 427
 - ignoreDataAfterErrors, 430
 - intervalhorizon, 426
 - intervalhorizonON, 425
 - intervalstart, 426
 - intervalstartON, 426
 - kount2, 430
 - kounter, 430
 - lbConelIdx, 429
 - lbConelIdxPresent, 428
 - lbMatrixIdx, 429
 - lbMatrixIdxPresent, 428
 - m_miConStageInfo, 425
 - m_miObjStageInfo, 425
 - m_miVarStageInfo, 425
 - matrixIdx, 429
 - matrixIdxPresent, 428
 - matrixTermInObj, 430
 - name, 427
 - namePresent, 427
 - nconcovered, 425
 - normScaleFactor, 427
 - normScaleFactorPresent, 427
 - numberOf, 427
 - numberOfColumns, 426
 - numberOfColumnsPresent, 426
 - numberOfCones, 426
 - numberOfConesPresent, 426
 - numberOfEI, 426
 - numberOfMatrices, 426
 - numberOfMatricesPresent, 426
 - numberOfMatrixCon, 428
 - numberOfMatrixExpr, 428
 - numberOfMatrixObj, 428
 - numberOfMatrixTerms, 428
 - numberOfMatrixTermsPresent, 428
 - numberOfMatrixVar, 428
 - numberOfRows, 426
 - numberOfRowsPresent, 426
 - nvarcovered, 425
 - OSiLParserData, 423
 - objReferenceMatrixIdx, 430
 - objReferenceMatrixIdxPresent, 429
 - orderConelIdx, 429
 - orderConelIdxPresent, 429
 - osillينو, 423
 - OSiLParserData, 423
 - parser_errors, 431
 - qtermcoefattON, 423
 - qtermcount, 423
 - qtermidattON, 423
 - qtermidxOneattON, 423
 - qtermidxTwoattON, 423
 - qtermidxattON, 423
 - referenceMatrixIdx, 427
 - referenceMatrixIdxPresent, 427
 - scanner, 423
 - secondAxisDirection, 428
 - secondAxisDirectionPresent, 428
 - semidefiniteness, 428
 - semidefinitenessPresent, 428

- shape, 430
- shapePresent, 430
- stageConstraintStartIdx, 425
- stageConstraintsON, 424
- stageConstraintsOrdered, 424
- stageObjectiveStartIdx, 425
- stageObjectivesON, 424
- stageObjectivesOrdered, 424
- stageVariableStartIdx, 424
- stageVariablesON, 424
- stageVariablesOrdered, 424
- stageconcount, 425
- stagecount, 424
- stagename, 424
- stagenameON, 424
- stageobjcount, 425
- stagevarcount, 425
- suppressFurtherErrorMessages, 430
- tempVal, 430
- templateMatrixIdx, 430
- templateMatrixIdxPresent, 429
- timeDomainInterval, 424
- timeDomainStages, 424
- ubConelIdx, 429
- ubConelIdxPresent, 429
- ubMatrixIdx, 429
- ubMatrixIdxPresent, 428
- varReferenceMatrixIdx, 430
- varReferenceMatrixIdxPresent, 429
- varType, 430
- varTypePresent, 429
- OSiLReader, 431
 - ~OSiLReader, 431
 - OSiLReader, 431
 - OSiLReader, 431
 - readOSiL, 432
- OSiLWriter, 432
 - ~OSiLWriter, 433
 - m_bWhiteSpace, 433
 - m_bWriteBase64, 433
 - m_sB64encoded, 434
 - OSiLWriter, 433
 - OSiLWriter, 433
 - writeOSiL, 433
- OSiRand
 - OSMathUtil.h, 1326
- OSmps2OS, 494
 - ~OSmps2OS, 496
 - createOSObjects, 496
 - jobID, 497
 - OSmps2OS, 496
 - osinstance, 497
 - OSmps2OS, 496
 - osol, 497
 - osolreader, 497
 - osoption, 497
 - setJobID, 496
 - setOsol, 496
- OSmps2osil, 497
 - ~OSmps2osil, 499
 - createOSInstance, 499
 - OSmps2osil, 499
 - osinstance, 499
 - OSmps2osil, 499
- OSnLMNode, 504
 - ~OSnLMNode, 507
 - copyNodeAndDescendants, 509
 - createExpressionTreeFromPostfix, 508
 - createExpressionTreeFromPrefix, 507
 - getPostfixFromExpressionTree, 508
 - getPrefixFromExpressionTree, 507
 - IsEqual, 509
 - OSnLMNode, 507
 - OSnLMNode, 507
 - postOrderOSnLNodeTraversal, 508
 - preOrderOSnLNodeTraversal, 508
- OSnLMNodeDiagonalMatrixFromVector, 509
 - ~OSnLMNodeDiagonalMatrixFromVector, 510
 - cloneExprNode, 511
 - getTokenName, 511
 - OSnLMNodeDiagonalMatrixFromVector, 510
 - OSnLMNodeDiagonalMatrixFromVector, 510
- OSnLMNodeIdentityMatrix, 511
 - ~OSnLMNodeIdentityMatrix, 512
 - cloneExprNode, 513
 - getTokenName, 513
 - OSnLMNodeIdentityMatrix, 512
 - OSnLMNodeIdentityMatrix, 512
- OSnLMNodeMatrixCon, 513
 - ~OSnLMNodeMatrixCon, 515
 - cloneExprNode, 515
 - getNonlinearExpressionInXML, 515
 - getTokenName, 515
 - getTokenNumber, 515
 - idx, 515
 - IsEqual, 515
 - OSnLMNodeMatrixCon, 515
 - OSnLMNodeMatrixCon, 515
- OSnLMNodeMatrixDiagonal, 516
 - ~OSnLMNodeMatrixDiagonal, 517
 - cloneExprNode, 517
 - getTokenName, 517
 - OSnLMNodeMatrixDiagonal, 517
 - OSnLMNodeMatrixDiagonal, 517
- OSnLMNodeMatrixDotTimes, 517
 - ~OSnLMNodeMatrixDotTimes, 519
 - cloneExprNode, 519
 - getTokenName, 519

- OSnLMNodeMatrixDotTimes, 519
- OSnLMNodeMatrixDotTimes, 519
- OSnLMNodeMatrixInverse, 519
 - ~OSnLMNodeMatrixInverse, 521
 - cloneExprNode, 521
 - getTokenName, 521
 - OSnLMNodeMatrixInverse, 521
 - OSnLMNodeMatrixInverse, 521
- OSnLMNodeMatrixLowerTriangle, 521
 - ~OSnLMNodeMatrixLowerTriangle, 523
 - cloneExprNode, 523
 - getNonlinearExpressionInXML, 523
 - getTokenName, 523
 - includeDiagonal, 524
 - IsEqual, 523
 - OSnLMNodeMatrixLowerTriangle, 523
 - OSnLMNodeMatrixLowerTriangle, 523
- OSnLMNodeMatrixMinus, 524
 - ~OSnLMNodeMatrixMinus, 525
 - cloneExprNode, 526
 - getTokenName, 526
 - OSnLMNodeMatrixMinus, 525
 - OSnLMNodeMatrixMinus, 525
- OSnLMNodeMatrixNegate, 526
 - ~OSnLMNodeMatrixNegate, 527
 - cloneExprNode, 528
 - getTokenName, 528
 - OSnLMNodeMatrixNegate, 527
 - OSnLMNodeMatrixNegate, 527
- OSnLMNodeMatrixObj, 528
 - ~OSnLMNodeMatrixObj, 530
 - cloneExprNode, 530
 - getNonlinearExpressionInXML, 530
 - getTokenName, 530
 - getTokenNumber, 530
 - idx, 530
 - IsEqual, 530
 - OSnLMNodeMatrixObj, 530
 - OSnLMNodeMatrixObj, 530
- OSnLMNodeMatrixPlus, 531
 - ~OSnLMNodeMatrixPlus, 532
 - cloneExprNode, 532
 - getTokenName, 532
 - OSnLMNodeMatrixPlus, 532
 - OSnLMNodeMatrixPlus, 532
- OSnLMNodeMatrixProduct, 532
 - ~OSnLMNodeMatrixProduct, 534
 - cloneExprNode, 534
 - getTokenName, 534
 - OSnLMNodeMatrixProduct, 534
 - OSnLMNodeMatrixProduct, 534
- OSnLMNodeMatrixReference, 535
 - ~OSnLMNodeMatrixReference, 537
 - cloneExprNode, 537
- getNonlinearExpressionInXML, 537
- getTokenName, 537
- getTokenNumber, 537
- idx, 537
- IsEqual, 537
- OSnLMNodeMatrixReference, 537
- OSnLMNodeMatrixReference, 537
- OSnLMNodeMatrixScalarTimes, 538
 - ~OSnLMNodeMatrixScalarTimes, 539
 - cloneExprNode, 539
 - getTokenName, 539
 - OSnLMNodeMatrixScalarTimes, 539
 - OSnLMNodeMatrixScalarTimes, 539
- OSnLMNodeMatrixSubmatrixAt, 539
 - ~OSnLMNodeMatrixSubmatrixAt, 541
 - cloneExprNode, 541
 - getTokenName, 541
 - OSnLMNodeMatrixSubmatrixAt, 541
 - OSnLMNodeMatrixSubmatrixAt, 541
- OSnLMNodeMatrixSum, 541
 - ~OSnLMNodeMatrixSum, 543
 - cloneExprNode, 543
 - getTokenName, 543
 - OSnLMNodeMatrixSum, 543
 - OSnLMNodeMatrixSum, 543
- OSnLMNodeMatrixTimes, 543
 - ~OSnLMNodeMatrixTimes, 545
 - cloneExprNode, 545
 - getTokenName, 545
 - OSnLMNodeMatrixTimes, 545
 - OSnLMNodeMatrixTimes, 545
- OSnLMNodeMatrixTranspose, 545
 - ~OSnLMNodeMatrixTranspose, 547
 - cloneExprNode, 547
 - getTokenName, 547
 - OSnLMNodeMatrixTranspose, 547
 - OSnLMNodeMatrixTranspose, 547
- OSnLMNodeMatrixUpperTriangle, 547
 - ~OSnLMNodeMatrixUpperTriangle, 549
 - cloneExprNode, 549
 - getNonlinearExpressionInXML, 549
 - getTokenName, 549
 - includeDiagonal, 550
 - IsEqual, 549
 - OSnLMNodeMatrixUpperTriangle, 549
 - OSnLMNodeMatrixUpperTriangle, 549
- OSnLMNodeMatrixVar, 550
 - ~OSnLMNodeMatrixVar, 552
 - cloneExprNode, 552
 - getNonlinearExpressionInXML, 552
 - getTokenName, 552
 - getTokenNumber, 552
 - idx, 552
 - IsEqual, 552

- OSnLMNodeMatrixVar, 552
- OSnLMNodeMatrixVar, 552
- OSnLNode, 553
 - ~OSnLNode, 556
 - calculateFunction, 557
 - constructADTape, 557
 - copyNodeAndDescendants, 559
 - createExpressionTreeFromPostfix, 558
 - createExpressionTreeFromPrefix, 557
 - getPostfixFromExpressionTree, 558
 - getPrefixFromExpressionTree, 557
 - getVariableIndexMap, 556
 - isEqual, 559
 - m_ADTape, 559
 - m_dFunctionValue, 559
 - OSnLNode, 556
 - OSnLNode, 556
 - postOrderOSnLNodeTraversal, 558
 - preOrderOSnLNodeTraversal, 558
- OSnLNode.h
 - ADdouble, 1066
 - ADvector, 1066
- OSnLNodeAbs, 559
 - ~OSnLNodeAbs, 561
 - calculateFunction, 561
 - cloneExprNode, 562
 - constructADTape, 562
 - getTokenName, 561
 - OSnLNodeAbs, 561
 - OSnLNodeAbs, 561
- OSnLNodeAllDiff, 562
 - ~OSnLNodeAllDiff, 564
 - calculateFunction, 564
 - cloneExprNode, 565
 - constructADTape, 565
 - getTokenName, 564
 - OSnLNodeAllDiff, 564
 - OSnLNodeAllDiff, 564
- OSnLNodeCos, 565
 - ~OSnLNodeCos, 567
 - calculateFunction, 567
 - cloneExprNode, 568
 - constructADTape, 568
 - getTokenName, 567
 - OSnLNodeCos, 567
 - OSnLNodeCos, 567
- OSnLNodeDivide, 568
 - ~OSnLNodeDivide, 570
 - calculateFunction, 570
 - cloneExprNode, 571
 - constructADTape, 571
 - getTokenName, 570
 - OSnLNodeDivide, 570
 - OSnLNodeDivide, 570
- OSnLNodeE, 571
 - ~OSnLNodeE, 573
 - calculateFunction, 574
 - cloneExprNode, 574
 - constructADTape, 574
 - getNonlinearExpressionInXML, 574
 - getTokenName, 573
 - getTokenNumber, 573
 - OSnLNodeE, 573
 - OSnLNodeE, 573
- OSnLNodeErf, 575
 - ~OSnLNodeErf, 576
 - calculateFunction, 577
 - cloneExprNode, 577
 - constructADTape, 577
 - getTokenName, 577
 - OSnLNodeErf, 576
 - OSnLNodeErf, 576
- OSnLNodeExp, 577
 - ~OSnLNodeExp, 579
 - calculateFunction, 579
 - cloneExprNode, 580
 - constructADTape, 580
 - getTokenName, 579
 - OSnLNodeExp, 579
 - OSnLNodeExp, 579
- OSnLNodeIf, 580
 - ~OSnLNodeIf, 582
 - calculateFunction, 582
 - cloneExprNode, 583
 - constructADTape, 583
 - getTokenName, 582
 - OSnLNodeIf, 582
 - OSnLNodeIf, 582
- OSnLNodeLn, 583
 - ~OSnLNodeLn, 585
 - calculateFunction, 585
 - cloneExprNode, 586
 - constructADTape, 586
 - getTokenName, 585
 - OSnLNodeLn, 585
 - OSnLNodeLn, 585
- OSnLNodeMatrixDeterminant, 586
 - ~OSnLNodeMatrixDeterminant, 588
 - calculateFunction, 588
 - cloneExprNode, 589
 - constructADTape, 589
 - getTokenName, 588
 - OSnLNodeMatrixDeterminant, 588
 - OSnLNodeMatrixDeterminant, 588
- OSnLNodeMatrixToScalar, 589
 - ~OSnLNodeMatrixToScalar, 591
 - calculateFunction, 591
 - cloneExprNode, 592

- constructADTape, 592
- getTokenName, 591
- OSnLNodeMatrixToScalar, 591
- OSnLNodeMatrixToScalar, 591
- OSnLNodeMatrixTrace, 592
 - ~OSnLNodeMatrixTrace, 594
 - calculateFunction, 594
 - cloneExprNode, 595
 - constructADTape, 595
 - getTokenName, 594
 - OSnLNodeMatrixTrace, 594
 - OSnLNodeMatrixTrace, 594
- OSnLNodeMax, 595
 - ~OSnLNodeMax, 597
 - calculateFunction, 597
 - cloneExprNode, 598
 - constructADTape, 598
 - getTokenName, 597
 - OSnLNodeMax, 597
 - OSnLNodeMax, 597
- OSnLNodeMin, 598
 - ~OSnLNodeMin, 600
 - calculateFunction, 600
 - cloneExprNode, 601
 - constructADTape, 601
 - getTokenName, 600
 - OSnLNodeMin, 600
 - OSnLNodeMin, 600
- OSnLNodeMinus, 601
 - ~OSnLNodeMinus, 603
 - calculateFunction, 603
 - cloneExprNode, 604
 - constructADTape, 604
 - getTokenName, 603
 - OSnLNodeMinus, 603
 - OSnLNodeMinus, 603
- OSnLNodeNegate, 604
 - ~OSnLNodeNegate, 606
 - calculateFunction, 606
 - cloneExprNode, 607
 - constructADTape, 607
 - getTokenName, 606
 - OSnLNodeNegate, 606
 - OSnLNodeNegate, 606
- OSnLNodeNumber, 607
 - ~OSnLNodeNumber, 610
 - calculateFunction, 610
 - cloneExprNode, 610
 - constructADTape, 611
 - getNonlinearExpressionInXML, 610
 - getTokenName, 610
 - getTokenNumber, 610
 - id, 611
 - isEqual, 611
 - OSnLNodeNumber, 610
 - OSnLNodeNumber, 610
 - type, 611
 - value, 611
- OSnLNodePI, 611
 - ~OSnLNodePI, 613
 - calculateFunction, 614
 - cloneExprNode, 614
 - constructADTape, 614
 - getNonlinearExpressionInXML, 614
 - getTokenName, 613
 - getTokenNumber, 613
 - OSnLNodePI, 613
 - OSnLNodePI, 613
- OSnLNodePlus, 615
 - ~OSnLNodePlus, 616
 - calculateFunction, 617
 - cloneExprNode, 617
 - constructADTape, 617
 - getTokenName, 617
 - OSnLNodePlus, 616
 - OSnLNodePlus, 616
- OSnLNodePower, 617
 - ~OSnLNodePower, 619
 - calculateFunction, 619
 - cloneExprNode, 620
 - constructADTape, 620
 - getTokenName, 619
 - OSnLNodePower, 619
 - OSnLNodePower, 619
- OSnLNodeProduct, 620
 - ~OSnLNodeProduct, 622
 - calculateFunction, 622
 - cloneExprNode, 623
 - constructADTape, 623
 - getTokenName, 622
 - OSnLNodeProduct, 622
 - OSnLNodeProduct, 622
- OSnLNodeSin, 623
 - ~OSnLNodeSin, 625
 - calculateFunction, 625
 - cloneExprNode, 626
 - constructADTape, 626
 - getTokenName, 625
 - OSnLNodeSin, 625
 - OSnLNodeSin, 625
- OSnLNodeSqrt, 626
 - ~OSnLNodeSqrt, 628
 - calculateFunction, 628
 - cloneExprNode, 629
 - constructADTape, 629
 - getTokenName, 628
 - OSnLNodeSqrt, 628
 - OSnLNodeSqrt, 628

- OSnLNodeSquare, 629
 - ~OSnLNodeSquare, 631
 - calculateFunction, 631
 - cloneExprNode, 632
 - constructADTape, 632
 - getTokenName, 631
 - OSnLNodeSquare, 631
 - OSnLNodeSquare, 631
- OSnLNodeSum, 632
 - ~OSnLNodeSum, 634
 - calculateFunction, 634
 - cloneExprNode, 635
 - constructADTape, 635
 - getTokenName, 634
 - OSnLNodeSum, 634
 - OSnLNodeSum, 634
- OSnLNodeTimes, 635
 - ~OSnLNodeTimes, 637
 - calculateFunction, 637
 - cloneExprNode, 638
 - constructADTape, 638
 - getTokenName, 637
 - OSnLNodeTimes, 637
 - OSnLNodeTimes, 637
- OSnLNodeVariable, 638
 - ~OSnLNodeVariable, 640
 - calculateFunction, 641
 - cloneExprNode, 642
 - coef, 642
 - constructADTape, 642
 - getNonlinearExpressionInXML, 641
 - getTokenName, 641
 - getTokenNumber, 641
 - getVariableIndexMap, 641
 - idx, 642
 - isEqual, 642
 - OSnLNodeVariable, 640
 - OSnLNodeVariable, 640
- OSnLParserData, 642
 - ~OSnLParserData, 646
 - allDiffVec, 650
 - categoryAttribute, 646
 - categoryAttributePresent, 646
 - conTypeAttribute, 647
 - conTypeAttributePresent, 646
 - descriptionAttribute, 647
 - descriptionAttributePresent, 647
 - enumTypeAttribute, 647
 - enumTypeAttributePresent, 647
 - errorText, 651
 - iOption, 648
 - iOther, 648
 - idxAttribute, 648
 - idxAttributePresent, 648
 - ignoreDataAfterErrors, 652
 - includeDiagonalAttribute, 651
 - includeDiagonalAttributePresent, 651
 - kounter, 648
 - lbValueAttribute, 647
 - lbValueAttributePresent, 647
 - matrixProductVec, 651
 - matrixSumVec, 651
 - matrixidxattON, 651
 - matrixreftypeattON, 651
 - maxVec, 650
 - minVec, 651
 - nameAttribute, 647
 - nameAttributePresent, 647
 - nIMNodeMatrixCon, 649
 - nIMNodeMatrixObj, 649
 - nIMNodeMatrixRef, 649
 - nIMNodeMatrixVar, 649
 - nINodeNumberPoint, 649
 - nINodePoint, 648
 - nINodeVariablePoint, 649
 - nINodeVec, 650
 - nInodenum, 649
 - numberOf, 648
 - numberidattON, 650
 - numbertypeattON, 649
 - numbervalueattON, 650
 - OSnLParserData, 646
 - objTypeAttribute, 646
 - objTypeAttributePresent, 646
 - OSnLParserData, 646
 - parser_errors, 651
 - productVec, 650
 - shapeAttributePresent, 648
 - solverAttribute, 648
 - solverAttributePresent, 647
 - sumVec, 650
 - suppressFurtherErrorMessages, 652
 - templnt, 648
 - tempStr, 648
 - tempVal, 648
 - tmpnlcount, 649
 - typeAttribute, 646
 - typeAttributePresent, 646
 - ubValueAttribute, 647
 - ubValueAttributePresent, 647
 - unitAttribute, 648
 - unitAttributePresent, 648
 - valueAttribute, 647
 - valueAttributePresent, 647
 - varTypeAttribute, 646
 - varTypeAttributePresent, 646
 - variablecoefattON, 650
 - variableidxattON, 650

- OSnLParserData.h
 - osnl_empty_vectors, 1088
- OSnI2OS, 499
 - ~OSnI2OS, 502
 - createOSObjects, 503
 - getASL, 502
 - jobID, 504
 - numkount, 504
 - OSnI2OS, 502
 - op_type, 504
 - operand, 504
 - osinstance, 504
 - OSnI2OS, 502
 - osol, 504
 - osolreader, 504
 - osoption, 504
 - readNI, 502
 - setASL, 502
 - setIBVar, 503
 - setJobID, 502
 - setOsol, 502
 - setVar, 503
 - walkTree, 503
- OSoLParserData, 652
 - ~OSoLParserData, 657
 - categoryAttribute, 667
 - categoryAttributePresent, 667
 - conTypeAttribute, 668
 - conTypeAttributePresent, 668
 - constraintsPresent, 663
 - contactPresent, 659
 - currentSOS, 665
 - dependenciesPresent, 660
 - descriptionAttribute, 669
 - descriptionAttributePresent, 669
 - directoriesToDeletePresent, 662
 - directoriesToMakePresent, 661
 - enumTypeAttribute, 668
 - enumTypeAttributePresent, 668
 - errorText, 671
 - filesToDeletePresent, 662
 - filesToMakePresent, 661
 - fromPaths, 669
 - groupWeight, 666
 - groupWeightAttributePresent, 664
 - iOption, 670
 - iOther, 670
 - idxArray, 670
 - idxAttribute, 669
 - idxAttributePresent, 663
 - ignoreDataAfterErrors, 671
 - initialBasisStatusPresent, 664
 - initialConstraintValuesPresent, 665
 - initialDualVariableValuesPresent, 665
 - initialObjectiveBoundsPresent, 665
 - initialObjectiveValuesPresent, 665
 - initialVariableValuesPresent, 664
 - initialVariableValuesStringPresent, 664
 - inputDirectoriesToMovePresent, 661
 - inputFilesToMovePresent, 661
 - instanceLocationPresent, 658
 - instanceLocationTypeattON, 658
 - instanceNamePresent, 658
 - itemContent, 667
 - itemList, 670
 - jobDependencies, 669
 - jobIDPresent, 658
 - kounter, 670
 - lbDualValue, 666
 - lbValArray, 670
 - lbValAttributePresent, 663
 - lbValueAttribute, 668
 - lbValueAttributePresent, 668
 - lbValueString, 670
 - licensePresent, 658
 - makeCopy, 670
 - maxTimePresent, 660
 - maxTimeUnit, 660
 - maxTimeUnitPresent, 660
 - maxTimeValue, 660
 - minCPUNumberPresent, 659
 - minCPUSpeedPresent, 659
 - minCPUSpeedUnitPresent, 659
 - minDiskSpacePresent, 659
 - minDiskSpaceUnitPresent, 659
 - minMemoryPresent, 659
 - minMemoryUnitPresent, 659
 - namArray, 670
 - nameAttribute, 668
 - nameAttributePresent, 668
 - numberOf, 670
 - numberOfBasVar, 664
 - numberOfCon, 665
 - numberOfConAttributePresent, 664
 - numberOfConstraints, 663
 - numberOfConstraintsPresent, 663
 - numberOfDependencies, 660
 - numberOfDirectoriesToDelete, 662
 - numberOfDirectoriesToMake, 661
 - numberOfDuals, 666
 - numberOfEnumerations, 666
 - numberOfEnumerationsAttributePresent, 664
 - numberOfFilesToDelete, 662
 - numberOfFilesToMake, 661
 - numberOfInputDirectoriesToMove, 661
 - numberOfInputFilesToMove, 662
 - numberOfIntWt, 664
 - numberOfItems, 667

numberOfItemsPresent, 667
numberOfObj, 665
numberOfObjAttributePresent, 664
numberOfObjBounds, 665
numberOfObjValues, 665
numberOfObjectives, 663
numberOfObjectivesPresent, 663
numberOfOtherConstraintOptions, 665
numberOfOtherGeneralOptions, 659
numberOfOtherJobOptions, 662
numberOfOtherObjectiveOptions, 665
numberOfOtherServiceOptions, 660
numberOfOtherSystemOptions, 659
numberOfOtherVariableOptions, 664
numberOfOutputDirectoriesToMove, 662
numberOfOutputFilesToMove, 662
numberOfPathPairs, 662
numberOfProcessesToKill, 662
numberOfRequiredDirectories, 660
numberOfRequiredFiles, 661
numberOfSOS, 665
numberOfSOSVar, 665
numberOfSolverOptions, 666
numberOfVar, 664
numberOfVarAttributePresent, 664
numberOfVarStr, 664
numberOfVariables, 663
numberOfVariablesPresent, 663
OSoLParserData, 657
objTypeAttribute, 668
objTypeAttributePresent, 668
objectivesPresent, 663
osolgeneralPresent, 658
osoljobPresent, 658
osoloptimizationPresent, 658
OSoLParserData, 657
osolservicePresent, 658
osolsystemPresent, 658
otherGeneralOptionsPresent, 659
otherJobOptionsPresent, 662
otherOptionCategoryPresent, 666
otherOptionDescriptionPresent, 666
otherOptionNamePresent, 666
otherOptionNumberPresent, 666
otherOptionSolverPresent, 666
otherOptionType, 666
otherOptionTypePresent, 666
otherOptionValuePresent, 666
otherServiceOptionsPresent, 660
otherSystemOptionsPresent, 659
outputDirectoriesToMovePresent, 662
outputFilesToMovePresent, 662
parser_errors, 671
passwordPresent, 659
pathPairFrom, 661
pathPairFromPresent, 661
pathPairMakeCopy, 661
pathPairMakeCopyPresent, 661
pathPairTo, 661
pathPairToPresent, 661
paths, 669
processesToKill, 669
processesToKillPresent, 662
requestedStartTime, 660
requestedStartTimePresent, 660
requiredDirectoriesPresent, 660
requiredFilesPresent, 660
scanner, 671
serviceNamePresent, 658
serviceTypePresent, 660
serviceURIPresent, 658
solverAttribute, 669
solverAttributePresent, 669
solverOptionCategoryPresent, 667
solverOptionDescriptionPresent, 667
solverOptionNamePresent, 667
solverOptionSolverPresent, 667
solverOptionTypePresent, 667
solverOptionValuePresent, 667
solverOptionsPresent, 663
solverToInvokePresent, 658
sosIdx, 665
sosIdxAttributePresent, 664
statusDescription, 671
statusType, 671
suppressFurtherErrorMessages, 671
templnt, 670
tempStr, 671
tempVal, 671
toPaths, 669
transportTypeattON, 659
typeAttribute, 667
typeAttributePresent, 667
ubDualValue, 666
ubValArray, 670
ubValAttributePresent, 663
ubValueAttribute, 669
ubValueAttributePresent, 668
ubValueString, 670
unitAttribute, 669
unitAttributePresent, 669
usernamePresent, 658
valArray, 670
valAttributePresent, 663
valueAttribute, 668
valueAttributePresent, 668
valueString, 669
varTypeAttribute, 668

- varTypeAttributePresent, 667
- variablesPresent, 663
- OSoLReader, 671
 - ~OSoLReader, 672
 - OSoLReader, 672
 - OSoLReader, 672
 - readOSoL, 672
- OSoLWriter, 673
 - ~OSoLWriter, 674
 - m_bWhiteSpace, 674
 - m_bWriteBase64, 674
 - m_sB64encoded, 674
 - OSoLWriter, 674
 - OSoLWriter, 674
 - writeOSoL, 674
- OSosl2ampl, 720
 - ~OSosl2ampl, 721
 - OSosl2ampl, 721
 - OSosl2ampl, 721
 - writeSolFile, 721
- OSrL2Gams, 811
 - ~OSrL2Gams, 812
 - OSrL2Gams, 812
 - OSrL2Gams, 812
 - writeSolution, 812
- OSrLParserData, 812
 - ~OSrLParserData, 817
 - actualStartTimePresent, 825
 - categoryAttribute, 822
 - categoryAttributePresent, 821
 - conTypeAttribute, 823
 - conTypeAttributePresent, 822
 - descriptionAttribute, 822
 - descriptionAttributePresent, 821
 - dualValPair, 826
 - dualVals, 826
 - enumTypeAttribute, 823
 - enumTypeAttributePresent, 822
 - errorText, 826
 - generalInstanceNamePresent, 824
 - generalJobIDPresent, 824
 - generalMessagePresent, 823
 - generalServiceNamePresent, 823
 - generalServiceURIPresent, 823
 - generalSolverInvokedPresent, 824
 - generalStatusPresent, 823
 - generalTimeStampPresent, 824
 - iOther, 819
 - idx, 819
 - idxAttributePresent, 821
 - ignoreDataAfterErrors, 827
 - incr, 821
 - incrPresent, 821
 - itemContent, 818
 - ivar, 819
 - jobEndTimePresent, 825
 - jobStatusPresent, 825
 - jobSubmitTimePresent, 825
 - jobTimingInformationPresent, 825
 - jobUsedCPUNumberPresent, 825
 - jobUsedCPUSpeedPresent, 825
 - jobUsedDiskSpacePresent, 825
 - jobUsedMemoryPresent, 825
 - kounter, 819
 - mult, 820
 - multPresent, 821
 - nConPresent, 823
 - nObjPresent, 823
 - nVarPresent, 823
 - name, 820
 - nameAttribute, 822
 - nameAttributePresent, 821
 - numberAttributePresent, 821
 - numberOf, 819
 - numberOfCon, 819
 - numberOfConAttributePresent, 821
 - numberOfConIdxAttributePresent, 822
 - numberOfConstraints, 818
 - numberOfEnumerations, 819
 - numberOfEnumerationsAttributePresent, 822
 - numberOfIdx, 819
 - numberOfItems, 826
 - numberOfItemsPresent, 826
 - numberOfObj, 819
 - numberOfObjAttributePresent, 821
 - numberOfObjIdxAttributePresent, 821
 - numberOfObjectives, 818
 - numberOfOtherConstraintResults, 820
 - numberOfOtherObjectiveResults, 820
 - numberOfOtherVariableResults, 820
 - numberOfSolutions, 818
 - numberOfTimes, 818
 - numberOfVar, 819
 - numberOfVarAttributePresent, 821
 - numberOfVarIdx, 819
 - numberOfVarIdxAttributePresent, 821
 - numberOfVariables, 818
 - OSrLParserData, 817
 - objTypeAttribute, 823
 - objTypeAttributePresent, 822
 - objValPair, 826
 - objVals, 826
 - OSrLParserData, 817
 - otherGeneralResultsPresent, 824
 - otherJobResultsPresent, 825
 - otherServiceResultsPresent, 825
 - otherSystemResultsPresent, 824
 - otherVarStruct, 826

- otherVarVec, [826](#)
- outStr, [820](#)
- parser_errors, [826](#)
- primalValPair, [826](#)
- primalVals, [826](#)
- scanner, [818](#)
- scheduledStartTimePresent, [825](#)
- serviceCurrentJobCountPresent, [824](#)
- serviceCurrentStatePresent, [824](#)
- serviceTotalJobsSoFarPresent, [824](#)
- serviceUtilizationPresent, [825](#)
- solutionIdx, [820](#)
- statusDescription, [817](#)
- statusType, [817](#)
- suppressFurtherErrorMessages, [827](#)
- systemAvailableCPUNumberPresent, [824](#)
- systemAvailableCPUSpeedPresent, [824](#)
- systemAvailableDiskSpacePresent, [824](#)
- systemAvailableMemoryPresent, [824](#)
- systemInformationPresent, [824](#)
- templnt, [820](#)
- tempStr, [820](#)
- tempVal, [820](#)
- timeCategory, [817](#)
- timeDescription, [818](#)
- timeServiceStartedPresent, [825](#)
- timeType, [817](#)
- timeUnit, [817](#)
- timeValue, [817](#)
- tmpOtherDescription, [818](#)
- tmpOtherName, [818](#)
- tmpOtherValue, [818](#)
- typeAttribute, [823](#)
- typeAttributePresent, [822](#)
- unitAttribute, [823](#)
- unitAttributePresent, [822](#)
- valueAttribute, [823](#)
- valueAttributePresent, [822](#)
- varTypeAttribute, [823](#)
- varTypeAttributePresent, [822](#)
- weightedObjAttributePresent, [822](#)
- OSrLReader, [827](#)
 - ~OSrLReader, [828](#)
 - OSrLReader, [828](#)
 - OSrLReader, [828](#)
 - readOSrL, [828](#)
- OSrLWriter, [828](#)
 - ~OSrLWriter, [830](#)
 - m_bWhiteSpace, [830](#)
 - m_bWriteBase64, [830](#)
 - m_sB64encoded, [830](#)
 - OSrLWriter, [830](#)
 - OSrLWriter, [830](#)
 - writeOSrL, [830](#)
- OTHEREND
 - OSParseosol.tab.hpp, [1183](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OTHEROPTIONSSEND
 - OSParseosol.tab.hpp, [1183](#)
- OTHEROPTIONSSTART
 - OSParseosol.tab.hpp, [1182](#)
- OTHERRESULTSEND
 - OSParseosrl.tab.hpp, [1259](#)
- OTHERRESULTSSTART
 - OSParseosrl.tab.hpp, [1259](#)
- OTHERSOLVEROUTPUTEND
 - OSParseosrl.tab.hpp, [1260](#)
- OTHERSTART
 - OSParseosol.tab.hpp, [1183](#)
 - OSParseosrl.tab.hpp, [1259](#)
- OUTPUTFILESTOMOVEEND
 - OSParseosol.tab.hpp, [1186](#)
- obj
 - InitObjectiveBounds, [192](#)
 - InitObjectiveValues, [196](#)
 - Objectives, [359](#)
 - ObjectiveValues, [365](#)
 - OSMatlab, [487](#)
 - OtherObjectiveOption, [858](#)
 - OtherObjectiveResult, [861](#)
 - TimeDomainStageObjectives, [996](#)
- obj_body
 - CouenneSolver, [100](#)
- ObjCoef, [350](#)
 - ~ObjCoef, [351](#)
 - idx, [351](#)
 - IsEqual, [351](#)
 - ObjCoef, [351](#)
 - ObjCoef, [351](#)
 - value, [351](#)
- ObjReferenceMatrixElements, [365](#)
 - ~ObjReferenceMatrixElements, [367](#)
 - alignsOnBlockBoundary, [367](#)
 - cloneMatrixNode, [368](#)
 - deepCopyFrom, [368](#)
 - getMatrixNodeInXML, [367](#)
 - getMatrixType, [367](#)
 - getNodeName, [367](#)
 - getNodeType, [367](#)
 - IsEqual, [368](#)
 - ObjReferenceMatrixElements, [367](#)
 - ObjReferenceMatrixElements, [367](#)
 - setRandom, [368](#)
 - values, [369](#)
- objReferenceMatrixIdx
 - MatrixObj, [308](#)
 - OSILParserData, [430](#)
- objReferenceMatrixIdxPresent

- OSILParserData, 429
- ObjReferenceMatrixValues, 369
 - ~ObjReferenceMatrixValues, 370
 - deepCopyFrom, 370
 - el, 371
 - IsEqual, 370
 - ObjReferenceMatrixValues, 370
 - ObjReferenceMatrixValues, 370
 - setRandom, 370
- objType
 - OSMatlab, 487
 - OtherObjectiveOption, 858
 - OtherObjectiveResult, 861
- objTypeAttribute
 - OSnLParserData, 646
 - OSoLParserData, 668
 - OSrLParserData, 823
- objTypeAttributePresent
 - OSnLParserData, 646
 - OSoLParserData, 668
 - OSrLParserData, 822
- objValPair
 - OSrLParserData, 826
- objVals
 - OSrLParserData, 826
- ObjValue, 371
 - ~ObjValue, 372
 - idx, 373
 - IsEqual, 372
 - name, 373
 - ObjValue, 372
 - ObjValue, 372
 - setRandom, 372
 - value, 373
- Objective, 351
 - ~Objective, 353
 - coef, 353
 - constant, 353
 - IsEqual, 353
 - maxOrMin, 353
 - name, 353
 - numberOfObjCoef, 353
 - Objective, 353
 - weight, 353
- ObjectiveOption, 354
 - ~ObjectiveOption, 355
 - addOther, 356
 - deepCopyFrom, 356
 - initialBasisStatus, 357
 - initialObjectiveBounds, 357
 - initialObjectiveValues, 356
 - IsEqual, 356
 - numberOfOtherObjectiveOptions, 356
 - ObjectiveOption, 355
- ObjectiveOption, 355
 - other, 357
 - setOther, 356
 - setRandom, 356
- ObjectiveSolution, 359
 - ~ObjectiveSolution, 361
 - basisStatus, 362
 - IsEqual, 361
 - numberOfOtherObjectiveResults, 362
 - ObjectiveSolution, 361
 - ObjectiveSolution, 361
 - other, 362
 - setRandom, 361
 - values, 362
- ObjectiveValues, 362
 - ~ObjectiveValues, 364
 - IsEqual, 364
 - numberOfObj, 364
 - obj, 365
 - ObjectiveValues, 364
 - ObjectiveValues, 364
 - setRandom, 364
- Objectives, 357
 - ~Objectives, 359
 - IsEqual, 359
 - numberOfObjectives, 359
 - obj, 359
 - Objectives, 359
- objectives
 - InstanceData, 215
 - OptimizationOption, 376
 - OptimizationSolution, 382
 - TimeDomainStage, 991
- objectivesPresent
 - OSoLParserData, 663
- op_type
 - OSnl2OS, 504
- Open
 - OSOutputChannel, 727
- operand
 - OSnl2OS, 504
- operator*
 - OSSmartPtr, 835
- operator->
 - OSSmartPtr, 835
- operator=
 - OSSmartPtr, 835
- operator==
 - OSSmartPtr, 835
 - OSSmartPtr.hpp, 1352
- optimization
 - OSOption, 713
 - OSResult, 810
- OptimizationOption, 373

- ~OptimizationOption, 375
- constraints, 376
- deepCopyFrom, 376
- isEqual, 375
- numberOfConstraints, 376
- numberOfObjectives, 376
- numberOfVariables, 376
- objectives, 376
- OptimizationOption, 375
- OptimizationOption, 375
- setRandom, 375
- solverOptions, 376
- variables, 376
- OptimizationResult, 377
 - ~OptimizationResult, 378
 - isEqual, 378
 - numberOfConstraints, 379
 - numberOfObjectives, 379
 - numberOfSolutions, 379
 - numberOfVariables, 379
 - OptimizationResult, 378
 - OptimizationResult, 378
 - otherSolverOutput, 379
 - setRandom, 378
 - solution, 379
- OptimizationSolution, 379
 - ~OptimizationSolution, 381
 - constraints, 382
 - isEqual, 381
 - message, 382
 - objectives, 382
 - OptimizationSolution, 381
 - OptimizationSolution, 381
 - otherSolutionResults, 382
 - setRandom, 381
 - status, 382
 - targetObjectiveldx, 382
 - targetObjectiveName, 382
 - variables, 382
 - weightedObjectives, 382
- OptimizationSolutionStatus, 383
 - ~OptimizationSolutionStatus, 384
 - description, 385
 - isEqual, 384
 - numberOfSubstatuses, 385
 - OptimizationSolutionStatus, 384
 - OptimizationSolutionStatus, 384
 - setRandom, 384
 - substatus, 385
 - type, 385
- OptimizationSolutionSubstatus, 385
 - ~OptimizationSolutionSubstatus, 387
 - description, 387
 - isEqual, 387
 - OptimizationSolutionSubstatus, 387
 - OptimizationSolutionSubstatus, 387
 - setRandom, 387
 - type, 387
- optimize
 - LindoSolver, 257
- optionHeader
 - OSOption, 713
- orderConeldx
 - MatrixObj, 309
 - OSILParserData, 429
- orderConeldxPresent
 - OSILParserData, 429
- OrthantCone, 388
 - ~OrthantCone, 389
 - deepCopyFrom, 390
 - getConelInXML, 389
 - getConeName, 389
 - isEqual, 390
 - lb, 390
 - OrthantCone, 389
 - OrthantCone, 389
 - setRandom, 390
 - ub, 390
- os_dtoa
 - OSdtoa.h, 1321
- os_freeditoa
 - OSdtoa.h, 1321
- os_strtod
 - OSdtoa.h, 1321
- os_strtod_wrap
 - OSMathUtil.h, 1325
- osExpressionTree
 - NI, 342
- osOptionsStruc, 714
 - browser, 719
 - configFile, 717
 - dat, 718
 - datFile, 718
 - filePrintLevel, 719
 - gamsControlFile, 719
 - insList, 717
 - insListFile, 717
 - invokeHelp, 719
 - jobID, 719
 - logFile, 719
 - mps, 718
 - mpsFile, 718
 - nl, 718
 - nlFile, 718
 - osOptionsStruc, 716
 - osil, 717
 - osilFile, 717
 - osol, 717

- osolFile, [717](#)
- osOptionsStruc, [716](#)
- osplInput, [718](#)
- osplInputFile, [718](#)
- osplOutput, [718](#)
- osplOutputFile, [718](#)
- osrl, [717](#)
- osrlFile, [717](#)
- printLevel, [719](#)
- printModel, [720](#)
- printRowNumberAsString, [720](#)
- quit, [720](#)
- resetOptions, [716](#)
- serviceLocation, [717](#)
- serviceMethod, [717](#)
- solverName, [719](#)
- writeVersion, [719](#)
- osgl_empty_vectors
 - OSgLPParserData.h, [1085](#)
- osglCoef
 - OSgLPParserData, [414](#)
- osglCoefPresent
 - OSgLPParserData, [414](#)
- osglConstantPresent
 - OSgLPParserData, [414](#)
- osglCounter
 - OSgLPParserData, [409](#)
- osglDbIArray
 - OSgLPParserData, [409](#)
- osglIncr
 - OSgLPParserData, [409](#)
- osglIncrPresent
 - OSgLPParserData, [409](#)
- osglIntArray
 - OSgLPParserData, [409](#)
- osglMult
 - OSgLPParserData, [409](#)
- osglMultPresent
 - OSgLPParserData, [409](#)
- osglNonzeroCounter
 - OSgLPParserData, [415](#)
- osglNumberOfEI
 - OSgLPParserData, [409](#)
- osglNumberOfEIPresent
 - OSgLPParserData, [409](#)
- osglNumberOfNonzeros
 - OSgLPParserData, [415](#)
- osglSize
 - OSgLPParserData, [409](#)
- osglTempint
 - OSgLPParserData, [409](#)
- osglValArray
 - OSgLPParserData, [409](#)
- osiSolver
 - CoinSolver, [57](#)
- osil
 - DefaultSolver, [113](#)
 - OSCommandLine, [396](#)
 - OSMatlab, [488](#)
 - osOptionsStruc, [717](#)
- osilFile
 - OSCommandLine, [396](#)
 - osOptionsStruc, [717](#)
- osilOutputFile
 - OSCommandLine, [396](#)
- osillineno
 - OSILParserData, [423](#)
- osinstance
 - BonminProblem, [46](#)
 - DefaultSolver, [113](#)
 - IpoptProblem, [233](#)
 - KnitroProblem, [250](#)
 - OSCommandLine, [395](#)
 - OSgams2osil, [405](#)
 - OSMatlab, [488](#)
 - OSmps2OS, [497](#)
 - OSmps2osil, [499](#)
 - OSnl2OS, [504](#)
- osnl_empty_vectors
 - OSnLPParserData.h, [1088](#)
- osol
 - DefaultSolver, [113](#)
 - OSCommandLine, [396](#)
 - OSmps2OS, [497](#)
 - OSnl2OS, [504](#)
 - osOptionsStruc, [717](#)
- osolFile
 - OSCommandLine, [396](#)
 - osOptionsStruc, [717](#)
- osolOutputFile
 - OSCommandLine, [397](#)
- osolgeneralPresent
 - OSoLPParserData, [658](#)
- osoljobPresent
 - OSoLPParserData, [658](#)
- osoloptimizationPresent
 - OSoLPParserData, [658](#)
- osolreader
 - OSmps2OS, [497](#)
 - OSnl2OS, [504](#)
- osolservicePresent
 - OSoLPParserData, [658](#)
- osolsystemPresent
 - OSoLPParserData, [658](#)
- osoption
 - BonminProblem, [46](#)
 - DefaultSolver, [113](#)
 - IpoptProblem, [233](#)

- OSCommandLine, 395
- OSmps2OS, 497
- OSnl2OS, 504
- osoutput
 - OSOutput.h, 1327
- osplInput
 - OSCommandLine, 397
 - osOptionsStruc, 718
- osplInputFile
 - OSCommandLine, 397
 - osOptionsStruc, 718
- osplOutput
 - osOptionsStruc, 718
- osplOutputFile
 - OSCommandLine, 397
 - osOptionsStruc, 718
- osresult
 - DefaultSolver, 114
 - IpoptProblem, 233
 - KnitroProblem, 250
- osrl
 - DefaultSolver, 113
 - osOptionsStruc, 717
- osrlFile
 - OSCommandLine, 397
 - osOptionsStruc, 717
- other
 - ConstraintOption, 86
 - ConstraintSolution, 90
 - ObjectiveOption, 357
 - ObjectiveSolution, 362
 - OtherOptions, 875
 - OtherResults, 881
 - VariableOption, 1013
 - VariableSolution, 1018
- OtherConOption, 840
 - ~OtherConOption, 841
 - deepCopyFrom, 842
 - idx, 842
 - IsEqual, 842
 - lbValue, 842
 - name, 842
 - OtherConOption, 841
 - OtherConOption, 841
 - setRandom, 842
 - ubValue, 842
 - value, 842
- OtherConResult, 843
 - ~OtherConResult, 844
 - idx, 845
 - IsEqual, 844
 - name, 845
 - OtherConResult, 844
 - OtherConResult, 844
- setRandom, 844
- value, 845
- OtherConstraintOption, 845
 - ~OtherConstraintOption, 848
 - addCon, 848
 - category, 849
 - con, 849
 - conType, 850
 - deepCopyFrom, 848
 - description, 849
 - enumType, 850
 - enumeration, 850
 - IsEqual, 848
 - name, 849
 - numberOfCon, 849
 - numberOfEnumerations, 849
 - OtherConstraintOption, 848
 - OtherConstraintOption, 848
 - setCon, 848
 - setRandom, 848
 - solver, 849
 - type, 849
 - value, 849
- OtherConstraintResult, 850
 - ~OtherConstraintResult, 852
 - con, 853
 - conType, 853
 - description, 853
 - enumType, 854
 - enumeration, 853
 - IsEqual, 852
 - name, 853
 - numberOfCon, 853
 - numberOfEnumerations, 853
 - OtherConstraintResult, 852
 - OtherConstraintResult, 852
 - setRandom, 852
 - type, 853
 - value, 853
- otherGeneralOptionsPresent
 - OSoLParserData, 659
- otherGeneralResultsPresent
 - OSrLParserData, 824
- otherIndexes
 - Cone, 64
 - DualCone, 121
 - IntersectionCone, 226
 - PolarCone, 912
 - PolyhedralCone, 916
 - ProductCone, 923
 - QuadraticCone, 928
 - RotatedQuadraticCone, 935
 - SemidefiniteCone, 947
- otherJobOptionsPresent

- OSoLParserData, 662
- otherJobResultsPresent
 - OSrLParserData, 825
- OtherObjOption, 862
 - ~OtherObjOption, 863
 - deepCopyFrom, 864
 - idx, 864
 - isEqual, 864
 - lbValue, 864
 - name, 864
 - OtherObjOption, 863
 - OtherObjOption, 863
 - setRandom, 864
 - ubValue, 864
 - value, 864
- OtherObjResult, 865
 - ~OtherObjResult, 866
 - idx, 867
 - isEqual, 866
 - name, 867
 - OtherObjResult, 866
 - OtherObjResult, 866
 - setRandom, 866
 - value, 867
- OtherObjectiveOption, 854
 - ~OtherObjectiveOption, 856
 - addObj, 857
 - category, 858
 - deepCopyFrom, 856
 - description, 858
 - enumType, 858
 - enumeration, 858
 - isEqual, 856
 - name, 857
 - numberOfEnumerations, 857
 - numberOfObj, 857
 - obj, 858
 - objType, 858
 - OtherObjectiveOption, 856
 - OtherObjectiveOption, 856
 - setObj, 857
 - setRandom, 856
 - solver, 857
 - type, 858
 - value, 857
- OtherObjectiveResult, 858
 - ~OtherObjectiveResult, 860
 - description, 861
 - enumType, 862
 - enumeration, 861
 - isEqual, 860
 - name, 861
 - numberOfEnumerations, 861
 - numberOfObj, 861
 - obj, 861
 - objType, 861
 - OtherObjectiveResult, 860
 - OtherObjectiveResult, 860
 - setRandom, 860
 - type, 861
 - value, 861
- OtherOption, 867
 - ~OtherOption, 868
 - deepCopyFrom, 869
 - description, 869
 - isEqual, 869
 - name, 869
 - OtherOption, 868
 - OtherOption, 868
 - setRandom, 869
 - value, 869
- otherOptionCategoryPresent
 - OSoLParserData, 666
- otherOptionDescriptionPresent
 - OSoLParserData, 666
- OtherOptionEnumeration, 870
 - ~OtherOptionEnumeration, 871
 - deepCopyFrom, 872
 - description, 872
 - getDescription, 872
 - getValue, 872
 - isEqual, 871
 - OtherOptionEnumeration, 871
 - OtherOptionEnumeration, 871
 - setOtherOptionEnumeration, 872
 - setRandom, 871
 - value, 872
- otherOptionNamePresent
 - OSoLParserData, 666
- otherOptionNumberPresent
 - OSoLParserData, 666
- otherOptionSolverPresent
 - OSoLParserData, 666
- otherOptionType
 - OSoLParserData, 666
- otherOptionTypePresent
 - OSoLParserData, 666
- otherOptionValuePresent
 - OSoLParserData, 666
- OtherOptions, 873
 - ~OtherOptions, 874
 - addOther, 875
 - deepCopyFrom, 875
 - isEqual, 874
 - numberOfOtherOptions, 875
 - other, 875
 - OtherOptions, 874
 - OtherOptions, 874

- setOther, [875](#)
- setRandom, [874](#)
- otherOptions
 - GeneralOption, [153](#)
 - JobOption, [244](#)
 - ServiceOption, [950](#)
 - SystemOption, [984](#)
- OtherResult, [876](#)
 - ~OtherResult, [877](#)
 - description, [878](#)
 - IsEqual, [877](#)
 - name, [878](#)
 - OtherResult, [877](#)
 - OtherResult, [877](#)
 - setRandom, [877](#)
 - value, [878](#)
- OtherResults, [878](#)
 - ~OtherResults, [880](#)
 - IsEqual, [880](#)
 - numberOfOtherResults, [880](#)
 - other, [881](#)
 - OtherResults, [880](#)
 - OtherResults, [880](#)
 - setRandom, [880](#)
- otherResults
 - GeneralResult, [157](#)
 - JobResult, [248](#)
 - ServiceResult, [953](#)
 - SystemResult, [987](#)
- otherServiceOptionsPresent
 - OSoLParserData, [660](#)
- otherServiceResultsPresent
 - OSrLParserData, [825](#)
- OtherSolutionResult, [881](#)
 - ~OtherSolutionResult, [882](#)
 - category, [883](#)
 - description, [883](#)
 - IsEqual, [883](#)
 - item, [883](#)
 - name, [883](#)
 - numberOfItems, [883](#)
 - OtherSolutionResult, [882](#)
 - OtherSolutionResult, [882](#)
 - setRandom, [883](#)
 - value, [883](#)
- otherSolutionResult
 - OtherSolutionResults, [886](#)
- OtherSolutionResults, [884](#)
 - ~OtherSolutionResults, [885](#)
 - IsEqual, [885](#)
 - numberOfOtherSolutionResults, [886](#)
 - otherSolutionResult, [886](#)
 - OtherSolutionResults, [885](#)
 - OtherSolutionResults, [885](#)
- setRandom, [885](#)
- otherSolutionResults
 - OptimizationSolution, [382](#)
- OtherSolverOutput, [886](#)
 - ~OtherSolverOutput, [887](#)
 - IsEqual, [888](#)
 - numberOfSolverOutputs, [888](#)
 - OtherSolverOutput, [887](#)
 - OtherSolverOutput, [887](#)
 - setRandom, [888](#)
 - solverOutput, [888](#)
- otherSolverOutput
 - OptimizationResult, [379](#)
- otherSystemOptionsPresent
 - OSoLParserData, [659](#)
- otherSystemResultsPresent
 - OSrLParserData, [824](#)
- otherVarIndex
 - OtherVariableResultStruct, [897](#)
- OtherVarOption, [898](#)
 - ~OtherVarOption, [899](#)
 - deepCopyFrom, [900](#)
 - idx, [900](#)
 - IsEqual, [899](#)
 - lbValue, [900](#)
 - name, [900](#)
 - OtherVarOption, [899](#)
 - OtherVarOption, [899](#)
 - setRandom, [899](#)
 - ubValue, [900](#)
 - value, [900](#)
- OtherVarResult, [901](#)
 - ~OtherVarResult, [902](#)
 - idx, [902](#)
 - IsEqual, [902](#)
 - name, [903](#)
 - OtherVarResult, [902](#)
 - OtherVarResult, [902](#)
 - setRandom, [902](#)
 - value, [903](#)
- otherVarStruct
 - OSrLParserData, [826](#)
- otherVarText
 - OtherVariableResultStruct, [897](#)
- otherVarVec
 - OSrLParserData, [826](#)
- OtherVariableOption, [888](#)
 - ~OtherVariableOption, [890](#)
 - addVar, [891](#)
 - category, [892](#)
 - deepCopyFrom, [891](#)
 - description, [892](#)
 - enumType, [892](#)
 - enumeration, [892](#)

- IsEqual, 890
- name, 891
- numberOfEnumerations, 891
- numberOfVar, 891
- OtherVariableOption, 890
- OtherVariableOption, 890
- setRandom, 890
- setVar, 891
- solver, 892
- type, 892
- value, 892
- var, 892
- varType, 892
- OtherVariableResult, 893
 - ~OtherVariableResult, 894
 - description, 895
 - enumType, 896
 - enumeration, 896
 - IsEqual, 895
 - name, 895
 - numberOfEnumerations, 895
 - numberOfVar, 895
 - OtherVariableResult, 894
 - OtherVariableResult, 894
 - setRandom, 895
 - type, 895
 - value, 895
 - var, 895
 - varType, 895
- OtherVariableResultStruct, 896
 - description, 897
 - name, 897
 - numberOfVar, 897
 - otherVarIndex, 897
 - otherVarText, 897
 - value, 897
- outStr
 - OSrLParserData, 820
- outputDirectoriesToMove
 - JobOption, 244
- outputDirectoriesToMovePresent
 - OSoLParserData, 662
- outputFilesToMove
 - JobOption, 244
- outputFilesToMovePresent
 - OSoLParserData, 662
- PASSWORDEND
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1215
 - OSParseosrl.tab.hpp, 1288
- PASSWORDSTART
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1215
- OSParseosrl.tab.hpp, 1288
- PATHEND
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289
- PATHPAIREND
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289
- PATHPAIRSTART
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289
- PATHSTART
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289
- PATTERNELEMENTSEND
 - OSParseosil.tab.hpp, 1136, 1145, 1155
 - OSParseosol.tab.hpp, 1210, 1220, 1230
 - OSParseosrl.tab.hpp, 1283, 1293, 1303
- PATTERNELEMENTSSTART
 - OSParseosil.tab.hpp, 1136, 1145, 1155
 - OSParseosol.tab.hpp, 1210, 1220, 1230
 - OSParseosrl.tab.hpp, 1283, 1293, 1303
- PIEND
 - OSParseosil.tab.hpp, 1137, 1147, 1156
 - OSParseosol.tab.hpp, 1212, 1222, 1231
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- PISTART
 - OSParseosil.tab.hpp, 1137, 1147, 1156
 - OSParseosol.tab.hpp, 1212, 1222, 1231
 - OSParseosrl.tab.hpp, 1285, 1295, 1304
- PLUSEND
 - OSParseosil.tab.hpp, 1136, 1146, 1156
 - OSParseosol.tab.hpp, 1211, 1221, 1230
 - OSParseosrl.tab.hpp, 1284, 1294, 1303
- PLUSSTART
 - OSParseosil.tab.hpp, 1136, 1146, 1156
 - OSParseosol.tab.hpp, 1211, 1221, 1230
 - OSParseosrl.tab.hpp, 1284, 1294, 1303
- POLARCONEEND
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- POLARCONESTART
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- POLYHEDRALCONEEND
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- POLYHEDRALCONESTART

- OSParseosil.tab.hpp, [1131](#)
- OSParseosol.tab.hpp, [1206](#)
- OSParseosrl.tab.hpp, [1279](#)
- POWEREND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- POWERSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- PROCESSEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- PROCESSESTOKILEND
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- PROCESSESTOKILLSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1289](#)
- PROCESSSTART
 - OSParseosil.tab.hpp, [1142](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- PRODUCTCONEEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- PRODUCTCONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- PRODUCTEND
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- PRODUCTSTART
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- PASSWORDEND
 - OSParseosol.tab.hpp, [1182](#)
- PASSWORDSTART
 - OSParseosol.tab.hpp, [1182](#)
- PATHEND
 - OSParseosol.tab.hpp, [1184](#)
- PATHPAIREND
 - OSParseosol.tab.hpp, [1184](#)
- PATHPAIRSTART
 - OSParseosol.tab.hpp, [1184](#)
- PATHSTART
 - OSParseosol.tab.hpp, [1184](#)
- OSParseosol.tab.hpp, [1184](#)
- PATTERNELEMENTSEND
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- PATTERNELEMENTSSTART
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- PIEND
 - OSParseosil.tab.hpp, [1127](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1274](#)
- PISTART
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1274](#)
- PLUSEND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- PLUSSTART
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- POLARCONEEND
 - OSParseosil.tab.hpp, [1110](#)
- POLARCONESTART
 - OSParseosil.tab.hpp, [1110](#)
- POLYHEDRALCONEEND
 - OSParseosil.tab.hpp, [1109](#)
- POLYHEDRALCONESTART
 - OSParseosil.tab.hpp, [1109](#)
- POWEREND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- POWERSTART
 - OSParseosil.tab.hpp, [1123](#)
 - OSParseosol.tab.hpp, [1198](#)
 - OSParseosrl.tab.hpp, [1271](#)
- PROCESSEND
 - OSParseosol.tab.hpp, [1186](#)
- PROCESSESTOKILEND
 - OSParseosol.tab.hpp, [1186](#)
- PROCESSESTOKILLSTART
 - OSParseosol.tab.hpp, [1186](#)
- PROCESSSTART
 - OSParseosol.tab.hpp, [1186](#)
- PRODUCTCONEEND
 - OSParseosil.tab.hpp, [1109](#)
- PRODUCTCONESTART
 - OSParseosil.tab.hpp, [1109](#)
- PRODUCTEND

- OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- PRODUCTSTART
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- parseString
 - OSCommandLineReader, [401](#)
- parser_errors
 - OSgLParseData, [410](#)
 - OSiLParseData, [431](#)
 - OSnLParseData, [651](#)
 - OSoLParseData, [671](#)
 - OSrLParseData, [826](#)
- password
 - GeneralOption, [153](#)
- passwordPresent
 - OSoLParseData, [659](#)
- path
 - DirectoriesAndFiles, [117](#)
- PathPair, [903](#)
 - ~PathPair, [904](#)
 - deepCopyFrom, [905](#)
 - from, [905](#)
 - IsEqual, [905](#)
 - makeCopy, [905](#)
 - PathPair, [904](#)
 - PathPair, [904](#)
 - setRandom, [905](#)
 - to, [905](#)
- pathPair
 - PathPairs, [909](#)
- pathPairFrom
 - OSoLParseData, [661](#)
- pathPairFromPresent
 - OSoLParseData, [661](#)
- pathPairMakeCopy
 - OSoLParseData, [661](#)
- pathPairMakeCopyPresent
 - OSoLParseData, [661](#)
- pathPairTo
 - OSoLParseData, [661](#)
- pathPairToPresent
 - OSoLParseData, [661](#)
- PathPairs, [906](#)
 - ~PathPairs, [907](#)
 - addPathPair, [908](#)
 - deepCopyFrom, [908](#)
 - IsEqual, [907](#)
 - numberOfPathPairs, [909](#)
 - pathPair, [909](#)
 - PathPairs, [907](#)
 - PathPairs, [907](#)
- setPathPair, [908](#)
 - setRandom, [907](#)
- paths
 - OSoLParseData, [669](#)
- PolarCone, [909](#)
 - ~PolarCone, [911](#)
 - coneType, [912](#)
 - deepCopyFrom, [911](#)
 - getConeName, [911](#)
 - idx, [912](#)
 - IsEqual, [911](#)
 - numberOfColumns, [912](#)
 - numberOfOtherIndexes, [912](#)
 - numberOfRows, [912](#)
 - otherIndexes, [912](#)
 - PolarCone, [911](#)
 - PolarCone, [911](#)
 - referenceConeIdx, [912](#)
 - setRandom, [911](#)
- PolyhedralCone, [913](#)
 - ~PolyhedralCone, [914](#)
 - coneType, [916](#)
 - deepCopyFrom, [915](#)
 - getConeInXML, [915](#)
 - getConeName, [915](#)
 - idx, [916](#)
 - IsEqual, [915](#)
 - numberOfColumns, [916](#)
 - numberOfOtherIndexes, [916](#)
 - numberOfRows, [915](#)
 - otherIndexes, [916](#)
 - PolyhedralCone, [914](#)
 - PolyhedralCone, [914](#)
 - referenceMatrixIdx, [916](#)
 - setRandom, [915](#)
- postOrderMatrixNodeTraversal
 - MatrixNode, [305](#)
- postOrderOSnLNodeTraversal
 - ExprNode, [136](#)
 - OSnLMNode, [508](#)
 - OSnLNode, [558](#)
- preOrderMatrixNodeTraversal
 - MatrixNode, [304](#)
- preOrderOSnLNodeTraversal
 - ExprNode, [135](#)
 - OSnLMNode, [508](#)
 - OSnLNode, [558](#)
- primalValPair
 - OSrLParseData, [826](#)
- primalVals
 - OSResult, [811](#)
 - OSrLParseData, [826](#)
- printLevel
 - OSCommandLine, [398](#)

- osOptionsStruc, [719](#)
- printModel
 - OSCommandLine, [399](#)
 - OSInstance, [474](#)
 - osOptionsStruc, [720](#)
- printRowNumberAsString
 - OSCommandLine, [399](#)
 - osOptionsStruc, [720](#)
- printSolutionAtEndOfAlgorithm
 - BonminProblem, [46](#)
- process
 - Processes, [919](#)
- processBlocks
 - MatrixType, [321](#)
 - OSMatrix, [492](#)
- processConstraints
 - LindoSolver, [258](#)
- processNonlinearExpressions
 - LindoSolver, [258](#)
- processQuadraticTerms
 - LindoSolver, [258](#)
- processVariables
 - LindoSolver, [257](#)
- Processes, [916](#)
 - ~Processes, [918](#)
 - addProcess, [919](#)
 - deepCopyFrom, [918](#)
 - IsEqual, [918](#)
 - numberOfProcesses, [919](#)
 - process, [919](#)
 - Processes, [918](#)
 - setProcess, [919](#)
 - setRandom, [918](#)
- processesToKill
 - JobOption, [244](#)
 - OSoLParserData, [669](#)
- processesToKillPresent
 - OSoLParserData, [662](#)
- ProductCone, [919](#)
 - ~ProductCone, [921](#)
 - coneType, [923](#)
 - deepCopyFrom, [922](#)
 - factors, [923](#)
 - getConeInXML, [921](#)
 - getConeName, [921](#)
 - idx, [923](#)
 - IsEqual, [922](#)
 - numberOfColumns, [922](#)
 - numberOfOtherIndexes, [922](#)
 - numberOfRows, [922](#)
 - otherIndexes, [923](#)
 - ProductCone, [921](#)
 - ProductCone, [921](#)
 - setRandom, [922](#)

- productVec
 - OSnLParserData, [650](#)
- QTERMEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QTERMSTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QUADRATICCOEFFICIENTSEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QUADRATICCOEFFICIENTSSTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QUADRATICCONEEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QUADRATICCONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- QUOTE
 - OSParseosil.tab.hpp, [1130](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- qIndex1
 - OSMatlab, [488](#)
- qIndex2
 - OSMatlab, [488](#)
- qRows
 - OSMatlab, [487](#)
- QTERMEND
 - OSParseosil.tab.hpp, [1108](#)
- QTERMSTART
 - OSParseosil.tab.hpp, [1108](#)
- qTerm
 - QuadraticCoefficients, [924](#)
- QUADRATICCONEEND
 - OSParseosil.tab.hpp, [1109](#)
- QUADRATICCONESTART
 - OSParseosil.tab.hpp, [1109](#)
- QUOTE
 - OSParseosil.tab.hpp, [1106](#)
 - OSParseosol.tab.hpp, [1176](#)
 - OSParseosrl.tab.hpp, [1250](#)
- qVal
 - OSMatlab, [488](#)
- qtermcoefattON

- OSILParserData, [423](#)
- qtermcount
 - OSILParserData, [423](#)
- qtermidattON
 - OSILParserData, [423](#)
- qtermidxOneattON
 - OSILParserData, [423](#)
- qtermidxTwoattON
 - OSILParserData, [423](#)
- qtermidxattON
 - OSILParserData, [423](#)
- QuadraticCoefficients, [923](#)
 - ~QuadraticCoefficients, [924](#)
 - IsEqual, [924](#)
 - numberOfQuadraticTerms, [924](#)
 - qTerm, [924](#)
 - QuadraticCoefficients, [924](#)
 - QuadraticCoefficients, [924](#)
- quadraticCoefficients
 - InstanceData, [215](#)
- QuadraticCone, [925](#)
 - ~QuadraticCone, [926](#)
 - axisDirection, [928](#)
 - coneType, [928](#)
 - deepCopyFrom, [927](#)
 - distortionMatrixIdx, [928](#)
 - getConeInXML, [927](#)
 - getConeName, [927](#)
 - idx, [928](#)
 - IsEqual, [927](#)
 - normScaleFactor, [928](#)
 - numberOfColumns, [928](#)
 - numberOfOtherIndexes, [928](#)
 - numberOfRows, [928](#)
 - otherIndexes, [928](#)
 - QuadraticCone, [926](#)
 - QuadraticCone, [926](#)
 - setRandom, [927](#)
- QuadraticTerm, [929](#)
 - ~QuadraticTerm, [929](#)
 - coef, [930](#)
 - idx, [930](#)
 - idxOne, [930](#)
 - idxTwo, [930](#)
 - IsEqual, [930](#)
 - QuadraticTerm, [929](#)
 - QuadraticTerm, [929](#)
- QuadraticTerms, [930](#)
 - ~QuadraticTerms, [931](#)
 - coefficients, [931](#)
 - QuadraticTerms, [931](#)
 - QuadraticTerms, [931](#)
 - rowIndexes, [931](#)
 - varOneIndexes, [931](#)

- varTwoIndexes, [931](#)
- quit
 - OSCommandLine, [399](#)
 - osOptionsStruc, [720](#)
- REFERENCEMATRIXIDXATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- REQUESTEDSTARTTIMEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- REQUESTEDSTARTTIMESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- REQUIREDDIRECTORIESEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- REQUIREDDIRECTORIESSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- REQUIREDFILESEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- REQUIREDFILESSTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1216](#)
 - OSParseosrl.tab.hpp, [1289](#)
- ROTATEDQUADRATICCONEEND
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- ROTATEDQUADRATICCONESTART
 - OSParseosil.tab.hpp, [1131](#)
 - OSParseosol.tab.hpp, [1206](#)
 - OSParseosrl.tab.hpp, [1279](#)
- ROWMAJORATT
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- ROWOFFSETSEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- ROWOFFSETSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1155](#)
 - OSParseosol.tab.hpp, [1211](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- RCVBUFSIZE

- OSWSUtil.h, [1045](#)
- REQUIREDFILESEND
 - OSParseosol.tab.hpp, [1184](#)
- REQUIREDFILESSTART
 - OSParseosol.tab.hpp, [1184](#)
- ROWMAJORATT
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- ROWOFFSETSEND
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- ROWOFFSETSSTART
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1270](#)
- readCommandLine
 - OSCommandLineReader, [400](#)
- readNI
 - OSnl2OS, [502](#)
- readOSiL
 - OSiLReader, [432](#)
- readOSoL
 - OSoLReader, [672](#)
- readOSrL
 - OSrLReader, [828](#)
- referenceConIdx
 - DualCone, [122](#)
 - PolarCone, [912](#)
- ReferenceCount
 - OSReferencedObject, [729](#)
- referenceMatrixIdx
 - OSiLParserData, [427](#)
 - PolyhedralCone, [916](#)
- referenceMatrixIdxPresent
 - OSiLParserData, [427](#)
- ReleaseRef
 - OSReferencedObject, [730](#)
- requestedStartTime
 - JobOption, [243](#)
 - OSoLParserData, [660](#)
- requestedStartTimePresent
 - OSoLParserData, [660](#)
- requiredDirectories
 - JobOption, [243](#)
- requiredDirectoriesPresent
 - OSoLParserData, [660](#)
- requiredFiles
 - JobOption, [243](#)
- requiredFilesPresent
 - OSoLParserData, [660](#)
- reset_options
 - OSCommandLine, [395](#)
- resetOptions
 - osOptionsStruc, [716](#)
- resultHeader
 - OSResult, [810](#)
- retrieve
 - OShL, [418](#)
 - OSSolverAgent, [839](#)
- returnBasisStatus
 - OSParameters.h, [1347](#)
- returnBasisStatusString
 - OSParameters.h, [1347](#)
- returnCPUSpeedUnit
 - OSParameters.h, [1345](#)
- returnConReferenceValueType
 - OSParameters.h, [1348](#)
- returnConReferenceValueTypeString
 - OSParameters.h, [1348](#)
- returnConeType
 - OSParameters.h, [1349](#)
- returnExprShapeString
 - OSParameters.h, [1348](#)
- returnGeneralResultStatus
 - OSParameters.h, [1346](#)
- returnJobStatus
 - OSParameters.h, [1347](#)
- returnLocationType
 - OSParameters.h, [1346](#)
- returnMatrixConstructorType
 - OSParameters.h, [1348](#)
- returnMatrixSymmetry
 - OSParameters.h, [1348](#)
- returnMatrixSymmetryString
 - OSParameters.h, [1348](#)
- returnMatrixType
 - OSParameters.h, [1347](#)
- returnMatrixTypeString
 - OSParameters.h, [1348](#)
- returnNIExprShape
 - OSParameters.h, [1348](#)
- returnServiceType
 - OSParameters.h, [1346](#)
- returnSolutionStatus
 - OSParameters.h, [1347](#)
- returnSolutionSubstatusType
 - OSParameters.h, [1347](#)
- returnStorageUnit
 - OSParameters.h, [1345](#)
- returnSystemCurrentState
 - OSParameters.h, [1347](#)
- returnTimeCategory
 - OSParameters.h, [1346](#)
- returnTimeType
 - OSParameters.h, [1346](#)
- returnTimeUnit

- OSParameters.h, 1346
- returnTransportType
 - OSParameters.h, 1346
- returnVarType
 - OSParameters.h, 1347
- reverseAD
 - OSInstance, 480
- RotatedQuadraticCone, 931
 - ~RotatedQuadraticCone, 933
 - coneType, 935
 - deepCopyFrom, 934
 - distortionMatrixIdx, 935
 - firstAxisDirection, 935
 - getConeInXML, 934
 - getConeName, 934
 - idx, 935
 - isEqual, 934
 - normScaleFactor, 935
 - numberOfColumns, 935
 - numberOfOtherIndexes, 935
 - numberOfRows, 934
 - otherIndexes, 935
 - RotatedQuadraticCone, 933
 - RotatedQuadraticCone, 933
 - secondAxisDirection, 935
 - setRandom, 934
- rowIdx
 - LinearConstraintCoefficients, 260
- rowIndexes
 - QuadraticTerms, 931
- rowMajor
 - MatrixElements, 292
 - OSgLPParserData, 414
- rowMajorPresent
 - OSgLPParserData, 414
- rowOffsets
 - ExpandedMatrixBlocks, 131
 - MatrixBlocks, 283
 - OSgLPParserData, 411
- RowReferenceMatrixElements, 936
 - ~RowReferenceMatrixElements, 938
 - alignsOnBlockBoundary, 938
 - cloneMatrixNode, 939
 - deepCopyFrom, 939
 - getMatrixNodeInXML, 938
 - getMatrixType, 938
 - getNodeName, 938
 - getNodeType, 938
 - isEqual, 939
 - RowReferenceMatrixElements, 938
 - RowReferenceMatrixElements, 938
 - setRandom, 939
 - values, 939
- runSolver
 - OSRunSolver.h, 1318
- SCALARMULTIPLIERATT
 - OSParseosil.tab.hpp, 1135, 1145, 1154
 - OSParseosol.tab.hpp, 1210, 1220, 1229
 - OSParseosrl.tab.hpp, 1283, 1292, 1302
- SCHEDULEDSTARTTIMEEND
 - OSParseosil.tab.hpp, 1152
 - OSParseosol.tab.hpp, 1227
 - OSParseosrl.tab.hpp, 1299
- SCHEDULEDSTARTTIMESTART
 - OSParseosil.tab.hpp, 1152
 - OSParseosol.tab.hpp, 1227
 - OSParseosrl.tab.hpp, 1299
- SECONDAXISDIRECTIONATT
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- SEMIDEFINITECONEEND
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- SEMIDEFINITECONESTART
 - OSParseosil.tab.hpp, 1131
 - OSParseosol.tab.hpp, 1206
 - OSParseosrl.tab.hpp, 1279
- SEMIDEFINITENESSATT
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280
- SERVICEEND
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1297
- SERVICENAMEEND
 - OSParseosil.tab.hpp, 1140, 1152
 - OSParseosol.tab.hpp, 1215, 1227
 - OSParseosrl.tab.hpp, 1288, 1299
- SERVICENAMESTART
 - OSParseosil.tab.hpp, 1140, 1152
 - OSParseosol.tab.hpp, 1215, 1227
 - OSParseosrl.tab.hpp, 1288, 1299
- SERVICESTART
 - OSParseosil.tab.hpp, 1140, 1150
 - OSParseosol.tab.hpp, 1215, 1225
 - OSParseosrl.tab.hpp, 1288, 1297
- SERVICETYPEEND
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289
- SERVICETYPESTART
 - OSParseosil.tab.hpp, 1141
 - OSParseosol.tab.hpp, 1216
 - OSParseosrl.tab.hpp, 1289

- SERVICEURIEND
 - OSParseosil.tab.hpp, [1140](#), [1152](#)
 - OSParseosol.tab.hpp, [1215](#), [1227](#)
 - OSParseosrl.tab.hpp, [1288](#), [1300](#)
- SERVICEURISTART
 - OSParseosil.tab.hpp, [1140](#), [1152](#)
 - OSParseosol.tab.hpp, [1215](#), [1227](#)
 - OSParseosrl.tab.hpp, [1288](#), [1299](#)
- SERVICEUTILIZATIONEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SERVICEUTILIZATIONSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SHAPEATT
 - OSParseosil.tab.hpp, [1134](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1301](#)
- SINEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- SINSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- SIZEOFATT
 - OSParseosil.tab.hpp, [1134](#), [1143](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1218](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1291](#), [1297](#)
- SOLUTIONEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLUTIONSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLVERATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- SOLVERINVOKEDEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLVERINVOKEDSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLVEROPTIONEND
 - OSParseosil.tab.hpp, [1143](#)
- OSParseosol.tab.hpp, [1218](#)
- OSParseosrl.tab.hpp, [1291](#)
- SOLVEROPTIONSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- SOLVEROPTIONSSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- SOLVEROPTIONSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1218](#)
 - OSParseosrl.tab.hpp, [1291](#)
- SOLVEROUTPUTEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLVEROUTPUTSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SOLVERTOINVOKEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- SOLVERTOINVOKESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- SOSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- SOSIDXATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1287](#)
- SOSSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- SOSVARIABLEBRANCHINGWEIGHTSEND
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- SOSVARIABLEBRANCHINGWEIGHTSSTART
 - OSParseosil.tab.hpp, [1143](#)
 - OSParseosol.tab.hpp, [1217](#)
 - OSParseosrl.tab.hpp, [1290](#)
- SQRTEND
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)

- SQRTSTART
 - OSParseosil.tab.hpp, [1136](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1230](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- SQUAREEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- SQUARESTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1304](#)
- STAGEEND
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1280](#)
- STAGESEND
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1280](#)
- STAGESSTART
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- STAGESTART
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1280](#)
- STARTATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1281](#)
- STARTIDXATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1208](#)
 - OSParseosrl.tab.hpp, [1281](#)
- STARTVECTOREND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- STARTVECTORSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- STATUSEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- STATUSSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SUBMITTIMEEND
 - OSParseosil.tab.hpp, [1152](#)
- OSParseosol.tab.hpp, [1227](#)
- OSParseosrl.tab.hpp, [1300](#)
- SUBMITTIMESTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SUBSTATUSEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SUBSTATUSSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SUMEND
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- SUMSTART
 - OSParseosil.tab.hpp, [1137](#), [1146](#), [1156](#)
 - OSParseosol.tab.hpp, [1211](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1284](#), [1294](#), [1303](#)
- SUPERBASICEND
 - OSParseosil.tab.hpp, [1142](#), [1152](#)
 - OSParseosol.tab.hpp, [1217](#), [1227](#)
 - OSParseosrl.tab.hpp, [1290](#), [1300](#)
- SUPERBASICSTART
 - OSParseosil.tab.hpp, [1142](#), [1152](#)
 - OSParseosol.tab.hpp, [1217](#), [1227](#)
 - OSParseosrl.tab.hpp, [1290](#), [1300](#)
- SYMMETRYATT
 - OSParseosil.tab.hpp, [1135](#), [1144](#), [1154](#)
 - OSParseosol.tab.hpp, [1209](#), [1219](#), [1229](#)
 - OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)
- SYSTEMEND
 - OSParseosil.tab.hpp, [1140](#), [1150](#)
 - OSParseosol.tab.hpp, [1215](#), [1225](#)
 - OSParseosrl.tab.hpp, [1288](#), [1297](#)
- SYSTEMINFORMATIONEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SYSTEMINFORMATIONSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- SYSTEMSTART
 - OSParseosil.tab.hpp, [1140](#), [1150](#)
 - OSParseosol.tab.hpp, [1215](#), [1225](#)
 - OSParseosrl.tab.hpp, [1288](#), [1297](#)
- sAgentAddress
 - OSMatlab, [488](#)
- SCALARMULTIPLIERATT
 - OSParseosil.tab.hpp, [1120](#)

- OSParseosol.tab.hpp, [1194](#)
- OSParseosrl.tab.hpp, [1267](#)
- SEMIDEFINITECONEEND
 - OSParseosil.tab.hpp, [1109](#)
- SEMIDEFINITENESSATT
 - OSParseosil.tab.hpp, [1111](#)
- SERVICEEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1254](#)
- SERVICENAMEEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1260](#)
- SERVICENAMESTART
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1260](#)
- SERVICESTART
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1254](#)
- SERVICETYPEEND
 - OSParseosol.tab.hpp, [1183](#)
- SERVICETYPESTART
 - OSParseosol.tab.hpp, [1183](#)
- SERVICEURIEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1260](#)
- SERVICEURISTART
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1260](#)
- SHAPEATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- SINEND
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- SINSTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- SIZEOFATT
 - OSParseosil.tab.hpp, [1117](#)
 - OSParseosol.tab.hpp, [1190](#)
 - OSParseosrl.tab.hpp, [1252](#)
- SOAPify
 - WSUtil, [1036](#)
- SOLUTIONEND
 - OSParseosrl.tab.hpp, [1261](#)
- SOLUTIONSTART
 - OSParseosrl.tab.hpp, [1260](#)
- SOLVERATT
 - OSParseosol.tab.hpp, [1180](#)
- SOLVERINVOKEDEND
 - OSParseosrl.tab.hpp, [1261](#)
- SOLVERINVOKEDSTART
 - OSParseosrl.tab.hpp, [1261](#)
- SOLVEROPTIONEND
 - OSParseosol.tab.hpp, [1189](#)
- SOLVEROPTIONSEND
 - OSParseosol.tab.hpp, [1189](#)
- SOLVEROPTIONSSTART
 - OSParseosol.tab.hpp, [1189](#)
- SOLVEROPTIONSTART
 - OSParseosol.tab.hpp, [1189](#)
- SOLVEROUTPUTEND
 - OSParseosrl.tab.hpp, [1261](#)
- SOLVEROUTPUTSTART
 - OSParseosrl.tab.hpp, [1261](#)
- SOLVERTOINVOKEEND
 - OSParseosol.tab.hpp, [1182](#)
- SOLVERTOINVOKESTART
 - OSParseosol.tab.hpp, [1182](#)
- SOSEND
 - OSParseosol.tab.hpp, [1188](#)
- SOSIDXATT
 - OSParseosol.tab.hpp, [1178](#)
- SOSSTART
 - OSParseosol.tab.hpp, [1188](#)
- SOSVariableBranchingWeights, [963](#)
 - ~SOSVariableBranchingWeights, [965](#)
 - addSOS, [966](#)
 - deepCopyFrom, [966](#)
 - isEqual, [965](#)
 - numberOfSOS, [966](#)
 - SOSVariableBranchingWeights, [965](#)
 - setRandom, [965](#)
 - setSOS, [966](#)
 - sos, [966](#)
 - SOSVariableBranchingWeights, [965](#)
- SOSWeights, [966](#)
 - ~SOSWeights, [968](#)
 - addVar, [969](#)
 - deepCopyFrom, [969](#)
 - groupWeight, [969](#)
 - isEqual, [968](#)
 - numberOfVar, [969](#)
 - SOSWeights, [968](#)
 - setRandom, [968](#)
 - setVar, [969](#)
 - sosIdx, [969](#)
 - SOSWeights, [968](#)
 - var, [970](#)
- SQRTEND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- SQRTSTART
 - OSParseosil.tab.hpp, [1124](#)

- OSParseosol.tab.hpp, [1199](#)
- OSParseosrl.tab.hpp, [1272](#)
- SQUAREEND
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- SQUARESTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- sSolverName
 - DefaultSolver, [114](#)
 - OSMatlab, [488](#)
- STAGEEND
 - OSParseosil.tab.hpp, [1113](#)
- STAGESEND
 - OSParseosil.tab.hpp, [1113](#)
- STAGESSTART
 - OSParseosil.tab.hpp, [1113](#)
- STAGESTART
 - OSParseosil.tab.hpp, [1113](#)
- STARTATT
 - OSParseosil.tab.hpp, [1113](#)
- STARTIDXATT
 - OSParseosil.tab.hpp, [1114](#)
- STARTVECTOREND
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- STARTVECTORSTART
 - OSParseosil.tab.hpp, [1120](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- STATUSEND
 - OSParseosrl.tab.hpp, [1261](#)
- STATUSSTART
 - OSParseosrl.tab.hpp, [1261](#)
- SUBMITTIMEEND
 - OSParseosrl.tab.hpp, [1261](#)
- SUBMITTimestart
 - OSParseosrl.tab.hpp, [1261](#)
- SUBSTATUSEND
 - OSParseosrl.tab.hpp, [1261](#)
- SUBSTATUSSTART
 - OSParseosrl.tab.hpp, [1261](#)
- SUMEND
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- SUMSTART
 - OSParseosil.tab.hpp, [1124](#)
 - OSParseosol.tab.hpp, [1199](#)
 - OSParseosrl.tab.hpp, [1272](#)
- SUPERBASICEND
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1261](#)
- SUPERBASICSTART
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1261](#)
- SYMMETRYATT
 - OSParseosil.tab.hpp, [1118](#)
 - OSParseosol.tab.hpp, [1193](#)
 - OSParseosrl.tab.hpp, [1266](#)
- SYSTEMEND
 - OSParseosol.tab.hpp, [1181](#)
 - OSParseosrl.tab.hpp, [1254](#)
- SYSTEMINFORMATIONEND
 - OSParseosrl.tab.hpp, [1262](#)
- SYSTEMSTART
 - OSParseosol.tab.hpp, [1180](#)
 - OSParseosrl.tab.hpp, [1254](#)
- ScalarExpressionTree, [940](#)
 - ~ScalarExpressionTree, [942](#)
 - calculateFunction, [943](#)
 - getPostfixFromExpressionTree, [942](#)
 - getPrefixFromExpressionTree, [942](#)
 - getVariableIndicesMap, [942](#)
 - IsEqual, [942](#)
 - m_treeRoot, [943](#)
 - ScalarExpressionTree, [942](#)
 - ScalarExpressionTree, [942](#)
- scalarMultiplier
 - BaseMatrix, [36](#)
 - OSgLPParserData, [413](#)
- scalarMultiplierPresent
 - OSgLPParserData, [414](#)
- scanner
 - OSgLPParserData, [410](#)
 - OSiLPParserData, [423](#)
 - OSoLPParserData, [671](#)
 - OSrLPParserData, [818](#)
- scheduledStartTime
 - JobResult, [247](#)
- scheduledStartTimePresent
 - OSrLPParserData, [825](#)
- scope
 - OS_AMPL_SUFFIX, [391](#)
- secondAxisDirection
 - OSiLPParserData, [428](#)
 - RotatedQuadraticCone, [935](#)
- secondAxisDirectionPresent
 - OSiLPParserData, [428](#)
- selectSolver
 - OSRunSolver.h, [1319](#)
- SemidefiniteCone, [943](#)
 - ~SemidefiniteCone, [945](#)
 - coneType, [947](#)
 - deepCopyFrom, [946](#)

- getConeInXML, 946
- getConeName, 945
- idx, 947
- IsEqual, 946
- isPositiveSemiDefinite, 947
- numberOfColumns, 946
- numberOfOtherIndexes, 947
- numberOfRows, 946
- otherIndexes, 947
- SemidefiniteCone, 945
- SemidefiniteCone, 945
- semidefiniteness, 947
- setRandom, 946
- semidefiniteness
 - OSiLParserData, 428
 - SemidefiniteCone, 947
- semidefinitenessPresent
 - OSiLParserData, 428
- send
 - OShL, 417
 - OSSolverAgent, 839
- sendSOAPMessage
 - WSUtil, 1035
- service
 - OSOption, 713
 - OSResult, 810
- serviceCurrentJobCountPresent
 - OSrLParserData, 824
- serviceCurrentStatePresent
 - OSrLParserData, 824
- serviceLocation
 - OSCommandLine, 396
 - osOptionsStruc, 717
- serviceMethod
 - OSCommandLine, 396
 - osOptionsStruc, 717
- serviceName
 - GeneralOption, 152
 - GeneralResult, 156
- serviceNamePresent
 - OSoLParserData, 658
- ServiceOption, 947
 - ~ServiceOption, 949
 - deepCopyFrom, 950
 - IsEqual, 949
 - otherOptions, 950
 - ServiceOption, 949
 - ServiceOption, 949
 - setRandom, 949
 - type, 950
- ServiceResult, 950
 - ~ServiceResult, 952
 - currentJobCount, 953
 - currentState, 953
 - IsEqual, 952
 - otherResults, 953
 - ServiceResult, 952
 - serviceUtilization, 953
 - ServiceResult, 952
 - setRandom, 952
 - timeServiceStarted, 953
 - totalJobsSoFar, 953
- serviceTotalJobsSoFarPresent
 - OSrLParserData, 824
- serviceTypePresent
 - OSoLParserData, 660
- serviceURI
 - GeneralOption, 152
 - GeneralResult, 156
- serviceURIPresent
 - OSoLParserData, 658
- serviceUtilization
 - ServiceResult, 953
- serviceUtilizationPresent
 - OSrLParserData, 825
- setASL
 - OSnl2OS, 502
- setActualStartTime
 - OSResult, 770
- setAllPrintLevels
 - OSOutputChannel, 726
- setAnOtherConstraintOption
 - OSOption, 712
- setAnOtherGeneralOption
 - OSOption, 705
- setAnOtherJobOption
 - OSOption, 708
- setAnOtherObjectiveOption
 - OSOption, 710
- setAnOtherServiceOption
 - OSOption, 706
- setAnOtherSolutionResult
 - OSResult, 808
- setAnOtherSystemOption
 - OSOption, 706
- setAnOtherVariableOption
 - OSOption, 709
- setAnOtherVariableResultDense
 - OSResult, 784, 785
- setAnOtherVariableResultSparse
 - OSResult, 783, 784
- setAnotherDirectoryToDelete
 - OSOption, 707
- setAnotherDirectoryToMake
 - OSOption, 707
- setAnotherFileToDelete
 - OSOption, 707
- setAnotherFileToMake

OSOption, 707
setAnotherInitBasisStatus
 OSOption, 708
setAnotherInitConValue
 OSOption, 711
setAnotherInitDualVarValue
 OSOption, 711
setAnotherInitObjBound
 OSOption, 710
setAnotherInitObjValue
 OSOption, 710
setAnotherInitVarValue
 OSOption, 708
setAnotherInitVarValueString
 OSOption, 708
setAnotherInputDirectoryToMove
 OSOption, 707
setAnotherInputFileToMove
 OSOption, 707
setAnotherIntegerVariableBranchingWeight
 OSOption, 709
setAnotherJobDependency
 OSOption, 707
setAnotherOutputDirectoryToMove
 OSOption, 707
setAnotherOutputFileToMove
 OSOption, 707
setAnotherProcessToKill
 OSOption, 708
setAnotherRequiredDirectory
 OSOption, 707
setAnotherRequiredFile
 OSOption, 707
setAnotherSOSVariableBranchingWeight
 OSOption, 709
setAnotherSolverOption
 OSOption, 713
setAvailableCPUNumberDescription
 OSResult, 766
setAvailableCPUNumberValue
 OSResult, 767
setAvailableCPUSpeedDescription
 OSResult, 766
setAvailableCPUSpeedUnit
 OSResult, 766
setAvailableCPUSpeedValue
 OSResult, 766
setAvailableDiskSpaceDescription
 OSResult, 764
setAvailableDiskSpaceUnit
 OSResult, 764
setAvailableDiskSpaceValue
 OSResult, 765
setAvailableMemoryDescription
 OSResult, 765
setAvailableMemoryUnit
 OSResult, 765
setAvailableMemoryValue
 OSResult, 765
setBasisStatus
 OSResult, 782
setCoinPackedMatrix
 CoinSolver, 56
setCon
 InitConstraintValues, 172
 InitDualVariableValues, 178
 OtherConstraintOption, 848
setConeNumber
 OSInstance, 468
setConstraintNumber
 OSInstance, 464
 OSResult, 776
setConstraintValuesDense
 OSResult, 800
setConstraints
 OSInstance, 465
setContact
 OSOption, 704, 705
setContactTransportType
 OSOption, 705
setCurrentJobCount
 OSResult, 768
setCurrentState
 OSResult, 768
setDirectoriesToDelete
 OSOption, 707
setDirectoriesToMake
 OSOption, 707
setDualValue
 OSResult, 800
setDualVariableValuesDense
 OSResult, 800
setDualVariableValuesSparse
 OSResult, 799
setFilesToDelete
 OSOption, 707
setFilesToMake
 OSOption, 707
setGeneralMessage
 OSResult, 761
setGeneralStatus
 OSResult, 760
setGeneralStatusDescription
 OSResult, 761
setGeneralStatusType
 OSResult, 760
setGeneralSubstatusDescription
 OSResult, 761

- setGeneralSubstatusName
 - OSResult, [761](#)
- setHeader
 - GeneralFileHeader, [142](#)
 - OSOption, [686](#)
 - OSResult, [745](#)
- setIBVar
 - OSnl2OS, [503](#)
- setInitBasisStatus
 - OSOption, [708](#)
- setInitBasisStatusDense
 - OSOption, [708](#)
- setInitBasisStatusSparse
 - OSOption, [708](#)
- setInitConValues
 - OSOption, [711](#)
- setInitConValuesDense
 - OSOption, [711](#)
- setInitConValuesSparse
 - OSOption, [711](#)
- setInitDualValues
 - OSOption, [711](#)
- setInitDualVarValuesDense
 - OSOption, [711](#)
- setInitDualVarValuesSparse
 - OSOption, [711](#)
- setInitObjBounds
 - OSOption, [710](#)
- setInitObjBoundsDense
 - OSOption, [710](#)
- setInitObjBoundsSparse
 - OSOption, [710](#)
- setInitObjValues
 - OSOption, [710](#)
- setInitObjValuesDense
 - OSOption, [710](#)
- setInitObjValuesSparse
 - OSOption, [710](#)
- setInitVarValues
 - OSOption, [708](#)
- setInitVarValuesDense
 - OSOption, [708](#)
- setInitVarValuesSparse
 - OSOption, [708](#)
- setInitVarValuesString
 - OSOption, [708](#)
- setInitVarValuesStringDense
 - OSOption, [708](#)
- setInitVarValuesStringSparse
 - OSOption, [708](#)
- setInputDirectoriesToMove
 - OSOption, [707](#)
- setInputFilesToMove
 - OSOption, [707](#)
- setInstanceCreator
 - OSInstance, [462](#)
- setInstanceDescription
 - OSInstance, [461](#)
- setInstanceLicence
 - OSInstance, [462](#)
- setInstanceLocation
 - OSOption, [704](#)
- setInstanceLocationType
 - OSOption, [704](#)
- setInstanceName
 - OSInstance, [461](#)
 - OSOption, [704](#)
 - OSResult, [762](#)
- setInstanceSource
 - OSInstance, [461](#)
- setIntVector
 - BasisStatus, [39](#)
 - IntVector, [228](#)
- setIntegerVariableBranchingWeights
 - OSOption, [708](#)
- setIntegerVariableBranchingWeightsDense
 - OSOption, [709](#)
- setIntegerVariableBranchingWeightsSparse
 - OSOption, [709](#)
- setJobDependencies
 - OSOption, [706](#)
- setJobEndTime
 - OSResult, [771](#)
- setJobID
 - JobDependencies, [240](#)
 - OSmps2OS, [496](#)
 - OSnl2OS, [502](#)
 - OSOption, [704](#)
 - OSResult, [762](#)
- setJobStatus
 - OSResult, [770](#)
- setJobSubmitTime
 - OSResult, [770](#)
- setLicense
 - OSOption, [704](#)
- setLinearConstraintCoefficients
 - OSInstance, [465](#)
- setMatrix
 - OSMatrix, [493](#)
- setMatrixNumber
 - OSInstance, [467](#)
- setMaxTime
 - OSOption, [706](#)
- setMaxTimeUnit
 - OSOption, [706](#)
- setMinCPUNumber
 - OSOption, [706](#)
- setMinCPUSpeed

- OSOption, 706
- setMinCPUSpeedUnit
 - OSOption, 706
- setMinDiskSpace
 - OSOption, 705
- setMinDiskSpaceUnit
 - OSOption, 705
- setMinMemorySize
 - OSOption, 705, 706
- setMinMemoryUnit
 - OSOption, 706
- setNonlinearExpressions
 - OSInstance, 467
- setNumberOfConstraints
 - OSOption, 708
- setNumberOfDualValues
 - OSResult, 798
- setNumberOfDualVariableValues
 - OSResult, 799
- setNumberOfGeneralSubstatuses
 - OSResult, 760
- setNumberOfObjValues
 - OSResult, 791
- setNumberOfObjectiveValues
 - OSResult, 792
- setNumberOfObjectives
 - OSOption, 708
- setNumberOfOtherConstraintOptions
 - OSOption, 711
- setNumberOfOtherConstraintResults
 - OSResult, 798
- setNumberOfOtherGeneralResults
 - OSResult, 763
- setNumberOfOtherJobResults
 - OSResult, 775
- setNumberOfOtherObjectiveOptions
 - OSOption, 710
- setNumberOfOtherObjectiveResults
 - OSResult, 791
- setNumberOfOtherServiceResults
 - OSResult, 769
- setNumberOfOtherSolutionResults
 - OSResult, 805
- setNumberOfOtherSystemResults
 - OSResult, 767
- setNumberOfOtherVariableOptions
 - OSOption, 709
- setNumberOfOtherVariableResults
 - OSResult, 783
- setNumberOfPrimalVariableValues
 - OSResult, 780
- setNumberOfQuadraticTerms
 - OSInstance, 466
- setNumberOfSolutionSubstatuses
 - OSResult, 777
- setNumberOfSolverOptions
 - OSOption, 712
- setNumberOfSolverOutputs
 - OSResult, 808
- setNumberOfTimes
 - OSResult, 772
- setNumberOfVarValues
 - OSResult, 781
- setNumberOfVarValuesString
 - OSResult, 782
- setNumberOfVariables
 - OSOption, 708
- setObj
 - InitObjectiveBounds, 191
 - InitObjectiveValues, 195
 - OtherObjectiveOption, 857
- setObjValue
 - OSResult, 793
- setObjectiveNumber
 - OSInstance, 463
 - OSResult, 776
- setObjectiveValuesDense
 - OSResult, 792
- setObjectiveValuesSparse
 - OSResult, 792
- setObjectives
 - OSInstance, 464
- setOptionDbI
 - OSOption, 713
- setOptionInt
 - OSOption, 713
- setOptionStr
 - OSOption, 713
- setOsI
 - OSmps2OS, 496
 - OSnl2OS, 502
- setOther
 - ConstraintOption, 85
 - ObjectiveOption, 356
 - OtherOptions, 875
 - VariableOption, 1012
- setOtherConstraintOptionAttributes
 - OSOption, 712
- setOtherConstraintOptionCon
 - OSOption, 712
- setOtherConstraintOptions
 - OSOption, 711
- setOtherConstraintResultCon
 - OSResult, 805
- setOtherConstraintResultConIdx
 - OSResult, 804
- setOtherConstraintResultConName
 - OSResult, 805

- setOtherConstraintResultConType
 - OSResult, [802](#)
- setOtherConstraintResultDescription
 - OSResult, [804](#)
- setOtherConstraintResultEnumType
 - OSResult, [803](#)
- setOtherConstraintResultName
 - OSResult, [801](#)
- setOtherConstraintResultNumberOfCon
 - OSResult, [801](#)
- setOtherConstraintResultNumberOfEnumerations
 - OSResult, [801](#)
- setOtherConstraintResultType
 - OSResult, [802](#)
- setOtherConstraintResultValue
 - OSResult, [803](#)
- setOtherGeneralOptions
 - OSOption, [705](#)
- setOtherGeneralResultDescription
 - OSResult, [764](#)
- setOtherGeneralResultName
 - OSResult, [763](#)
- setOtherGeneralResultValue
 - OSResult, [763](#)
- setOtherJobOptions
 - OSOption, [708](#)
- setOtherJobResultDescription
 - OSResult, [775](#)
- setOtherJobResultName
 - OSResult, [775](#)
- setOtherJobResultValue
 - OSResult, [775](#)
- setOtherObjectiveOptionAttributes
 - OSOption, [710](#)
- setOtherObjectiveOptionObj
 - OSOption, [711](#)
- setOtherObjectiveOptions
 - OSOption, [710](#)
- setOtherObjectiveResultDescription
 - OSResult, [796](#)
- setOtherObjectiveResultEnumType
 - OSResult, [795](#)
- setOtherObjectiveResultName
 - OSResult, [794](#)
- setOtherObjectiveResultNumberOfEnumerations
 - OSResult, [794](#)
- setOtherObjectiveResultNumberOfObj
 - OSResult, [793](#)
- setOtherObjectiveResultObj
 - OSResult, [798](#)
- setOtherObjectiveResultObjIdx
 - OSResult, [797](#)
- setOtherObjectiveResultObjName
 - OSResult, [797](#)
- setOtherObjectiveResultObjType
 - OSResult, [795](#)
- setOtherObjectiveResultType
 - OSResult, [794](#)
- setOtherObjectiveResultValue
 - OSResult, [796](#)
- setOtherOptionEnumeration
 - OSOption, [709](#)
 - OSResult, [790](#)
 - OtherOptionEnumeration, [872](#)
- setOtherServiceOptions
 - OSOption, [706](#)
- setOtherServiceResultDescription
 - OSResult, [770](#)
- setOtherServiceResultName
 - OSResult, [769](#)
- setOtherServiceResultValue
 - OSResult, [769](#)
- setOtherSolutionResultCategory
 - OSResult, [807](#)
- setOtherSolutionResultDescription
 - OSResult, [807](#)
- setOtherSolutionResultItem
 - OSResult, [807](#)
- setOtherSolutionResultName
 - OSResult, [806](#)
- setOtherSolutionResultNumberOfItems
 - OSResult, [807](#)
- setOtherSolutionResultValue
 - OSResult, [806](#)
- setOtherSystemOptions
 - OSOption, [706](#)
- setOtherSystemResultDescription
 - OSResult, [767](#)
- setOtherSystemResultName
 - OSResult, [767](#)
- setOtherSystemResultValue
 - OSResult, [767](#)
- setOtherVariableOptionAttributes
 - OSOption, [709](#)
- setOtherVariableOptionVar
 - OSOption, [710](#)
- setOtherVariableOptions
 - OSOption, [709](#)
- setOtherVariableResultDescription
 - OSResult, [788](#)
- setOtherVariableResultEnumType
 - OSResult, [788](#)
- setOtherVariableResultName
 - OSResult, [786](#)
- setOtherVariableResultNumberOfEnumerations
 - OSResult, [786](#)
- setOtherVariableResultNumberOfVar
 - OSResult, [786](#)

- setOtherVariableResultType
 - OSResult, [787](#)
- setOtherVariableResultValue
 - OSResult, [788](#)
- setOtherVariableResultVar
 - OSResult, [790](#)
- setOtherVariableResultVarIdx
 - OSResult, [789](#)
- setOtherVariableResultVarName
 - OSResult, [789](#)
- setOtherVariableResultVarType
 - OSResult, [787](#)
- setOutputDirectoriesToMove
 - OSOption, [707](#)
- setOutputFilesToMove
 - OSOption, [707](#)
- setPassword
 - OSOption, [704](#)
- setPath
 - DirectoriesAndFiles, [117](#)
- setPathPair
 - PathPairs, [908](#)
- setPathPairs
 - OSOption, [707](#)
- setPrimalVariableValuesDense
 - OSResult, [780](#)
- setPrimalVariableValuesSparse
 - OSResult, [780](#)
- SetPrintLevel
 - OSOutput, [723](#), [724](#)
- setPrintLevel
 - OSOutputChannel, [726](#)
- setProcess
 - Processes, [919](#)
- setProcessesToKill
 - OSOption, [708](#)
- setQuadraticCoefficients
 - OSInstance, [466](#)
- setQuadraticTermsInNonlinearExpressions
 - OSInstance, [467](#)
- setRandom
 - BasisStatus, [38](#)
 - BranchingWeight, [51](#)
 - CompletelyPositiveMatricesCone, [59](#)
 - Cone, [63](#)
 - Cones, [66](#)
 - ConReferenceMatrixElement, [68](#)
 - ConReferenceMatrixElements, [72](#)
 - ConReferenceMatrixValues, [74](#)
 - ConstantMatrixElements, [78](#)
 - ConstantMatrixValues, [80](#)
 - ConstraintOption, [84](#)
 - ConstraintSolution, [89](#)
 - ContactOption, [92](#)
 - CopositiveMatricesCone, [95](#)
 - CPUNumber, [102](#)
 - CPU Speed, [105](#)
 - DirectoriesAndFiles, [116](#)
 - DualCone, [121](#)
 - DualVariableValues, [124](#)
 - DualVarValue, [126](#)
 - GeneralFileHeader, [141](#)
 - GeneralMatrixElements, [146](#)
 - GeneralMatrixValues, [148](#)
 - GeneralOption, [152](#)
 - GeneralResult, [156](#)
 - GeneralStatus, [162](#)
 - GeneralSubstatus, [164](#)
 - InitBasStatus, [168](#)
 - InitConstraintValues, [171](#)
 - InitConValue, [175](#)
 - InitDualVariableValues, [177](#)
 - InitDualVarValue, [181](#)
 - InitialBasisStatus, [184](#)
 - InitObjBound, [187](#)
 - InitObjectiveBounds, [190](#)
 - InitObjectiveValues, [194](#)
 - InitObjValue, [198](#)
 - InitVariableValues, [201](#)
 - InitVariableValuesString, [205](#)
 - InitVarValue, [209](#)
 - InitVarValueString, [212](#)
 - InstanceLocationOption, [218](#)
 - IntegerVariableBranchingWeights, [220](#)
 - IntersectionCone, [225](#)
 - IntVector, [228](#)
 - JobDependencies, [239](#)
 - JobOption, [242](#)
 - JobResult, [247](#)
 - LinearMatrixElement, [262](#)
 - LinearMatrixElements, [266](#)
 - LinearMatrixElementTerm, [268](#)
 - LinearMatrixValues, [270](#)
 - Matrices, [274](#)
 - MatrixBlock, [278](#)
 - MatrixBlocks, [282](#)
 - MatrixNode, [306](#)
 - MatrixProgramming, [313](#)
 - MatrixTransformation, [317](#)
 - MatrixType, [322](#)
 - NonnegativeCone, [347](#)
 - NonpositiveCone, [350](#)
 - ObjectiveOption, [356](#)
 - ObjectiveSolution, [361](#)
 - ObjectiveValues, [364](#)
 - ObjReferenceMatrixElements, [368](#)
 - ObjReferenceMatrixValues, [370](#)
 - ObjValue, [372](#)

OptimizationOption, [375](#)
OptimizationResult, [378](#)
OptimizationSolution, [381](#)
OptimizationSolutionStatus, [384](#)
OptimizationSolutionSubstatus, [387](#)
OrthantCone, [390](#)
OSMatrix, [493](#)
OSOption, [686](#)
OSResult, [745](#)
OtherConOption, [842](#)
OtherConResult, [844](#)
OtherConstraintOption, [848](#)
OtherConstraintResult, [852](#)
OtherObjectiveOption, [856](#)
OtherObjectiveResult, [860](#)
OtherObjOption, [864](#)
OtherObjResult, [866](#)
OtherOption, [869](#)
OtherOptionEnumeration, [871](#)
OtherOptions, [874](#)
OtherResult, [877](#)
OtherResults, [880](#)
OtherSolutionResult, [883](#)
OtherSolutionResults, [885](#)
OtherSolverOutput, [888](#)
OtherVariableOption, [890](#)
OtherVariableResult, [895](#)
OtherVarOption, [899](#)
OtherVarResult, [902](#)
PathPair, [905](#)
PathPairs, [907](#)
PolarCone, [911](#)
PolyhedralCone, [915](#)
Processes, [918](#)
ProductCone, [922](#)
QuadraticCone, [927](#)
RotatedQuadraticCone, [934](#)
RowReferenceMatrixElements, [939](#)
SemidefiniteCone, [946](#)
ServiceOption, [949](#)
ServiceResult, [952](#)
SolverOption, [955](#)
SolverOptions, [959](#)
SolverOutput, [962](#)
SOSVariableBranchingWeights, [965](#)
SOSWeights, [968](#)
StorageCapacity, [980](#)
SystemOption, [983](#)
SystemResult, [986](#)
TimeMeasurement, [1001](#)
TimeSpan, [1004](#)
TimingInformation, [1007](#)
VariableOption, [1012](#)
VariableSolution, [1017](#)
VariableValues, [1020](#)
VariableValuesString, [1022](#)
VarReferenceMatrixElements, [1026](#)
VarReferenceMatrixValues, [1029](#)
VarValue, [1031](#)
VarValueString, [1033](#)
setRequestedStartTime
 OSOption, [706](#)
setRequiredDirectories
 OSOption, [707](#)
setRequiredFiles
 OSOption, [707](#)
setSOS
 SOSVariableBranchingWeights, [966](#)
setSOSVariableBranchingWeights
 OSOption, [709](#)
setScheduledStartTime
 OSResult, [770](#)
setServiceName
 OSOption, [704](#)
 OSResult, [762](#)
setServiceType
 OSOption, [706](#)
setServiceURI
 OSOption, [704](#)
 OSResult, [762](#)
setServiceUtilization
 OSResult, [769](#)
setSolutionMessage
 OSResult, [779](#)
setSolutionNumber
 OSResult, [776](#)
setSolutionStatus
 OSResult, [777](#)
setSolutionStatusDescription
 OSResult, [778](#)
setSolutionStatusType
 OSResult, [777](#)
setSolutionSubstatusDescription
 OSResult, [778](#)
setSolutionSubstatusType
 OSResult, [778](#)
setSolutionTargetObjectiveIdx
 OSResult, [778](#)
setSolutionTargetObjectiveName
 OSResult, [779](#)
setSolutionWeightedObjectives
 OSResult, [779](#)
setSolverInvoked
 OSResult, [763](#)
setSolverOptionContent
 OSOption, [712](#)
setSolverOptions
 BonminSolver, [49](#)

- CoinSolver, [56](#)
- CouenneSolver, [99](#)
- CsdpSolver, [109](#)
- DefaultSolver, [113](#)
- IpoptSolver, [237](#)
- KnitroSolver, [253](#)
- LindoSolver, [257](#)
- OSOption, [713](#)
- SolverOptions, [960](#)
- setSolverOutputCategory
 - OSResult, [809](#)
- setSolverOutputDescription
 - OSResult, [809](#)
- setSolverOutputItem
 - OSResult, [809](#)
- setSolverOutputName
 - OSResult, [808](#)
- setSolverOutputNumberOfItems
 - OSResult, [809](#)
- setSolverToInvoke
 - OSOption, [704](#)
- setSystemInformation
 - OSResult, [764](#)
- setTime
 - OSResult, [771](#)
- setTimeDomain
 - OSInstance, [481](#)
- setTimeDomainInterval
 - OSInstance, [482](#)
- setTimeDomainStageConstraintsOrdered
 - OSInstance, [482](#)
- setTimeDomainStageConstraintsUnordered
 - OSInstance, [482](#)
- setTimeDomainStageObjectivesOrdered
 - OSInstance, [482](#)
- setTimeDomainStageObjectivesUnordered
 - OSInstance, [482](#)
- setTimeDomainStageVariablesOrdered
 - OSInstance, [482](#)
- setTimeDomainStageVariablesUnordered
 - OSInstance, [482](#)
- setTimeDomainStages
 - OSInstance, [481](#)
- setTimeNumber
 - OSResult, [772](#)
- setTimeServiceStarted
 - OSResult, [768](#)
- setTimeStamp
 - OSResult, [763](#)
- setTimingInformation
 - OSResult, [771](#)
- setTotalJobsSoFar
 - OSResult, [768](#)
- setUsedCPUNumberDescription
 - OSResult, [774](#)
- setUsedCPUNumberValue
 - OSResult, [775](#)
- setUsedCPUSpeedDescription
 - OSResult, [774](#)
- setUsedCPUSpeedUnit
 - OSResult, [774](#)
- setUsedCPUSpeedValue
 - OSResult, [774](#)
- setUsedDiskSpaceDescription
 - OSResult, [772](#)
- setUsedDiskSpaceUnit
 - OSResult, [772](#)
- setUsedDiskSpaceValue
 - OSResult, [773](#)
- setUsedMemoryDescription
 - OSResult, [773](#)
- setUsedMemoryUnit
 - OSResult, [773](#)
- setUsedMemoryValue
 - OSResult, [773](#)
- setUserName
 - OSOption, [704](#)
- setVar
 - InitialBasisStatus, [185](#)
 - InitVariableValues, [202](#)
 - InitVariableValuesString, [206](#)
 - IntegerVariableBranchingWeights, [221](#)
 - OSnl2OS, [503](#)
 - OtherVariableOption, [891](#)
 - SOSWeights, [969](#)
- setVarValue
 - OSResult, [781](#)
- setVarValueString
 - OSResult, [782](#)
- setVariableNumber
 - OSInstance, [462](#)
 - OSResult, [776](#)
- setVariables
 - OSInstance, [463](#)
- shape
 - MatrixExpression, [296](#)
 - MatrixTransformation, [318](#)
 - NI, [341](#)
 - OSgLPParserData, [415](#)
 - OSiLPParserData, [430](#)
- shapeAttributePresent
 - OSnLPParserData, [648](#)
- shapePresent
 - OSgLPParserData, [415](#)
 - OSiLPParserData, [430](#)
- solution
 - OptimizationResult, [379](#)
- solutionIdx

- OSrLPParserData, [820](#)
- solve
 - BonminSolver, [49](#)
 - CoinSolver, [55](#)
 - CouenneSolver, [99](#)
 - CsdpSolver, [109](#)
 - DefaultSolver, [113](#)
 - IpoptSolver, [237](#)
 - KnitroSolver, [254](#)
 - LindoSolver, [257](#)
 - OShL, [417](#)
 - OSMatlab, [486](#)
 - OSSolverAgent, [838](#)
- solver
 - OtherConstraintOption, [849](#)
 - OtherObjectiveOption, [857](#)
 - OtherVariableOption, [892](#)
 - SolverOption, [956](#)
- solverAttribute
 - OSnLPParserData, [648](#)
 - OSoLPParserData, [669](#)
- solverAttributePresent
 - OSnLPParserData, [647](#)
 - OSoLPParserData, [669](#)
- solverInvoked
 - GeneralResult, [156](#)
- solverName
 - OSCommandLine, [396](#)
 - osOptionsStruc, [719](#)
- SolverOption, [953](#)
 - ~SolverOption, [955](#)
 - category, [956](#)
 - deepCopyFrom, [956](#)
 - description, [956](#)
 - IsEqual, [955](#)
 - item, [957](#)
 - name, [956](#)
 - numberOfItems, [956](#)
 - setRandom, [955](#)
 - solver, [956](#)
 - SolverOption, [955](#)
 - SolverOption, [955](#)
 - type, [956](#)
 - value, [956](#)
- solverOption
 - SolverOptions, [960](#)
- solverOptionCategoryPresent
 - OSoLPParserData, [667](#)
- solverOptionDescriptionPresent
 - OSoLPParserData, [667](#)
- solverOptionNamePresent
 - OSoLPParserData, [667](#)
- solverOptionSolverPresent
 - OSoLPParserData, [667](#)
- solverOptionTypePresent
 - OSoLPParserData, [667](#)
- solverOptionValuePresent
 - OSoLPParserData, [667](#)
- SolverOptions, [957](#)
 - ~SolverOptions, [959](#)
 - addSolverOption, [960](#)
 - deepCopyFrom, [960](#)
 - IsEqual, [959](#)
 - numberOfSolverOptions, [960](#)
 - setRandom, [959](#)
 - setSolverOptions, [960](#)
 - solverOption, [960](#)
 - SolverOptions, [959](#)
 - SolverOptions, [959](#)
- solverOptions
 - OptimizationOption, [376](#)
- solverOptionsPresent
 - OSoLPParserData, [663](#)
- SolverOutput, [961](#)
 - ~SolverOutput, [962](#)
 - category, [963](#)
 - description, [963](#)
 - IsEqual, [962](#)
 - item, [963](#)
 - name, [963](#)
 - numberOfItems, [963](#)
 - setRandom, [962](#)
 - SolverOutput, [962](#)
 - SolverOutput, [962](#)
- solverOutput
 - OtherSolverOutput, [888](#)
- solverToInvoke
 - GeneralOption, [153](#)
- solverToInvokePresent
 - OSoLPParserData, [658](#)
- solverType
 - OSMatlab, [488](#)
- sos
 - SOSVariableBranchingWeights, [966](#)
- sosConstraints
 - BonminProblem, [45](#)
- sosIdx
 - OSoLPParserData, [665](#)
 - SOSWeights, [969](#)
- sosIdxAttributePresent
 - OSoLPParserData, [664](#)
- sosVariableBranchingWeights
 - VariableOption, [1013](#)
- source
 - GeneralFileHeader, [142](#)
 - OSgLPParserData, [410](#)
- sourcePresent
 - OSgLPParserData, [410](#)

- SparseHessianMatrix, [970](#)
 - ~SparseHessianMatrix, [971](#)
 - bDeleteArrays, [971](#)
 - hessColIdx, [971](#)
 - hessDimension, [971](#)
 - hessRowIdx, [971](#)
 - hessValues, [971](#)
 - SparseHessianMatrix, [971](#)
 - SparseHessianMatrix, [971](#)
- SparseIntVector, [972](#)
 - ~SparseIntVector, [973](#)
 - bDeleteArrays, [973](#)
 - indexes, [973](#)
 - number, [973](#)
 - SparseIntVector, [972](#)
 - SparseIntVector, [972](#)
 - values, [973](#)
- SparseJacobianMatrix, [973](#)
 - ~SparseJacobianMatrix, [974](#)
 - bDeleteArrays, [974](#)
 - conVals, [975](#)
 - indexes, [975](#)
 - SparseJacobianMatrix, [974](#)
 - SparseJacobianMatrix, [974](#)
 - startSize, [974](#)
 - starts, [975](#)
 - valueSize, [975](#)
 - values, [975](#)
- sparseMat
 - OSMatlab, [486](#)
- SparseMatrix, [975](#)
 - ~SparseMatrix, [976](#)
 - bDeleteArrays, [977](#)
 - display, [976](#)
 - indexes, [977](#)
 - isColumnMajor, [977](#)
 - SparseMatrix, [976](#)
 - SparseMatrix, [976](#)
 - startSize, [977](#)
 - starts, [977](#)
 - valueSize, [977](#)
 - values, [977](#)
- SparseVector, [977](#)
 - ~SparseVector, [978](#)
 - bDeleteArrays, [978](#)
 - indexes, [979](#)
 - number, [979](#)
 - SparseVector, [978](#)
 - SparseVector, [978](#)
 - values, [979](#)
- stage
 - TimeDomainStages, [997](#)
- stageConstraintStartIdx
 - OSILParserData, [425](#)
- stageConstraintsON
 - OSILParserData, [424](#)
- stageConstraintsOrdered
 - OSILParserData, [424](#)
- stageObjectiveStartIdx
 - OSILParserData, [425](#)
- stageObjectivesON
 - OSILParserData, [424](#)
- stageObjectivesOrdered
 - OSILParserData, [424](#)
- stageVariableStartIdx
 - OSILParserData, [424](#)
- stageVariablesON
 - OSILParserData, [424](#)
- stageVariablesOrdered
 - OSILParserData, [424](#)
- stageconcount
 - OSILParserData, [425](#)
- stagecount
 - OSILParserData, [424](#)
- stagename
 - OSILParserData, [424](#)
- stagenameON
 - OSILParserData, [424](#)
- stageobjcount
 - OSILParserData, [425](#)
- stages
 - TimeDomain, [988](#)
- stagevarcount
 - OSILParserData, [425](#)
- start
 - LinearConstraintCoefficients, [260](#)
 - MatrixElements, [292](#)
 - TimeDomainInterval, [989](#)
- startIdx
 - TimeDomainStageConstraints, [993](#)
 - TimeDomainStageObjectives, [996](#)
 - TimeDomainStageVariables, [999](#)
- startSize
 - GeneralSparseMatrix, [159](#)
 - SparseJacobianMatrix, [974](#)
 - SparseMatrix, [977](#)
- starts
 - GeneralSparseMatrix, [159](#)
 - SparseJacobianMatrix, [975](#)
 - SparseMatrix, [977](#)
- status
 - BonminProblem, [46](#)
 - BonminSolver, [49](#)
 - CouenneSolver, [100](#)
 - JobResult, [247](#)
 - OptimizationSolution, [382](#)
- statusDescription
 - OSoLParserData, [671](#)

- OSrLParseData, [817](#)
- statusType
 - OSoLParseData, [671](#)
 - OSrLParseData, [817](#)
- StorageCapacity, [979](#)
 - ~StorageCapacity, [980](#)
 - deepCopyFrom, [981](#)
 - description, [981](#)
 - IsEqual, [980](#)
 - setRandom, [980](#)
 - StorageCapacity, [980](#)
 - StorageCapacity, [980](#)
 - unit, [981](#)
 - value, [981](#)
- submitTime
 - JobResult, [247](#)
- substatus
 - GeneralStatus, [162](#)
 - OptimizationSolutionStatus, [385](#)
- sumVec
 - OSnLParseData, [650](#)
- superbasic
 - BasisStatus, [41](#)
- suppressFurtherErrorMessages
 - OSgLParseData, [411](#)
 - OSiLParseData, [430](#)
 - OSnLParseData, [652](#)
 - OSoLParseData, [671](#)
 - OSrLParseData, [827](#)
- sval
 - YYSTYPE, [1038](#)
- symmetry
 - MatrixType, [322](#)
 - OSgLParseData, [412](#)
- symmetryPresent
 - OSgLParseData, [412](#)
- system
 - OSOption, [713](#)
 - OSResult, [810](#)
- systemAvailableCPUNumberPresent
 - OSrLParseData, [824](#)
- systemAvailableCPUSpeedPresent
 - OSrLParseData, [824](#)
- systemAvailableDiskSpacePresent
 - OSrLParseData, [824](#)
- systemAvailableMemoryPresent
 - OSrLParseData, [824](#)
- systemInformation
 - SystemResult, [986](#)
- systemInformationPresent
 - OSrLParseData, [824](#)
- SystemOption, [981](#)
 - ~SystemOption, [983](#)
 - deepCopyFrom, [984](#)

- IsEqual, [983](#)
- minCPUNumber, [984](#)
- minCPUSpeed, [984](#)
- minDiskSpace, [984](#)
- minMemorySize, [984](#)
- otherOptions, [984](#)
- setRandom, [983](#)
- SystemOption, [983](#)
- SystemOption, [983](#)
- SystemResult, [984](#)
 - ~SystemResult, [986](#)
 - availableCPUNumber, [987](#)
 - availableCPUSpeed, [987](#)
 - availableDiskSpace, [987](#)
 - availableMemory, [987](#)
 - IsEqual, [986](#)
 - otherResults, [987](#)
 - setRandom, [986](#)
 - systemInformation, [986](#)
 - SystemResult, [986](#)
 - SystemResult, [986](#)

TARGETMATRIXFIRSTCOLATT

- OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
- OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
- OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)

TARGETMATRIXFIRSTROWATT

- OSParseosil.tab.hpp, [1135](#), [1145](#), [1154](#)
- OSParseosol.tab.hpp, [1210](#), [1219](#), [1229](#)
- OSParseosrl.tab.hpp, [1282](#), [1292](#), [1302](#)

TARGETOBJECTIVEIDXATT

- OSParseosil.tab.hpp, [1149](#)
- OSParseosol.tab.hpp, [1224](#)
- OSParseosrl.tab.hpp, [1297](#)

TARGETOBJECTIVENAMEATT

- OSParseosil.tab.hpp, [1150](#)
- OSParseosol.tab.hpp, [1224](#)
- OSParseosrl.tab.hpp, [1297](#)

TEMPLATEMATRIXIDXATT

- OSParseosil.tab.hpp, [1133](#)
- OSParseosol.tab.hpp, [1207](#)
- OSParseosrl.tab.hpp, [1280](#)

TIMEDOMAINEND

- OSParseosil.tab.hpp, [1133](#)
- OSParseosol.tab.hpp, [1207](#)
- OSParseosrl.tab.hpp, [1280](#)

TIMEDOMAINSTART

- OSParseosil.tab.hpp, [1133](#)
- OSParseosol.tab.hpp, [1207](#)
- OSParseosrl.tab.hpp, [1280](#)

TIMEEND

- OSParseosil.tab.hpp, [1152](#)
- OSParseosol.tab.hpp, [1227](#)
- OSParseosrl.tab.hpp, [1300](#)

- TIMESEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- TIMESERVICESTARTEDEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMESERVICESTARTEDSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMESSTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1157](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1295](#), [1304](#)
- TIMESTAMPEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMESTAMPSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMESTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMINGINFORMATIONEND
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TIMINGINFORMATIONSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TOATT
 - OSParseosil.tab.hpp, [1139](#)
 - OSParseosol.tab.hpp, [1214](#)
 - OSParseosrl.tab.hpp, [1286](#)
- TOTALJOBSSOFAREND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TOTALJOBSSOFARSTART
 - OSParseosil.tab.hpp, [1152](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- TRANSFORMATIONEND
 - OSParseosil.tab.hpp, [1136](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- TRANSFORMATIONSTART
 - OSParseosil.tab.hpp, [1136](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1303](#)
- TRANSPORTTYPEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- TWOQUOTES
 - OSParseosil.tab.hpp, [1131](#), [1138](#), [1148](#)
 - OSParseosol.tab.hpp, [1205](#), [1213](#), [1223](#)
 - OSParseosrl.tab.hpp, [1278](#), [1286](#), [1296](#)
- TYPEATT
 - OSParseosil.tab.hpp, [1134](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1209](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1282](#), [1287](#), [1297](#)
- TEMPLATEMATRIXIDXATT
 - OSParseosil.tab.hpp, [1112](#)
- TIMEDOMAINEND
 - OSParseosil.tab.hpp, [1113](#)
- TIMEDOMAINSTART
 - OSParseosil.tab.hpp, [1113](#)
- TIMEEND
 - OSParseosrl.tab.hpp, [1262](#)
- TIMESEND
 - OSParseosil.tab.hpp, [1127](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1274](#)
- TIMESSTART
 - OSParseosil.tab.hpp, [1127](#)
 - OSParseosol.tab.hpp, [1201](#)
 - OSParseosrl.tab.hpp, [1274](#)
- TIMESTAMPEND
 - OSParseosrl.tab.hpp, [1262](#)
- TIMESTAMPSTART
 - OSParseosrl.tab.hpp, [1262](#)
- TIMESTART
 - OSParseosrl.tab.hpp, [1262](#)
- TIMINGINFORMATIONEND
 - OSParseosrl.tab.hpp, [1262](#)
- TOATT
 - OSParseosol.tab.hpp, [1177](#)
- TOTALJOBSSOFAREND
 - OSParseosrl.tab.hpp, [1262](#)
- TOTALJOBSSOFARSTART
 - OSParseosrl.tab.hpp, [1262](#)
- TRANSFORMATIONEND
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1269](#)
- TRANSFORMATIONSTART
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1197](#)
 - OSParseosrl.tab.hpp, [1269](#)
- TRANSPORTTYPEATT
 - OSParseosol.tab.hpp, [1180](#)

TWOQUOTES

OSParseosil.tab.hpp, [1107](#)
 OSParseosol.tab.hpp, [1176](#)
 OSParseosrl.tab.hpp, [1250](#)

TYPEATT

OSParseosil.tab.hpp, [1118](#)
 OSParseosol.tab.hpp, [1177](#)
 OSParseosrl.tab.hpp, [1252](#)

takeOverOSInstance

OSgams2osil, [405](#)

targetMatrixFirstCol

BaseMatrix, [36](#)
 OSgLPParserData, [412](#)

targetMatrixFirstColPresent

OSgLPParserData, [413](#)

targetMatrixFirstRow

BaseMatrix, [36](#)
 OSgLPParserData, [412](#)

targetMatrixFirstRowPresent

OSgLPParserData, [413](#)

targetObjectiveIdx

OptimizationSolution, [382](#)

targetObjectiveName

OptimizationSolution, [382](#)

tempC

OSgLPParserData, [411](#)

templnt

OSnLPParserData, [648](#)
 OSoLPParserData, [670](#)
 OSrLPParserData, [820](#)

tempStr

OSnLPParserData, [648](#)
 OSoLPParserData, [671](#)
 OSrLPParserData, [820](#)

tempVal

OSiLPParserData, [430](#)
 OSnLPParserData, [648](#)
 OSoLPParserData, [671](#)
 OSrLPParserData, [820](#)

templateMatrixIdx

MatrixCon, [285](#)
 MatrixObj, [308](#)
 MatrixVar, [325](#)
 OSiLPParserData, [430](#)

templateMatrixIdxPresent

OSiLPParserData, [429](#)

time

TimingInformation, [1008](#)

timeCategory

OSrLPParserData, [817](#)

timeDescription

OSrLPParserData, [818](#)

TimeDomain, [987](#)

~TimeDomain, [988](#)

interval, [988](#)

stages, [988](#)

TimeDomain, [988](#)

TimeDomain, [988](#)

timeDomain

InstanceData, [216](#)

TimeDomainInterval, [988](#)

~TimeDomainInterval, [989](#)

horizon, [989](#)

start, [989](#)

TimeDomainInterval, [989](#)

TimeDomainInterval, [989](#)

timeDomainInterval

OSiLPParserData, [424](#)

TimeDomainStage, [989](#)

~TimeDomainStage, [990](#)

constraints, [991](#)

name, [991](#)

objectives, [991](#)

TimeDomainStage, [990](#)

TimeDomainStage, [990](#)

variables, [991](#)

TimeDomainStageCon, [991](#)

~TimeDomainStageCon, [992](#)

idx, [992](#)

TimeDomainStageCon, [992](#)

TimeDomainStageCon, [992](#)

TimeDomainStageConstraints, [992](#)

~TimeDomainStageConstraints, [993](#)

con, [993](#)

numberOfConstraints, [993](#)

startIdx, [993](#)

TimeDomainStageConstraints, [993](#)

TimeDomainStageConstraints, [993](#)

TimeDomainStageObj, [993](#)

~TimeDomainStageObj, [994](#)

idx, [994](#)

TimeDomainStageObj, [994](#)

TimeDomainStageObj, [994](#)

TimeDomainStageObjectives, [994](#)

~TimeDomainStageObjectives, [995](#)

numberOfObjectives, [996](#)

obj, [996](#)

startIdx, [996](#)

TimeDomainStageObjectives, [995](#)

TimeDomainStageObjectives, [995](#)

TimeDomainStageVar, [997](#)

~TimeDomainStageVar, [998](#)

idx, [998](#)

TimeDomainStageVar, [998](#)

TimeDomainStageVar, [998](#)

TimeDomainStageVariables, [998](#)

~TimeDomainStageVariables, [999](#)

numberOfVariables, [999](#)

- startIdx, 999
- TimeDomainStageVariables, 999
- TimeDomainStageVariables, 999
- var, 999
- TimeDomainStages, 996
 - ~TimeDomainStages, 997
 - numberOfStages, 997
 - stage, 997
 - TimeDomainStages, 997
 - TimeDomainStages, 997
- timeDomainStages
 - OSILParserData, 424
- TimeMeasurement, 1000
 - ~TimeMeasurement, 1001
 - category, 1002
 - description, 1002
 - IsEqual, 1001
 - setRandom, 1001
 - TimeMeasurement, 1001
 - TimeMeasurement, 1001
 - type, 1002
- timeServiceStarted
 - ServiceResult, 953
- timeServiceStartedPresent
 - OSrLParseData, 825
- TimeSpan, 1002
 - ~TimeSpan, 1004
 - deepCopyFrom, 1004
 - IsEqual, 1004
 - setRandom, 1004
 - TimeSpan, 1004
 - TimeSpan, 1004
 - unit, 1005
 - value, 1005
- timeStamp
 - GeneralResult, 157
- timeType
 - OSrLParseData, 817
- timeUnit
 - OSrLParseData, 817
- timeValue
 - OSrLParseData, 817
- TimingInformation, 1005
 - ~TimingInformation, 1007
 - IsEqual, 1007
 - numberOfTimes, 1008
 - setRandom, 1007
 - time, 1008
 - TimingInformation, 1007
 - TimingInformation, 1007
- timingInformation
 - JobResult, 247
- tminlp
 - BonminSolver, 49
 - CouenneSolver, 100
- tmpOtherDescription
 - OSrLParseData, 818
- tmpOtherName
 - OSrLParseData, 818
- tmpOtherValue
 - OSrLParseData, 818
- tmpnlcount
 - OSnLParseData, 649
- to
 - PathPair, 905
- toPaths
 - OSoLParseData, 669
- totalJobsSoFar
 - ServiceResult, 953
- transformation
 - MatrixTransformation, 318
- transportType
 - ContactOption, 93
- transportTypeattON
 - OSoLParseData, 659
- type
 - GeneralStatus, 162
 - MatrixType, 323
 - OptimizationSolutionStatus, 385
 - OptimizationSolutionSubstatus, 387
 - OS_AMPL_SUFFIX, 391
 - OSgLParseData, 412
 - OSnLNodeNumber, 611
 - OtherConstraintOption, 849
 - OtherConstraintResult, 853
 - OtherObjectiveOption, 858
 - OtherObjectiveResult, 861
 - OtherVariableOption, 892
 - OtherVariableResult, 895
 - ServiceOption, 950
 - SolverOption, 956
 - TimeMeasurement, 1002
 - Variable, 1009
- typeAttribute
 - OSnLParseData, 646
 - OSoLParseData, 667
 - OSrLParseData, 823
- typeAttributePresent
 - OSnLParseData, 646
 - OSoLParseData, 667
 - OSrLParseData, 822
- typePresent
 - OSgLParseData, 412
- UBCONEIDXATT
 - OSParseosil.tab.hpp, 1132
 - OSParseosol.tab.hpp, 1207
 - OSParseosrl.tab.hpp, 1280

- UBDUALVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- UBMATRIXIDXATT
 - OSParseosil.tab.hpp, [1132](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- UBVALUEATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1287](#)
- UNITATT
 - OSParseosil.tab.hpp, [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1287](#), [1297](#)
- UNKNOWNEND
 - OSParseosil.tab.hpp, [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1217](#), [1227](#)
 - OSParseosrl.tab.hpp, [1290](#), [1300](#)
- UNKNOWNSTART
 - OSParseosil.tab.hpp, [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1217](#), [1227](#)
 - OSParseosrl.tab.hpp, [1290](#), [1300](#)
- USEDPCUNUMBEREND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDPCUNUMBERSTART
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDPCUSPEEDEND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDPCUSPEEDSTART
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDDISKSPACEEND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDDISKSPACESTART
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1227](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDMEMORYEND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1228](#)
 - OSParseosrl.tab.hpp, [1300](#)
- USEDMEMORYSTART
 - OSParseosil.tab.hpp, [1153](#)
- OSParseosol.tab.hpp, [1228](#)
- OSParseosrl.tab.hpp, [1300](#)
- USERNAMEEND
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- USERNAMESTART
 - OSParseosil.tab.hpp, [1141](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- UBCONEIDXATT
 - OSParseosil.tab.hpp, [1112](#)
- UBDUALVALUEATT
 - OSParseosol.tab.hpp, [1180](#)
- UBMATRIXIDXATT
 - OSParseosil.tab.hpp, [1112](#)
- UBVALUEATT
 - OSParseosol.tab.hpp, [1180](#)
- UNITATT
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1253](#)
- UNKNOWNEND
 - OSParseosol.tab.hpp, [1188](#)
 - OSParseosrl.tab.hpp, [1262](#)
- UNKNOWNSTART
 - OSParseosol.tab.hpp, [1187](#)
 - OSParseosrl.tab.hpp, [1262](#)
- USEDPCUNUMBEREND
 - OSParseosrl.tab.hpp, [1263](#)
- USEDPCUNUMBERSTART
 - OSParseosrl.tab.hpp, [1262](#)
- USEDPCUSPEEDEND
 - OSParseosrl.tab.hpp, [1263](#)
- USEDPCUSPEEDSTART
 - OSParseosrl.tab.hpp, [1263](#)
- USEDDISKSPACEEND
 - OSParseosrl.tab.hpp, [1263](#)
- USEDDISKSPACESTART
 - OSParseosrl.tab.hpp, [1263](#)
- USEDMEMORYEND
 - OSParseosrl.tab.hpp, [1263](#)
- USEDMEMORYSTART
 - OSParseosrl.tab.hpp, [1263](#)
- USERNAMEEND
 - OSParseosol.tab.hpp, [1182](#)
- USERNAMESTART
 - OSParseosol.tab.hpp, [1182](#)
- ub
 - Constraint, [82](#)
 - OrthantCone, [390](#)
 - Variable, [1009](#)
- ubConeldx
 - MatrixCon, [286](#)
 - MatrixVar, [326](#)

- OSILParserData, [429](#)
- ubConelIdxPresent
 - OSILParserData, [429](#)
- ubDualValue
 - InitDualVarValue, [182](#)
 - OSoLParserData, [666](#)
- ubMatrixIdx
 - MatrixCon, [286](#)
 - MatrixVar, [326](#)
 - OSILParserData, [429](#)
- ubMatrixIdxPresent
 - OSILParserData, [428](#)
- ubValArray
 - OSoLParserData, [670](#)
- ubValAttributePresent
 - OSoLParserData, [663](#)
- ubValue
 - InitObjBound, [188](#)
 - OtherConOption, [842](#)
 - OtherObjOption, [864](#)
 - OtherVarOption, [900](#)
- ubValueAttribute
 - OSnLParserData, [647](#)
 - OSoLParserData, [669](#)
- ubValueAttributePresent
 - OSnLParserData, [647](#)
 - OSoLParserData, [668](#)
- ubValueString
 - OSoLParserData, [670](#)
- unit
 - CPU Speed, [105](#)
 - MaxTime, [330](#)
 - MinCPUSpeed, [334](#)
 - MinDiskSpace, [336](#)
 - MinMemorySize, [338](#)
 - StorageCapacity, [981](#)
 - TimeSpan, [1005](#)
- unitAttribute
 - OSnLParserData, [648](#)
 - OSoLParserData, [669](#)
 - OSrLParserData, [823](#)
- unitAttributePresent
 - OSnLParserData, [648](#)
 - OSoLParserData, [669](#)
 - OSrLParserData, [822](#)
- unknown
 - BasisStatus, [41](#)
- usedCPUNumber
 - JobResult, [248](#)
- usedCPUSpeed
 - JobResult, [248](#)
- usedDiskSpace
 - JobResult, [247](#)
- usedMemory

- JobResult, [247](#)
- userName
 - GeneralOption, [153](#)
- usernamePresent
 - OSoLParserData, [658](#)
- VALUEATT
 - OSParseosil.tab.hpp, [1131](#), [1139](#), [1149](#)
 - OSParseosol.tab.hpp, [1206](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1278](#), [1287](#), [1297](#)
- VALUESEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- VALUESSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- VALUESSTRINGEND
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1228](#)
 - OSParseosrl.tab.hpp, [1300](#)
- VALUESSTRINGSTART
 - OSParseosil.tab.hpp, [1153](#)
 - OSParseosol.tab.hpp, [1228](#)
 - OSParseosrl.tab.hpp, [1300](#)
- VALUETYPEATT
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1230](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- VAREND
 - OSParseosil.tab.hpp, [1133](#), [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1217](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1290](#), [1301](#)
- VARIABLEEND
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1222](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- VARIABLESEND
 - OSParseosil.tab.hpp, [1133](#), [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1217](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1290](#), [1301](#)
- VARIABLESSTART
 - OSParseosil.tab.hpp, [1133](#), [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1217](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1290](#), [1301](#)
- VARIABLESTART
 - OSParseosil.tab.hpp, [1137](#), [1147](#), [1156](#)
 - OSParseosol.tab.hpp, [1212](#), [1221](#), [1231](#)
 - OSParseosrl.tab.hpp, [1285](#), [1294](#), [1304](#)
- VARIDXEND
 - OSParseosil.tab.hpp, [1136](#), [1145](#), [1153](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1228](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1301](#)

- VARIDXSTART
 - OSParseosil.tab.hpp, [1136](#), [1145](#), [1153](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1228](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1301](#)
- VARREFERENCEELEMENTSEND
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- VARREFERENCEELEMENTSSTART
 - OSParseosil.tab.hpp, [1135](#), [1145](#), [1155](#)
 - OSParseosol.tab.hpp, [1210](#), [1220](#), [1229](#)
 - OSParseosrl.tab.hpp, [1283](#), [1293](#), [1302](#)
- VARREFERENCEMATRIXXATT
 - OSParseosil.tab.hpp, [1133](#)
 - OSParseosol.tab.hpp, [1207](#)
 - OSParseosrl.tab.hpp, [1280](#)
- VARSTART
 - OSParseosil.tab.hpp, [1133](#), [1142](#), [1153](#)
 - OSParseosol.tab.hpp, [1208](#), [1217](#), [1228](#)
 - OSParseosrl.tab.hpp, [1281](#), [1290](#), [1300](#)
- VARTYPEATT
 - OSParseosil.tab.hpp, [1132](#), [1140](#), [1149](#)
 - OSParseosol.tab.hpp, [1207](#), [1214](#), [1224](#)
 - OSParseosrl.tab.hpp, [1280](#), [1287](#), [1297](#)
- VALUEATT
 - OSParseosil.tab.hpp, [1107](#)
 - OSParseosol.tab.hpp, [1178](#)
 - OSParseosrl.tab.hpp, [1253](#)
- VALUESEND
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- VALUESSTART
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1195](#)
 - OSParseosrl.tab.hpp, [1268](#)
- VALUESSTRINGEND
 - OSParseosrl.tab.hpp, [1263](#)
- VALUESSTRINGSTART
 - OSParseosrl.tab.hpp, [1263](#)
- VALUETYPEATT
 - OSParseosil.tab.hpp, [1121](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1269](#)
- VAREND
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1186](#)
 - OSParseosrl.tab.hpp, [1263](#)
- VARIABLEEND
 - OSParseosil.tab.hpp, [1126](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- VARIABLESEND
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1186](#)
 - OSParseosrl.tab.hpp, [1263](#)
- VARIABLESSTART
 - OSParseosil.tab.hpp, [1113](#)
 - OSParseosol.tab.hpp, [1186](#)
 - OSParseosrl.tab.hpp, [1263](#)
- VARIABLESTART
 - OSParseosil.tab.hpp, [1125](#)
 - OSParseosol.tab.hpp, [1200](#)
 - OSParseosrl.tab.hpp, [1273](#)
- VARIDXEND
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1264](#)
- VARIDXSTART
 - OSParseosil.tab.hpp, [1122](#)
 - OSParseosol.tab.hpp, [1196](#)
 - OSParseosrl.tab.hpp, [1263](#)
- VARSTART
 - OSParseosil.tab.hpp, [1114](#)
 - OSParseosol.tab.hpp, [1186](#)
 - OSParseosrl.tab.hpp, [1263](#)
- VARTYPEATT
 - OSParseosil.tab.hpp, [1111](#)
 - OSParseosol.tab.hpp, [1179](#)
 - OSParseosrl.tab.hpp, [1253](#)
- vType
 - GeneralSparseMatrix, [159](#)
- valArray
 - OSoLParserData, [670](#)
- valAttributePresent
 - OSoLParserData, [663](#)
- value
 - BranchingWeight, [52](#)
 - ConReferenceMatrixElement, [69](#)
 - ContactOption, [93](#)
 - CPUNumber, [103](#)
 - CPUSpeed, [105](#)
 - DualVarValue, [127](#)
 - IndexStringPair, [166](#)
 - IndexValuePair, [166](#)
 - InitBasStatus, [169](#)
 - InitConValue, [175](#)
 - InitObjValue, [199](#)
 - InitVarValue, [210](#)
 - InitVarValueString, [213](#)
 - InstanceLocationOption, [218](#)
 - LinearConstraintCoefficients, [261](#)
 - MaxTime, [330](#)
 - MinCPUNumber, [332](#)
 - MinCPUSpeed, [334](#)
 - MinDiskSpace, [336](#)
 - MinMemorySize, [339](#)
 - ObjCoef, [351](#)

- ObjValue, 373
- OSnLNodeNumber, 611
- OtherConOption, 842
- OtherConResult, 845
- OtherConstraintOption, 849
- OtherConstraintResult, 853
- OtherObjectiveOption, 857
- OtherObjectiveResult, 861
- OtherObjOption, 864
- OtherObjResult, 867
- OtherOption, 869
- OtherOptionEnumeration, 872
- OtherResult, 878
- OtherSolutionResult, 883
- OtherVariableOption, 892
- OtherVariableResult, 895
- OtherVariableResultStruct, 897
- OtherVarOption, 900
- OtherVarResult, 903
- SolverOption, 956
- StorageCapacity, 981
- TimeSpan, 1005
- VarValue, 1032
- VarValueString, 1034
- valueAttribute
 - OSnLParserData, 647
 - OSoLParserData, 668
 - OSrLParserData, 823
- valueAttributePresent
 - OSnLParserData, 647
 - OSoLParserData, 668
 - OSrLParserData, 822
- valueSize
 - GeneralSparseMatrix, 159
 - SparseJacobianMatrix, 975
 - SparseMatrix, 977
- valueString
 - OSoLParserData, 669
- valueType
 - ConReferenceMatrixElement, 69
 - OSgLParserData, 415
- valueTypePresent
 - OSgLParserData, 415
- values
 - ConReferenceMatrixElements, 72
 - ConstantMatrixElements, 78
 - GeneralMatrixElements, 146
 - GeneralSparseMatrix, 159
 - LinearMatrixElements, 267
 - ObjectiveSolution, 362
 - ObjReferenceMatrixElements, 369
 - RowReferenceMatrixElements, 939
 - SparseIntVector, 973
 - SparseJacobianMatrix, 975
 - SparseMatrix, 977
 - SparseVector, 979
 - VariableSolution, 1018
 - VarReferenceMatrixElements, 1027
- valuesString
 - VariableSolution, 1018
- var
 - InitialBasisStatus, 185
 - InitVariableValues, 203
 - InitVariableValuesString, 207
 - IntegerVariableBranchingWeights, 222
 - OtherVariableOption, 892
 - OtherVariableResult, 895
 - SOSWeights, 970
 - TimeDomainStageVariables, 999
 - Variables, 1015
 - VariableValues, 1021
 - VariableValuesString, 1023
- varIdx
 - LinearMatrixElement, 263
- varOneIndexes
 - QuadraticTerms, 931
- VarReferenceMatrixElements, 1023
 - ~VarReferenceMatrixElements, 1025
 - alignsOnBlockBoundary, 1026
 - cloneMatrixNode, 1026
 - deepCopyFrom, 1027
 - getMatrixNodeInXML, 1026
 - getMatrixType, 1025
 - getNodeName, 1026
 - getNodeName, 1026
 - getNodeName, 1026
 - isEqual, 1026
 - setRandom, 1026
 - values, 1027
 - VarReferenceMatrixElements, 1025
 - VarReferenceMatrixElements, 1025
- varReferenceMatrixIdx
 - MatrixVar, 325
 - OSiLParserData, 430
- varReferenceMatrixIdxPresent
 - OSiLParserData, 429
- VarReferenceMatrixValues, 1027
 - ~VarReferenceMatrixValues, 1029
 - deepCopyFrom, 1029
 - el, 1029
 - isEqual, 1029
 - setRandom, 1029
 - VarReferenceMatrixValues, 1029
 - VarReferenceMatrixValues, 1029
- varTwoIndexes
 - QuadraticTerms, 931
- varType
 - MatrixVar, 326
 - OSiLParserData, 430

- OSMatlab, 487
- OtherVariableOption, 892
- OtherVariableResult, 895
- varTypeAttribute
 - OSnLPParserData, 646
 - OSoLPParserData, 668
 - OSrLPParserData, 823
- varTypeAttributePresent
 - OSnLPParserData, 646
 - OSoLPParserData, 667
 - OSrLPParserData, 822
- varTypePresent
 - OSiLPParserData, 429
- VarValue, 1030
 - ~VarValue, 1031
 - idx, 1031
 - IsEqual, 1031
 - name, 1032
 - setRandom, 1031
 - value, 1032
 - VarValue, 1031
 - VarValue, 1031
- VarValueString, 1032
 - ~VarValueString, 1033
 - idx, 1034
 - IsEqual, 1033
 - name, 1034
 - setRandom, 1033
 - value, 1034
 - VarValueString, 1033
 - VarValueString, 1033
- Variable, 1008
 - ~Variable, 1009
 - IsEqual, 1009
 - lb, 1009
 - name, 1010
 - type, 1009
 - ub, 1009
 - Variable, 1009
- VariableOption, 1010
 - ~VariableOption, 1011
 - addOther, 1012
 - deepCopyFrom, 1012
 - initialBasisStatus, 1013
 - initialVariableValues, 1013
 - initialVariableValuesString, 1013
 - integerVariableBranchingWeights, 1013
 - IsEqual, 1012
 - numberOfOtherVariableOptions, 1012
 - other, 1013
 - setOther, 1012
 - setRandom, 1012
 - sosVariableBranchingWeights, 1013
 - VariableOption, 1011
- VariableOption, 1011
- VariableSolution, 1015
 - ~VariableSolution, 1017
 - basisStatus, 1018
 - IsEqual, 1017
 - numberOfOtherVariableResults, 1017
 - other, 1018
 - setRandom, 1017
 - values, 1018
 - valuesString, 1018
 - VariableSolution, 1017
 - VariableSolution, 1017
- VariableValues, 1018
 - ~VariableValues, 1020
 - IsEqual, 1020
 - numberOfVar, 1020
 - setRandom, 1020
 - var, 1021
 - VariableValues, 1020
 - VariableValues, 1020
- VariableValuesString, 1021
 - ~VariableValuesString, 1022
 - IsEqual, 1022
 - numberOfVar, 1023
 - setRandom, 1022
 - var, 1023
 - VariableValuesString, 1022
 - VariableValuesString, 1022
- variablecoefattON
 - OSnLPParserData, 650
- variableidxattON
 - OSnLPParserData, 650
- Variables, 1013
 - ~Variables, 1015
 - IsEqual, 1015
 - numberOfVariables, 1015
 - var, 1015
 - Variables, 1015
- variables
 - InstanceData, 215
 - OptimizationOption, 376
 - OptimizationSolution, 382
 - TimeDomainStage, 991
- variablesPresent
 - OSoLPParserData, 663
- verifyBasisStatus
 - OSParameters.h, 1347
- verifyCPUSpeedUnit
 - OSParameters.h, 1345
- verifyConReferenceValueType
 - OSParameters.h, 1348
- verifyConeType
 - OSParameters.h, 1349
- verifyForm

- CsdpSolver, [109](#)
- verifyGeneralResultStatus
 - OSParameters.h, [1346](#)
- verifyJobStatus
 - OSParameters.h, [1347](#)
- verifyLocationType
 - OSParameters.h, [1346](#)
- verifyMatrixConstructorType
 - OSParameters.h, [1348](#)
- verifyMatrixSymmetry
 - OSParameters.h, [1348](#)
- verifyMatrixType
 - OSParameters.h, [1348](#)
- verifyNIEExprShape
 - OSParameters.h, [1349](#)
- verifyServiceType
 - OSParameters.h, [1346](#)
- verifySolutionStatus
 - OSParameters.h, [1347](#)
- verifySolutionSubstatusType
 - OSParameters.h, [1347](#)
- verifyStorageUnit
 - OSParameters.h, [1345](#)
- verifySystemCurrentState
 - OSParameters.h, [1347](#)
- verifyTimeCategory
 - OSParameters.h, [1346](#)
- verifyTimeType
 - OSParameters.h, [1346](#)
- verifyTimeUnit
 - OSParameters.h, [1346](#)
- verifyTransportType
 - OSParameters.h, [1346](#)
- verifyVarType
 - OSParameters.h, [1347](#)
- vl
 - OSMatlab, [487](#)
- vu
 - OSMatlab, [487](#)
- WEIGHTATT
 - OSParseosil.tab.hpp, [1140](#)
 - OSParseosol.tab.hpp, [1215](#)
 - OSParseosrl.tab.hpp, [1288](#)
- WEIGHTEDOBJECTIVESATT
 - OSParseosil.tab.hpp, [1150](#)
 - OSParseosol.tab.hpp, [1224](#)
 - OSParseosrl.tab.hpp, [1297](#)
- WEIGHTATT
 - OSParseosol.tab.hpp, [1180](#)
- WSUtil, [1034](#)
 - ~WSUtil, [1035](#)
 - createFormDataUpload, [1037](#)
 - createSOAPMessage, [1036](#)
 - deSOAPify, [1036](#)
 - getOSxL, [1037](#)
 - SOAPify, [1036](#)
 - sendSOAPMessage, [1035](#)
 - WSUtil, [1035](#)
 - WSUtil, [1035](#)
- walkTree
 - OSnl2OS, [503](#)
- weight
 - Objective, [353](#)
- weightedObjAttributePresent
 - OSrLParserData, [822](#)
- weightedObjectives
 - OptimizationSolution, [382](#)
- writeBasisStatus
 - OSgLWriter.h, [1054](#)
- writeDblVectorData
 - OSgLWriter.h, [1054](#)
- writeFileFromChar
 - FileUtil, [139](#)
- writeFileFromString
 - FileUtil, [139](#)
- writeGeneralFileHeader
 - OSgLWriter.h, [1054](#)
- writeIntVectorData
 - OSgLWriter.h, [1053](#)
- writeOSiL
 - OSiLWriter, [433](#)
- writeOSoL
 - OSoLWriter, [674](#)
- writeOSrL
 - OSrLWriter, [830](#)
- writeOtherOptionEnumeration
 - OSgLWriter.h, [1054](#)
- writeResult
 - BonminSolver, [49](#)
 - CoinSolver, [56](#), [57](#)
 - CouenneSolver, [99](#)
- writeSolFile
 - OSosrl2ampl, [721](#)
- writeSolution
 - OSrL2Gams, [812](#)
- writeStringData
 - OSStringUtil.h, [1354](#)
- writeVersion
 - OSCommandLine, [399](#)
 - osOptionsStruc, [719](#)
- YYLTYPE, [1037](#)
 - first_column, [1038](#)
 - first_line, [1038](#)
 - last_column, [1038](#)
 - last_line, [1038](#)
 - OSParseosil.tab.hpp, [1130](#)

- OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- YYLTYPE_IS_TRIVIAL
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- YYSTYPE, [1038](#)
 - dval, [1038](#)
 - ival, [1038](#)
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
 - sval, [1038](#)
- YYSTYPE_IS_TRIVIAL
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1277](#)
- yylytype
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- yystate
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)
- yyltokentype
 - OSParseosil.tab.hpp, [1130](#)
 - OSParseosol.tab.hpp, [1205](#)
 - OSParseosrl.tab.hpp, [1278](#)