

OS

2.10

Generated by Doxygen 1.8.9.1

Thu Oct 8 2015 23:12:15

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	27
2.1	Class List	27
3	File Index	39
3.1	File List	39
4	Class Documentation	41
4.1	Base64 Class Reference	41
4.1.1	Detailed Description	41
4.1.2	Member Function Documentation	42
4.1.2.1	encodeb64	42
4.1.2.2	decodeb64	42
4.2	BaseMatrix Class Reference	42
4.2.1	Detailed Description	43
4.2.2	Member Function Documentation	43
4.2.2.1	getNodeTypes	43
4.2.2.2	getNodeName	44
4.2.2.3	getMatrixType	44
4.2.2.4	getMatrixNodeInXML	44
4.2.2.5	alignsOnBlockBoundary	44
4.2.2.6	cloneMatrixNode	44
4.3	BasisStatus Class Reference	45
4.3.1	Detailed Description	45
4.3.2	Member Function Documentation	45
4.3.2.1	setRandom	45
4.3.2.2	deepCopyFrom	46

4.3.2.3	setIntVector	46
4.3.2.4	addIdx	46
4.3.2.5	getNumberOfEI	46
4.3.2.6	getEI	47
4.3.2.7	getIntVector	48
4.3.2.8	getBasisDense	48
4.4	BonminProblem Class Reference	49
4.4.1	Detailed Description	50
4.4.2	Member Function Documentation	50
4.4.2.1	get_variables_types	50
4.4.2.2	get_variables_linearity	50
4.4.2.3	get_constraints_linearity	50
4.4.2.4	get_nlp_info	50
4.4.2.5	get_bounds_info	51
4.4.2.6	finalize_solution	51
4.5	BonminSolver Class Reference	51
4.5.1	Detailed Description	52
4.5.2	Member Function Documentation	52
4.5.2.1	setSolverOptions	52
4.6	BranchingWeight Class Reference	52
4.6.1	Detailed Description	53
4.6.2	Member Function Documentation	53
4.6.2.1	setRandom	53
4.6.2.2	deepCopyFrom	54
4.7	CoinSolver Class Reference	54
4.7.1	Detailed Description	55
4.7.2	Member Function Documentation	55
4.7.2.1	solve	55
4.7.2.2	buildSolverInstance	55
4.7.2.3	setSolverOptions	55
4.7.2.4	setCoinPackedMatrix	56
4.7.2.5	getCoinSolverType	56
4.7.2.6	dataEchoCheck	56
4.8	CompletelyPositiveMatricesCone Class Reference	56
4.8.1	Detailed Description	57
4.8.2	Member Function Documentation	57
4.8.2.1	getConeName	57

4.8.2.2	getConeInXML	57
4.8.2.3	setRandom	57
4.8.2.4	deepCopyFrom	58
4.9	Cone Class Reference	58
4.9.1	Detailed Description	59
4.9.2	Member Function Documentation	59
4.9.2.1	getConeName	59
4.9.2.2	getConeInXML	59
4.9.2.3	setRandom	59
4.9.2.4	deepCopyFrom	60
4.9.3	Member Data Documentation	60
4.9.3.1	numberOfOtherIndexes	60
4.10	Cones Class Reference	60
4.10.1	Detailed Description	61
4.10.2	Member Function Documentation	61
4.10.2.1	setRandom	61
4.10.2.2	deepCopyFrom	61
4.11	ConReferenceMatrixElement Class Reference	62
4.11.1	Detailed Description	62
4.11.2	Member Function Documentation	62
4.11.2.1	setRandom	62
4.11.2.2	deepCopyFrom	63
4.11.3	Member Data Documentation	63
4.11.3.1	conReference	63
4.11.3.2	valueType	63
4.12	ConReferenceMatrixElements Class Reference	63
4.12.1	Detailed Description	64
4.12.2	Member Function Documentation	64
4.12.2.1	getNodeType	64
4.12.2.2	getMatrixType	64
4.12.2.3	getNodeName	65
4.12.2.4	getMatrixNodeInXML	65
4.12.2.5	alignsOnBlockBoundary	65
4.12.2.6	cloneMatrixNode	65
4.12.2.7	setRandom	65
4.12.2.8	deepCopyFrom	66
4.13	ConReferenceMatrixValues Class Reference	66

4.13.1 Detailed Description	66
4.13.2 Member Function Documentation	67
4.13.2.1 setRandom	67
4.13.2.2 deepCopyFrom	67
4.14 ConstantMatrixElements Class Reference	67
4.14.1 Detailed Description	68
4.14.2 Member Function Documentation	68
4.14.2.1 getNodeType	68
4.14.2.2 getNodeName	68
4.14.2.3 getMatrixType	68
4.14.2.4 getMatrixNodeInXML	68
4.14.2.5 alignsOnBlockBoundary	69
4.14.2.6 cloneMatrixNode	70
4.14.2.7 setRandom	70
4.14.2.8 deepCopyFrom	70
4.15 ConstantMatrixValues Class Reference	70
4.15.1 Detailed Description	71
4.15.2 Member Function Documentation	71
4.15.2.1 isEqual	71
4.15.2.2 setRandom	71
4.15.2.3 deepCopyFrom	72
4.16 Constraint Class Reference	72
4.16.1 Detailed Description	73
4.17 ConstraintOption Class Reference	73
4.17.1 Detailed Description	74
4.17.2 Member Function Documentation	74
4.17.2.1 setRandom	74
4.17.2.2 deepCopyFrom	74
4.17.2.3 setOther	74
4.17.2.4 addOther	75
4.18 Constraints Class Reference	75
4.18.1 Detailed Description	75
4.19 ConstraintSolution Class Reference	76
4.19.1 Detailed Description	76
4.19.2 Member Function Documentation	77
4.19.2.1 setRandom	77
4.20 ContactOption Class Reference	77

4.20.1 Detailed Description	77
4.20.2 Member Function Documentation	78
4.20.2.1 setRandom	78
4.20.2.2 deepCopyFrom	78
4.21 CopositiveMatricesCone Class Reference	78
4.21.1 Detailed Description	79
4.21.2 Member Function Documentation	79
4.21.2.1 getConeName	79
4.21.2.2 getConeInXML	79
4.21.2.3 setRandom	79
4.21.2.4 deepCopyFrom	80
4.22 CouenneSolver Class Reference	80
4.22.1 Detailed Description	81
4.22.2 Member Function Documentation	81
4.22.2.1 setSolverOptions	81
4.23 CPUNumber Class Reference	81
4.23.1 Detailed Description	82
4.23.2 Member Function Documentation	82
4.23.2.1 setRandom	82
4.23.2.2 deepCopyFrom	83
4.24 CPUSpeed Class Reference	83
4.24.1 Detailed Description	84
4.24.2 Member Function Documentation	84
4.24.2.1 setRandom	84
4.24.2.2 deepCopyFrom	84
4.25 CsdpSolver Class Reference	85
4.25.1 Detailed Description	85
4.25.2 Member Function Documentation	86
4.25.2.1 buildSolverInstance	86
4.25.2.2 setSolverOptions	86
4.26 DefaultSolver Class Reference	86
4.26.1 Detailed Description	87
4.26.2 Member Data Documentation	87
4.26.2.1 sSolverName	87
4.27 DirectoriesAndFiles Class Reference	88
4.27.1 Detailed Description	88
4.27.2 Member Function Documentation	89

4.27.2.1	setRandom	89
4.27.2.2	deepCopyFrom	89
4.27.2.3	setPath	89
4.27.2.4	addPath	89
4.28	DoubleVector Class Reference	89
4.28.1	Detailed Description	90
4.29	DualCone Class Reference	90
4.29.1	Detailed Description	91
4.29.2	Member Function Documentation	91
4.29.2.1	getConeName	91
4.29.2.2	setRandom	91
4.29.2.3	deepCopyFrom	91
4.29.3	Member Data Documentation	91
4.29.3.1	numberOfOtherIndexes	91
4.30	DualVariableValues Class Reference	92
4.30.1	Detailed Description	92
4.30.2	Member Function Documentation	93
4.30.2.1	setRandom	93
4.31	DualVarValue Class Reference	93
4.31.1	Detailed Description	93
4.31.2	Member Function Documentation	94
4.31.2.1	setRandom	94
4.32	ErrorClass Class Reference	94
4.32.1	Detailed Description	95
4.32.2	Constructor & Destructor Documentation	95
4.32.2.1	ErrorClass	95
4.33	ExpandedMatrixBlocks Class Reference	95
4.33.1	Detailed Description	96
4.33.2	Constructor & Destructor Documentation	96
4.33.2.1	ExpandedMatrixBlocks	96
4.33.3	Member Function Documentation	97
4.33.3.1	display	97
4.33.3.2	getBlock	97
4.33.3.3	isBlockDiagonal	97
4.33.4	Member Data Documentation	97
4.33.4.1	isRowMajor	97
4.33.4.2	blockRows	97

4.33.4.3	blockColumns	98
4.33.4.4	blocks	98
4.34	ExprNode Class Reference	98
4.34.1	Detailed Description	99
4.34.2	Member Function Documentation	99
4.34.2.1	getTokenNumber	99
4.34.2.2	getTokenName	100
4.34.2.3	getNonlinearExpressionInXML	100
4.34.2.4	getPrefixFromExpressionTree	100
4.34.2.5	preOrderOSnLNodeTraversal	100
4.34.2.6	getPostfixFromExpressionTree	101
4.34.2.7	postOrderOSnLNodeTraversal	101
4.34.2.8	cloneExprNode	101
4.34.3	Member Data Documentation	102
4.34.3.1	inodeType	102
4.35	FileUtil Class Reference	102
4.35.1	Detailed Description	102
4.35.2	Member Function Documentation	103
4.35.2.1	getFileAsString	103
4.35.2.2	getFileAsChar	103
4.35.2.3	writeFileFromString	103
4.35.2.4	writeFileFromChar	103
4.35.2.5	writeFileFromChar	104
4.36	GeneralFileHeader Class Reference	104
4.36.1	Detailed Description	105
4.36.2	Member Function Documentation	105
4.36.2.1	setRandom	105
4.36.2.2	deepCopyFrom	105
4.36.2.3	getHeaderItem	105
4.36.2.4	setHeader	106
4.37	GeneralMatrixElements Class Reference	106
4.37.1	Detailed Description	107
4.37.2	Member Function Documentation	107
4.37.2.1	getNodeType	107
4.37.2.2	getMatrixType	107
4.37.2.3	getNodeName	107
4.37.2.4	getMatrixNodeInXML	107

4.37.2.5	alignsOnBlockBoundary	107
4.37.2.6	cloneMatrixNode	108
4.37.2.7	setRandom	108
4.37.2.8	deepCopyFrom	108
4.38	GeneralMatrixValues Class Reference	108
4.38.1	Detailed Description	109
4.38.2	Member Function Documentation	109
4.38.2.1	setRandom	109
4.38.2.2	deepCopyFrom	109
4.39	GeneralOption Class Reference	110
4.39.1	Detailed Description	111
4.39.2	Member Function Documentation	111
4.39.2.1	setRandom	111
4.39.2.2	deepCopyFrom	111
4.40	GeneralResult Class Reference	112
4.40.1	Detailed Description	112
4.40.2	Member Function Documentation	113
4.40.2.1	setRandom	113
4.41	GeneralSparseMatrix Class Reference	113
4.41.1	Detailed Description	114
4.41.2	Constructor & Destructor Documentation	114
4.41.2.1	GeneralSparseMatrix	114
4.41.3	Member Function Documentation	115
4.41.3.1	display	115
4.41.3.2	isDiagonal	115
4.41.4	Member Data Documentation	115
4.41.4.1	isRowMajor	115
4.41.4.2	symmetry	115
4.41.4.3	index	115
4.41.4.4	vType	115
4.41.4.5	value	116
4.42	GeneralStatus Class Reference	116
4.42.1	Detailed Description	116
4.42.2	Member Function Documentation	117
4.42.2.1	setRandom	117
4.43	GeneralSubstatus Class Reference	117
4.43.1	Detailed Description	118

4.43.2	Member Function Documentation	118
4.43.2.1	setRandom	118
4.44	IndexStringPair Struct Reference	118
4.44.1	Detailed Description	119
4.45	IndexValuePair Struct Reference	119
4.45.1	Detailed Description	119
4.46	InitBasStatus Class Reference	119
4.46.1	Detailed Description	120
4.46.2	Member Function Documentation	120
4.46.2.1	setRandom	120
4.46.2.2	deepCopyFrom	121
4.47	InitConstraintValues Class Reference	121
4.47.1	Detailed Description	122
4.47.2	Member Function Documentation	122
4.47.2.1	setRandom	122
4.47.2.2	deepCopyFrom	122
4.47.2.3	setCon	123
4.47.2.4	setCon	123
4.47.2.5	setCon	123
4.47.2.6	addCon	123
4.47.2.7	addCon	123
4.48	InitConValue Class Reference	124
4.48.1	Detailed Description	124
4.48.2	Member Function Documentation	125
4.48.2.1	setRandom	125
4.48.2.2	deepCopyFrom	125
4.49	InitDualVariableValues Class Reference	125
4.49.1	Detailed Description	126
4.49.2	Member Function Documentation	127
4.49.2.1	setRandom	127
4.49.2.2	deepCopyFrom	127
4.49.2.3	setCon	127
4.49.2.4	setCon	127
4.49.2.5	setCon	127
4.49.2.6	addCon	128
4.49.2.7	addCon	128
4.50	InitDualVarValue Class Reference	128

4.50.1 Detailed Description	129
4.50.2 Member Function Documentation	129
4.50.2.1 setRandom	129
4.50.2.2 deepCopyFrom	129
4.51 InitialBasisStatus Class Reference	130
4.51.1 Detailed Description	131
4.51.2 Member Function Documentation	131
4.51.2.1 setRandom	131
4.51.2.2 deepCopyFrom	131
4.51.2.3 setVar	131
4.51.2.4 addVar	132
4.52 InitObjBound Class Reference	132
4.52.1 Detailed Description	133
4.52.2 Member Function Documentation	133
4.52.2.1 setRandom	133
4.52.2.2 deepCopyFrom	133
4.53 InitObjectiveBounds Class Reference	134
4.53.1 Detailed Description	134
4.53.2 Member Function Documentation	135
4.53.2.1 setRandom	135
4.53.2.2 deepCopyFrom	135
4.53.2.3 setObj	135
4.53.2.4 setObj	135
4.53.2.5 setObj	136
4.53.2.6 addObj	136
4.53.2.7 addObj	136
4.54 InitObjectiveValues Class Reference	136
4.54.1 Detailed Description	137
4.54.2 Member Function Documentation	138
4.54.2.1 setRandom	138
4.54.2.2 deepCopyFrom	139
4.54.2.3 setObj	139
4.54.2.4 setObj	139
4.54.2.5 setObj	139
4.54.2.6 addObj	139
4.54.2.7 addObj	140
4.55 InitObjValue Class Reference	140

4.55.1 Detailed Description	141
4.55.2 Member Function Documentation	141
4.55.2.1 setRandom	141
4.55.2.2 deepCopyFrom	141
4.56 InitVariableValues Class Reference	142
4.56.1 Detailed Description	142
4.56.2 Member Function Documentation	143
4.56.2.1 setRandom	143
4.56.2.2 deepCopyFrom	143
4.56.2.3 setVar	143
4.56.2.4 setVar	143
4.56.2.5 setVar	144
4.56.2.6 addVar	144
4.56.2.7 addVar	144
4.57 InitVariableValuesString Class Reference	144
4.57.1 Detailed Description	145
4.57.2 Member Function Documentation	146
4.57.2.1 setRandom	146
4.57.2.2 deepCopyFrom	147
4.57.2.3 setVar	147
4.57.2.4 setVar	147
4.57.2.5 setVar	147
4.57.2.6 addVar	148
4.57.2.7 addVar	148
4.58 InitVarValue Class Reference	148
4.58.1 Detailed Description	149
4.58.2 Member Function Documentation	149
4.58.2.1 setRandom	149
4.58.2.2 deepCopyFrom	149
4.59 InitVarValueString Class Reference	150
4.59.1 Detailed Description	150
4.59.2 Member Function Documentation	151
4.59.2.1 setRandom	151
4.59.2.2 deepCopyFrom	151
4.60 InstanceData Class Reference	151
4.60.1 Detailed Description	152
4.61 InstanceLocationOption Class Reference	153

4.61.1 Detailed Description	153
4.61.2 Member Function Documentation	154
4.61.2.1 setRandom	154
4.61.2.2 deepCopyFrom	154
4.62 IntegerVariableBranchingWeights Class Reference	154
4.62.1 Detailed Description	155
4.62.2 Member Function Documentation	155
4.62.2.1 setRandom	155
4.62.2.2 deepCopyFrom	156
4.62.2.3 setVar	156
4.62.2.4 setVar	156
4.62.2.5 setVar	156
4.62.2.6 addVar	157
4.62.2.7 addVar	158
4.63 IntersectionCone Class Reference	158
4.63.1 Detailed Description	159
4.63.2 Member Function Documentation	159
4.63.2.1 getConeName	159
4.63.2.2 getConeInXML	159
4.63.2.3 setRandom	159
4.63.2.4 deepCopyFrom	160
4.63.3 Member Data Documentation	161
4.63.3.1 numberOfOtherIndexes	161
4.64 Interval Class Reference	161
4.64.1 Detailed Description	161
4.65 IntVector Class Reference	161
4.65.1 Detailed Description	162
4.65.2 Member Function Documentation	162
4.65.2.1 setRandom	162
4.65.2.2 deepCopyFrom	162
4.65.2.3 setIntVector	163
4.65.2.4 extendIntVector	163
4.65.2.5 getEI	163
4.65.2.6 getEI	163
4.66 IpoptProblem Class Reference	163
4.66.1 Detailed Description	164
4.67 IpoptSolver Class Reference	165

4.67.1 Detailed Description	165
4.67.2 Member Function Documentation	166
4.67.2.1 buildSolverInstance	166
4.67.2.2 setSolverOptions	166
4.68 JobDependencies Class Reference	166
4.68.1 Detailed Description	167
4.68.2 Member Function Documentation	167
4.68.2.1 setRandom	167
4.68.2.2 deepCopyFrom	167
4.68.2.3 setJobID	168
4.68.2.4 addJobID	169
4.69 JobOption Class Reference	169
4.69.1 Detailed Description	170
4.69.2 Member Function Documentation	170
4.69.2.1 setRandom	170
4.69.2.2 deepCopyFrom	171
4.70 JobResult Class Reference	171
4.70.1 Detailed Description	172
4.70.2 Member Function Documentation	172
4.70.2.1 setRandom	172
4.70.3 Member Data Documentation	173
4.70.3.1 usedMemory	173
4.71 KnitroProblem Class Reference	173
4.71.1 Detailed Description	173
4.72 KnitroSolver Class Reference	173
4.72.1 Detailed Description	174
4.72.2 Member Function Documentation	174
4.72.2.1 setSolverOptions	174
4.73 LindoSolver Class Reference	175
4.73.1 Detailed Description	176
4.73.2 Member Function Documentation	176
4.73.2.1 setSolverOptions	176
4.73.2.2 optimize	176
4.73.2.3 processVariables	176
4.73.2.4 processConstraints	177
4.73.2.5 generateLindoModel	177
4.73.2.6 addSlackVars	177

4.73.2.7	processQuadraticTerms	177
4.73.2.8	processNonlinearExpressions	177
4.73.2.9	lindoAPIErrorCheck	177
4.74	LinearConstraintCoefficients Class Reference	178
4.74.1	Detailed Description	178
4.74.2	Member Data Documentation	179
4.74.2.1	iNumberOfStartElements	179
4.75	LinearMatrixElement Class Reference	179
4.75.1	Detailed Description	179
4.75.2	Member Function Documentation	179
4.75.2.1	setRandom	179
4.75.2.2	deepCopyFrom	180
4.76	LinearMatrixElements Class Reference	180
4.76.1	Detailed Description	181
4.76.2	Member Function Documentation	181
4.76.2.1	getNodeType	181
4.76.2.2	getMatrixType	181
4.76.2.3	getNodeName	181
4.76.2.4	getMatrixNodeInXML	181
4.76.2.5	alignsOnBlockBoundary	181
4.76.2.6	cloneMatrixNode	182
4.76.2.7	setRandom	182
4.76.2.8	deepCopyFrom	182
4.76.3	Member Data Documentation	182
4.76.3.1	value	182
4.77	LinearMatrixElementTerm Class Reference	183
4.77.1	Detailed Description	183
4.77.2	Member Function Documentation	183
4.77.2.1	setRandom	183
4.77.2.2	deepCopyFrom	183
4.78	LinearMatrixValues Class Reference	184
4.78.1	Detailed Description	184
4.78.2	Member Function Documentation	184
4.78.2.1	setRandom	184
4.78.2.2	deepCopyFrom	185
4.79	MathUtil Class Reference	185
4.79.1	Detailed Description	185

4.79.2	Member Function Documentation	186
4.79.2.1	convertLinearConstraintCoefficientMatrixToTheOtherMajor	186
4.79.2.2	format_os_dtoa	187
4.80	Matrices Class Reference	187
4.80.1	Detailed Description	188
4.80.2	Member Function Documentation	188
4.80.2.1	setRandom	188
4.80.2.2	deepCopyFrom	188
4.81	MatrixBlock Class Reference	188
4.81.1	Detailed Description	189
4.81.2	Member Function Documentation	189
4.81.2.1	getNodeTypes	189
4.81.2.2	getNodeName	189
4.81.2.3	getMatrixType	189
4.81.2.4	getMatrixNodeInXML	190
4.81.2.5	alignsOnBlockBoundary	190
4.81.2.6	expandElements	190
4.81.2.7	cloneMatrixNode	190
4.81.2.8	setRandom	191
4.81.2.9	deepCopyFrom	192
4.82	MatrixBlocks Class Reference	192
4.82.1	Detailed Description	193
4.82.2	Member Function Documentation	193
4.82.2.1	getNodeTypes	193
4.82.2.2	getNodeName	193
4.82.2.3	getMatrixType	193
4.82.2.4	getMatrixNodeInXML	193
4.82.2.5	alignsOnBlockBoundary	194
4.82.2.6	cloneMatrixNode	195
4.82.2.7	setRandom	195
4.82.2.8	deepCopyFrom	195
4.83	MatrixCon Class Reference	196
4.83.1	Detailed Description	196
4.84	MatrixConstraints Class Reference	197
4.84.1	Detailed Description	197
4.85	MatrixConstraintSolution Class Reference	197
4.85.1	Detailed Description	198

4.86	MatrixConstructor Class Reference	198
4.86.1	Detailed Description	198
4.87	MatrixElements Class Reference	199
4.87.1	Detailed Description	199
4.87.2	Member Function Documentation	199
4.87.2.1	IsEqual	199
4.88	MatrixElementValues Class Reference	200
4.88.1	Detailed Description	200
4.88.2	Member Function Documentation	200
4.88.2.1	deepCopyFrom	200
4.88.3	Member Data Documentation	201
4.88.3.1	numberOfEl	201
4.89	MatrixExpression Class Reference	201
4.89.1	Detailed Description	202
4.89.2	Member Data Documentation	202
4.89.2.1	shape	202
4.90	MatrixExpressions Class Reference	202
4.90.1	Detailed Description	202
4.91	MatrixExpressionTree Class Reference	203
4.91.1	Detailed Description	203
4.91.2	Member Function Documentation	203
4.91.2.1	getPrefixFromExpressionTree	203
4.91.2.2	getPostfixFromExpressionTree	204
4.92	MatrixNode Class Reference	204
4.92.1	Detailed Description	205
4.92.2	Member Function Documentation	205
4.92.2.1	getNodeType	205
4.92.2.2	getMatrixType	205
4.92.2.3	getNodeName	205
4.92.2.4	getMatrixNodeInXML	206
4.92.2.5	getPrefixFromNodeTree	206
4.92.2.6	getPostfixFromNodeTree	206
4.92.2.7	postOrderMatrixNodeTraversal	206
4.92.2.8	cloneMatrixNode	206
4.92.2.9	alignsOnBlockBoundary	207
4.92.2.10	setRandom	208
4.92.2.11	deepCopyFrom	208

4.93	MatrixObj Class Reference	208
4.93.1	Detailed Description	209
4.94	MatrixObjectives Class Reference	209
4.94.1	Detailed Description	210
4.95	MatrixObjectiveSolution Class Reference	210
4.95.1	Detailed Description	211
4.96	MatrixProgramming Class Reference	211
4.96.1	Detailed Description	211
4.96.2	Member Function Documentation	212
4.96.2.1	setRandom	212
4.96.2.2	deepCopyFrom	213
4.97	MatrixProgrammingSolution Class Reference	213
4.97.1	Detailed Description	214
4.97.2	Member Function Documentation	214
4.97.2.1	setRandom	214
4.97.2.2	deepCopyFrom	214
4.98	MatrixTransformation Class Reference	214
4.98.1	Detailed Description	215
4.98.2	Member Function Documentation	215
4.98.2.1	getNodeType	215
4.98.2.2	getNodeName	215
4.98.2.3	getMatrixType	216
4.98.2.4	getMatrixNodeInXML	216
4.98.2.5	alignsOnBlockBoundary	216
4.98.2.6	cloneMatrixNode	216
4.98.2.7	setRandom	216
4.98.2.8	deepCopyFrom	217
4.98.3	Member Data Documentation	217
4.98.3.1	shape	217
4.99	MatrixType Class Reference	217
4.99.1	Detailed Description	219
4.99.2	Member Function Documentation	219
4.99.2.1	alignsOnBlockBoundary	219
4.99.2.2	printExpandedMatrix	219
4.99.2.3	getRowPartitionSize	219
4.99.2.4	getRowPartition	220
4.99.2.5	getColumnPartitionSize	220

4.99.2.6	getColumnPartition	220
4.99.2.7	expandElements	220
4.99.2.8	convertToOtherMajor	220
4.99.2.9	processBlockPartition	221
4.99.2.10	processBlocks	221
4.99.2.11	processBlocks	221
4.99.2.12	extractBlock	222
4.99.2.13	getBlocks	222
4.99.2.14	disassembleMatrix	224
4.99.2.15	setRandom	224
4.99.2.16	deepCopyFrom	225
4.99.3	Member Data Documentation	225
4.99.3.1	symmetry	225
4.99.3.2	type	225
4.100	MatrixVar Class Reference	225
4.100.1	Detailed Description	226
4.100.2	Member Data Documentation	226
4.100.2.1	varType	226
4.101	MatrixVariables Class Reference	227
4.101.1	Detailed Description	227
4.102	MatrixVariableSolution Class Reference	227
4.102.1	Detailed Description	228
4.103	MatrixVariableValues Class Reference	228
4.103.1	Detailed Description	229
4.104	MaxTime Class Reference	229
4.104.1	Detailed Description	229
4.105	MinCPUNumber Class Reference	230
4.105.1	Detailed Description	230
4.106	MinCPUSpeed Class Reference	231
4.106.1	Detailed Description	231
4.107	MinDiskSpace Class Reference	232
4.107.1	Detailed Description	233
4.108	MinMemorySize Class Reference	233
4.108.1	Detailed Description	234
4.109	MixedRowReferenceMatrixElements Class Reference	234
4.109.1	Detailed Description	235
4.109.2	Member Function Documentation	235

4.109.2.1	getNodeType	235
4.109.2.2	getMatrixType	235
4.109.2.3	getNodeName	235
4.109.2.4	getMatrixNodeInXML	235
4.109.2.5	alignsOnBlockBoundary	236
4.109.2.6	cloneMatrixNode	236
4.109.2.7	setRandom	236
4.109.2.8	deepCopyFrom	236
4.109.3	Member Data Documentation	236
4.109.3.1	value	237
4.110	NI Class Reference	237
4.110.1	Detailed Description	237
4.110.2	Member Data Documentation	238
4.110.2.1	shape	238
4.111	NonlinearExpressions Class Reference	238
4.111.1	Detailed Description	238
4.112	NonnegativeCone Class Reference	239
4.112.1	Detailed Description	239
4.112.2	Member Function Documentation	239
4.112.2.1	getConeName	239
4.112.2.2	getConeInXML	240
4.112.2.3	setRandom	240
4.112.2.4	deepCopyFrom	240
4.113	NonpositiveCone Class Reference	240
4.113.1	Detailed Description	241
4.113.2	Member Function Documentation	241
4.113.2.1	getConeName	241
4.113.2.2	getConeInXML	241
4.113.2.3	setRandom	241
4.113.2.4	deepCopyFrom	242
4.114	ObjCoef Class Reference	242
4.114.1	Detailed Description	243
4.115	Objective Class Reference	243
4.115.1	Detailed Description	243
4.116	ObjectiveOption Class Reference	244
4.116.1	Detailed Description	244
4.116.2	Member Function Documentation	245

4.116.2.1 setRandom	245
4.116.2.2 deepCopyFrom	245
4.116.2.3 setOther	245
4.116.2.4 addOther	245
4.117Objectives Class Reference	246
4.117.1 Detailed Description	246
4.118ObjectiveSolution Class Reference	246
4.118.1 Detailed Description	247
4.118.2 Member Function Documentation	247
4.118.2.1 setRandom	247
4.119ObjectiveValues Class Reference	248
4.119.1 Detailed Description	248
4.119.2 Member Function Documentation	249
4.119.2.1 setRandom	249
4.120ObjReferenceMatrixElements Class Reference	249
4.120.1 Detailed Description	250
4.120.2 Member Function Documentation	250
4.120.2.1 getNodeType	250
4.120.2.2 getMatrixType	250
4.120.2.3 getNodeName	250
4.120.2.4 getMatrixNodeInXML	250
4.120.2.5 alignsOnBlockBoundary	250
4.120.2.6 cloneMatrixNode	251
4.120.2.7 setRandom	251
4.120.2.8 deepCopyFrom	251
4.121ObjReferenceMatrixValues Class Reference	251
4.121.1 Detailed Description	252
4.121.2 Member Function Documentation	252
4.121.2.1 setRandom	252
4.121.2.2 deepCopyFrom	252
4.122ObjValue Class Reference	253
4.122.1 Detailed Description	253
4.122.2 Member Function Documentation	254
4.122.2.1 setRandom	254
4.123OptimizationOption Class Reference	255
4.123.1 Detailed Description	256
4.123.2 Member Function Documentation	256

4.123.2.1 setRandom	256
4.123.2.2 deepCopyFrom	256
4.124 OptimizationResult Class Reference	257
4.124.1 Detailed Description	257
4.124.2 Member Function Documentation	258
4.124.2.1 setRandom	258
4.125 OptimizationSolution Class Reference	258
4.125.1 Detailed Description	259
4.125.2 Member Function Documentation	259
4.125.2.1 setRandom	259
4.126 OptimizationSolutionStatus Class Reference	260
4.126.1 Detailed Description	260
4.126.2 Member Function Documentation	261
4.126.2.1 setRandom	261
4.127 OptimizationSolutionSubstatus Class Reference	261
4.127.1 Detailed Description	262
4.127.2 Member Function Documentation	262
4.127.2.1 setRandom	262
4.128 OrthantCone Class Reference	262
4.128.1 Detailed Description	263
4.128.2 Member Function Documentation	263
4.128.2.1 getConeName	263
4.128.2.2 getConeInXML	263
4.128.2.3 setRandom	263
4.128.2.4 deepCopyFrom	264
4.129 OS_AMPL_SUFFIX Struct Reference	264
4.129.1 Detailed Description	264
4.130 OSCommandLine Class Reference	264
4.130.1 Detailed Description	267
4.130.2 Member Function Documentation	267
4.130.2.1 convertSolverNameToUpperCase	267
4.130.3 Member Data Documentation	267
4.130.3.1 osinstance	267
4.130.3.2 osoption	267
4.130.3.3 serviceMethod	267
4.130.3.4 osilOutputFile	267
4.130.3.5 osolOutputFile	268

4.130.3.6 osplOutputFile	268
4.130.3.7 browser	268
4.130.3.8 quit	268
4.131OSCommandLineReader Class Reference	268
4.131.1 Detailed Description	269
4.131.2 Constructor & Destructor Documentation	269
4.131.2.1 OSCommandLineReader	269
4.131.3 Member Function Documentation	269
4.131.3.1 readCommandLine	269
4.131.3.2 parseString	269
4.132OSExpressionTree Class Reference	270
4.132.1 Detailed Description	270
4.132.2 Member Data Documentation	271
4.132.2.1 m_bIndexMapGenerated	271
4.132.2.2 bADMustReTape	271
4.133OSGams2osil Class Reference	271
4.133.1 Detailed Description	271
4.133.2 Member Function Documentation	271
4.133.2.1 createOSInstance	271
4.133.2.2 takeOverOSInstance	272
4.133.2.3 getOSInstance	272
4.134OSGeneral Class Reference	272
4.134.1 Detailed Description	272
4.135OSgLParserData Class Reference	272
4.135.1 Detailed Description	273
4.135.2 Member Data Documentation	273
4.135.2.1 matrixWithMatrixVarIdx	273
4.136OShL Class Reference	274
4.136.1 Detailed Description	274
4.136.2 Member Function Documentation	274
4.136.2.1 solve	274
4.136.2.2 getJobID	275
4.136.2.3 send	275
4.136.2.4 kill	275
4.136.2.5 retrieve	276
4.136.2.6 knock	277
4.137OSiLParserData Class Reference	277

4.137.1 Detailed Description	279
4.137.2 Member Data Documentation	279
4.137.2.1 qtermcount	279
4.137.2.2 timeDomainStages	279
4.137.2.3 stageVariablesON	279
4.137.2.4 stageVariablesOrdered	279
4.137.2.5 stageVariableStartIdx	280
4.137.2.6 nvarcovered	280
4.138OSiLReader Class Reference	280
4.138.1 Detailed Description	280
4.138.2 Member Function Documentation	280
4.138.2.1 readOSiL	280
4.139OSiLWriter Class Reference	281
4.139.1 Detailed Description	281
4.139.2 Member Function Documentation	281
4.139.2.1 writeOSiL	281
4.140OSInstance Class Reference	282
4.140.1 Detailed Description	290
4.140.2 Member Function Documentation	290
4.140.2.1 getInstanceName	290
4.140.2.2 getInstanceSource	291
4.140.2.3 getInstanceDescription	291
4.140.2.4 getInstanceCreator	291
4.140.2.5 getInstanceLicence	291
4.140.2.6 getVariableNumber	291
4.140.2.7 getVariableNames	291
4.140.2.8 getVariableTypes	292
4.140.2.9 getNumberOfIntegerVariables	292
4.140.2.10getNumberOfBinaryVariables	293
4.140.2.11getNumberOfSemiContinuousVariables	293
4.140.2.12getNumberOfSemiIntegerVariables	293
4.140.2.13getNumberOfStringVariables	293
4.140.2.14getVariableLowerBounds	293
4.140.2.15getVariableUpperBounds	293
4.140.2.16getObjectiveNumber	294
4.140.2.17getObjectiveNames	294
4.140.2.18getObjectiveMaxOrMins	294

4.140.2.19	getObjectiveCoefficientNumbers	294
4.140.2.20	getObjectiveConstants	295
4.140.2.21	getObjectiveWeights	295
4.140.2.22	getObjectiveCoefficients	295
4.140.2.23	getDenseObjectiveCoefficients	296
4.140.2.24	getConstraintNumber	296
4.140.2.25	getConstraintNames	296
4.140.2.26	getConstraintLowerBounds	296
4.140.2.27	getConstraintUpperBounds	296
4.140.2.28	getConstraintConstants	297
4.140.2.29	getConstraintTypes	297
4.140.2.30	getLinearConstraintCoefficientNumber	297
4.140.2.31	getLinearConstraintCoefficientMajor	297
4.140.2.32	getLinearConstraintCoefficientsInColumnMajor	298
4.140.2.33	getLinearConstraintCoefficientsInRowMajor	298
4.140.2.34	getNumberOfQuadraticTerms	298
4.140.2.35	getQuadraticTerms	298
4.140.2.36	getQuadraticRowIndexes	299
4.140.2.37	getNumberOfQuadraticRowIndexes	299
4.140.2.38	getNumberOfNonlinearExpressions	299
4.140.2.39	getNonlinearExpressions	299
4.140.2.40	getNonlinearExpressionTree	299
4.140.2.41	getNonlinearExpressionTreeMod	300
4.140.2.42	getNonlinearExpressionTreeInPostfix	300
4.140.2.43	getNonlinearExpressionTreeModInPostfix	300
4.140.2.44	getNonlinearExpressionTreeInPrefix	300
4.140.2.45	getNonlinearExpressionTreeInInfix	300
4.140.2.46	getNonlinearExpressionTreeModInPrefix	301
4.140.2.47	getNumberOfNonlinearObjectives	301
4.140.2.48	getNumberOfNonlinearConstraints	301
4.140.2.49	getAllNonlinearExpressionTrees	301
4.140.2.50	getAllNonlinearExpressionTreesMod	301
4.140.2.51	getNonlinearExpressionTreeIndexes	301
4.140.2.52	getNumberOfNonlinearExpressionTreeIndexes	301
4.140.2.53	getNonlinearExpressionTreeModIndexes	302
4.140.2.54	getNumberOfNonlinearExpressionTreeModIndexes	302
4.140.2.55	getMatrixNumber	302

4.140.2.56	getMatrixType	302
4.140.2.57	getMatrixSymmetry	302
4.140.2.58	getNumberOfColumnsForMatrix	303
4.140.2.59	getNumberOfRowsForMatrix	303
4.140.2.60	getNumberOfValuesForMatrix	303
4.140.2.61	getMatrixName	303
4.140.2.62	matrixHasBase	304
4.140.2.63	getMatrix	304
4.140.2.64	getMatrixCoefficientsInColumnMajor	304
4.140.2.65	getMatrixCoefficientsInRowMajor	304
4.140.2.66	getMatrixBlockInColumnMajorForm	305
4.140.2.67	getNumberOfMatrixVariables	306
4.140.2.68	getNumberOfMatrixObjectives	306
4.140.2.69	getNumberOfMatrixConstraints	306
4.140.2.70	getNumberOfMatrixExpressions	306
4.140.2.71	getMatrixExpressions	307
4.140.2.72	getMatrixExpressionTree	307
4.140.2.73	getMatrixExpressionTreeInPostfix	307
4.140.2.74	getMatrixExpressionTreeModInPostfix	307
4.140.2.75	getMatrixExpressionTreeInPrefix	307
4.140.2.76	getMatrixExpressionTreeInInfix	307
4.140.2.77	getAllMatrixExpressionTrees	308
4.140.2.78	getAllMatrixExpressionTreesMod	308
4.140.2.79	getMatrixExpressionTreeIndexes	308
4.140.2.80	getNumberOfMatrixExpressionTreeIndexes	308
4.140.2.81	getTimeDomainFormat	308
4.140.2.82	getTimeDomainStageNumber	309
4.140.2.83	getTimeDomainStageNames	309
4.140.2.84	getTimeDomainStageNumberOfVariables	309
4.140.2.85	getTimeDomainStageNumberOfConstraints	309
4.140.2.86	getTimeDomainStageNumberOfObjectives	309
4.140.2.87	getTimeDomainStageVarList	309
4.140.2.88	getTimeDomainStageConList	310
4.140.2.89	getTimeDomainStageObjList	310
4.140.2.90	getTimeDomainIntervalStart	310
4.140.2.91	getTimeDomainIntervalHorizon	310
4.140.2.92	setInstanceName	310

4.140.2.93	setInstanceSource	310
4.140.2.94	setInstanceDescription	311
4.140.2.95	setInstanceCreator	311
4.140.2.96	setInstanceLicence	311
4.140.2.97	setVariableNumber	311
4.140.2.98	addVariable	312
4.140.2.99	setVariables	312
4.140.2.100	setObjectiveNumber	313
4.140.2.101	addObjective	313
4.140.2.102	setObjectives	313
4.140.2.103	setConstraintNumber	314
4.140.2.104	addConstraint	314
4.140.2.105	setConstraints	314
4.140.2.106	setLinearConstraintCoefficients	315
4.140.2.107	copyLinearConstraintCoefficients	315
4.140.2.108	setNumberOfQuadraticTerms	316
4.140.2.109	setQuadraticCoefficients	316
4.140.2.110	setQuadraticTermsInNonlinearExpressions	316
4.140.2.111	setNonlinearExpressions	317
4.140.2.112	setMatrixNumber	317
4.140.2.113	addMatrix	317
4.140.2.114	setConeNumber	318
4.140.2.115	addCone	318
4.140.2.116	addCone	319
4.140.2.117	addCone	320
4.140.2.118	addCone	320
4.140.2.119	addCone	321
4.140.2.120	addCone	322
4.140.2.121	addCone	322
4.140.2.122	addCone	323
4.140.2.123	printModel	324
4.140.2.124	printModel	324
4.140.2.125	initializeNonLinearStructures	324
4.140.2.126	calculateFunctionValue	324
4.140.2.127	calculateAllConstraintFunctionValues	325
4.140.2.128	calculateAllConstraintFunctionValues	325
4.140.2.129	calculateAllObjectiveFunctionValues	325

4.140.2.130	CalculateAllObjectiveFunctionValues	326
4.140.2.131	CalculateAllConstraintFunctionGradients	326
4.140.2.132	CalculateConstraintFunctionGradient	326
4.140.2.133	CalculateConstraintFunctionGradient	327
4.140.2.134	CalculateAllObjectiveFunctionGradients	327
4.140.2.135	CalculateObjectiveFunctionGradient	327
4.140.2.136	CalculateObjectiveFunctionGradient	328
4.140.2.137	CalculateLagrangianHessian	328
4.140.2.138	CalculateHessian	328
4.140.2.139	GetSparseJacobianFromColumnMajor	329
4.140.2.140	GetSparseJacobianFromRowMajor	329
4.140.2.141	GetLagrangianExpTree	329
4.140.2.142	GetAllNonlinearVariablesIndexMap	329
4.140.2.143	GetLagrangianHessianSparsityPattern	329
4.140.2.144	AddQTermsToExpressionTree	329
4.140.2.145	AddQTermsToExpressionTree	330
4.140.2.146	GetJacobianSparsityPattern	330
4.140.2.147	CreateOSADFun	330
4.140.2.148	ForwardAD	330
4.140.2.149	ReverseAD	330
4.140.2.150	GetADSparsityHessian	331
4.140.2.151	GetIterateResults	331
4.140.2.152	GetZeroOrderResults	331
4.140.2.153	GetFirstOrderResults	332
4.140.2.154	GetSecondOrderResults	333
4.140.2.155	InitForAlgDiff	333
4.140.2.156	InitObjGradients	333
4.140.2.157	GetTimeDomainStageVariablesOrdered	333
4.140.2.158	GetTimeDomainStageVariablesUnordered	334
4.140.2.159	GetTimeDomainStageConstraintsOrdered	334
4.140.2.160	GetTimeDomainStageConstraintsUnordered	334
4.140.2.161	GetTimeDomainStageObjectivesOrdered	334
4.140.2.162	GetTimeDomainStageObjectivesUnordered	334
4.141	OSMatlab Class Reference	334
4.141.1	Detailed Description	336
4.141.2	Member Function Documentation	336
4.141.2.1	solve	336

4.142OSMatrix Class Reference	337
4.142.1 Detailed Description	337
4.142.2 Member Function Documentation	337
4.142.2.1 createConstructorTreeFromPrefix	337
4.142.2.2 getNodeType	338
4.142.2.3 getNodeName	338
4.142.2.4 getMatrixType	338
4.142.2.5 expandElements	338
4.142.2.6 alignsOnBlockBoundary	339
4.142.2.7 setMatrix	339
4.142.2.8 getMatrixNodeInXML	340
4.142.2.9 cloneMatrixNode	340
4.142.2.10setRandom	340
4.142.2.11deepCopyFrom	340
4.143OSMatrixWithMatrixConIdx Class Reference	341
4.143.1 Detailed Description	341
4.143.2 Member Function Documentation	341
4.143.2.1 getMatrixNodeInXML	341
4.143.2.2 cloneMatrixNode	342
4.143.2.3 setRandom	342
4.143.2.4 deepCopyFrom	342
4.144OSMatrixWithMatrixObjIdx Class Reference	342
4.144.1 Detailed Description	343
4.144.2 Member Function Documentation	343
4.144.2.1 getMatrixNodeInXML	343
4.144.2.2 cloneMatrixNode	343
4.144.2.3 setRandom	343
4.144.2.4 deepCopyFrom	343
4.145OSMatrixWithMatrixVarIdx Class Reference	344
4.145.1 Detailed Description	344
4.145.2 Member Function Documentation	344
4.145.2.1 getMatrixNodeInXML	344
4.145.2.2 cloneMatrixNode	345
4.145.2.3 setRandom	345
4.145.2.4 deepCopyFrom	345
4.146OSmps2OS Class Reference	345
4.146.1 Detailed Description	346

4.146.2 Member Function Documentation	346
4.146.2.1 createOSObjects	346
4.146.3 Member Data Documentation	347
4.146.3.1 osol	347
4.146.3.2 jobID	347
4.147OSmps2osil Class Reference	347
4.147.1 Detailed Description	347
4.147.2 Member Function Documentation	348
4.147.2.1 createOSInstance	348
4.148OSnl2OS Class Reference	348
4.148.1 Detailed Description	349
4.148.2 Constructor & Destructor Documentation	350
4.148.2.1 OSnl2OS	350
4.148.3 Member Function Documentation	350
4.148.3.1 getASL	350
4.148.3.2 readNI	350
4.148.3.3 setASL	350
4.148.3.4 createOSObjects	351
4.148.3.5 setVar	351
4.148.3.6 setIBVar	351
4.148.3.7 walkTree	351
4.148.4 Member Data Documentation	351
4.148.4.1 osol	351
4.148.4.2 jobID	352
4.149OSnLMNode Class Reference	352
4.149.1 Detailed Description	353
4.149.2 Member Function Documentation	353
4.149.2.1 createExpressionTreeFromPrefix	353
4.149.2.2 getPrefixFromExpressionTree	353
4.149.2.3 preOrderOSnLNodeTraversal	353
4.149.2.4 createExpressionTreeFromPostfix	354
4.149.2.5 getPostfixFromExpressionTree	354
4.149.2.6 postOrderOSnLNodeTraversal	354
4.149.2.7 copyNodeAndDescendants	354
4.150OSnLMNodeDiagonalMatrixFromVector Class Reference	355
4.150.1 Detailed Description	355
4.150.2 Member Function Documentation	355

4.150.2.1 getTokenName	355
4.150.2.2 cloneExprNode	355
4.151 OSnLMNodeIdentityMatrix Class Reference	356
4.151.1 Detailed Description	356
4.151.2 Member Function Documentation	356
4.151.2.1 getTokenName	356
4.151.2.2 cloneExprNode	356
4.152 OSnLMNodeMatrixCon Class Reference	357
4.152.1 Detailed Description	357
4.152.2 Member Function Documentation	357
4.152.2.1 getTokenName	357
4.152.2.2 getTokenNumber	357
4.152.2.3 getNonlinearExpressionInXML	358
4.152.2.4 cloneExprNode	358
4.152.2.5 copyNodeAndDescendants	358
4.153 OSnLMNodeMatrixDiagonal Class Reference	358
4.153.1 Detailed Description	358
4.153.2 Member Function Documentation	359
4.153.2.1 getTokenName	359
4.153.2.2 cloneExprNode	359
4.154 OSnLMNodeMatrixDotTimes Class Reference	359
4.154.1 Detailed Description	359
4.154.2 Member Function Documentation	359
4.154.2.1 getTokenName	359
4.154.2.2 cloneExprNode	360
4.155 OSnLMNodeMatrixInverse Class Reference	360
4.155.1 Detailed Description	360
4.155.2 Member Function Documentation	360
4.155.2.1 getTokenName	360
4.155.2.2 cloneExprNode	361
4.156 OSnLMNodeMatrixLowerTriangle Class Reference	361
4.156.1 Detailed Description	361
4.156.2 Member Function Documentation	361
4.156.2.1 getTokenName	361
4.156.2.2 getNonlinearExpressionInXML	362
4.156.2.3 cloneExprNode	362
4.156.2.4 copyNodeAndDescendants	362

4.157OSnLMNodeMatrixMinus Class Reference	362
4.157.1 Detailed Description	363
4.157.2 Member Function Documentation	363
4.157.2.1 getTokenName	363
4.157.2.2 cloneExprNode	363
4.158OSnLMNodeMatrixNegate Class Reference	363
4.158.1 Detailed Description	363
4.158.2 Member Function Documentation	364
4.158.2.1 getTokenName	364
4.158.2.2 cloneExprNode	364
4.159OSnLMNodeMatrixObj Class Reference	364
4.159.1 Detailed Description	365
4.159.2 Member Function Documentation	365
4.159.2.1 getTokenName	365
4.159.2.2 getTokenNumber	365
4.159.2.3 getNonlinearExpressionInXML	365
4.159.2.4 cloneExprNode	365
4.159.2.5 copyNodeAndDescendants	365
4.160OSnLMNodeMatrixPlus Class Reference	366
4.160.1 Detailed Description	366
4.160.2 Member Function Documentation	366
4.160.2.1 getTokenName	366
4.160.2.2 cloneExprNode	366
4.161OSnLMNodeMatrixProduct Class Reference	366
4.161.1 Detailed Description	367
4.161.2 Member Function Documentation	367
4.161.2.1 getTokenName	367
4.161.2.2 cloneExprNode	368
4.162OSnLMNodeMatrixReference Class Reference	368
4.162.1 Detailed Description	368
4.162.2 Member Function Documentation	369
4.162.2.1 getTokenName	369
4.162.2.2 getTokenNumber	369
4.162.2.3 getNonlinearExpressionInXML	369
4.162.2.4 cloneExprNode	369
4.162.2.5 copyNodeAndDescendants	369
4.163OSnLMNodeMatrixScalarTimes Class Reference	369

4.163.1 Detailed Description	370
4.163.2 Member Function Documentation	370
4.163.2.1 getTokenName	370
4.163.2.2 cloneExprNode	370
4.164OSnLMNodeMatrixSubmatrixAt Class Reference	370
4.164.1 Detailed Description	371
4.164.2 Member Function Documentation	371
4.164.2.1 getTokenName	371
4.164.2.2 cloneExprNode	371
4.165OSnLMNodeMatrixSum Class Reference	371
4.165.1 Detailed Description	372
4.165.2 Member Function Documentation	372
4.165.2.1 getTokenName	372
4.165.2.2 cloneExprNode	372
4.166OSnLMNodeMatrixTimes Class Reference	372
4.166.1 Detailed Description	372
4.166.2 Member Function Documentation	373
4.166.2.1 getTokenName	373
4.166.2.2 cloneExprNode	373
4.167OSnLMNodeMatrixTranspose Class Reference	373
4.167.1 Detailed Description	373
4.167.2 Member Function Documentation	373
4.167.2.1 getTokenName	373
4.167.2.2 cloneExprNode	374
4.168OSnLMNodeMatrixUpperTriangle Class Reference	374
4.168.1 Detailed Description	374
4.168.2 Member Function Documentation	375
4.168.2.1 getTokenName	375
4.168.2.2 getNonlinearExpressionInXML	375
4.168.2.3 cloneExprNode	375
4.168.2.4 copyNodeAndDescendants	375
4.169OSnLMNodeMatrixVar Class Reference	375
4.169.1 Detailed Description	376
4.169.2 Member Function Documentation	376
4.169.2.1 getTokenName	376
4.169.2.2 getTokenNumber	376
4.169.2.3 getNonlinearExpressionInXML	376

4.169.2.4 cloneExprNode	377
4.169.2.5 copyNodeAndDescendants	377
4.170OSnLNode Class Reference	377
4.170.1 Detailed Description	378
4.170.2 Member Function Documentation	378
4.170.2.1 getVariableIndexMap	378
4.170.2.2 calculateFunction	379
4.170.2.3 constructADTape	379
4.170.2.4 createExpressionTreeFromPrefix	379
4.170.2.5 getPrefixFromExpressionTree	380
4.170.2.6 preOrderOSnLNodeTraversal	380
4.170.2.7 createExpressionTreeFromPostfix	380
4.170.2.8 getPostfixFromExpressionTree	380
4.170.2.9 postOrderOSnLNodeTraversal	381
4.170.2.10 copyNodeAndDescendants	381
4.171OSnLNodeAbs Class Reference	381
4.171.1 Detailed Description	382
4.171.2 Member Function Documentation	382
4.171.2.1 getTokenName	382
4.171.2.2 calculateFunction	382
4.171.2.3 cloneExprNode	383
4.171.2.4 constructADTape	383
4.172OSnLNodeAllDiff Class Reference	383
4.172.1 Detailed Description	384
4.172.2 Member Function Documentation	384
4.172.2.1 getTokenName	384
4.172.2.2 calculateFunction	384
4.172.2.3 cloneExprNode	385
4.172.2.4 constructADTape	385
4.173OSnLNodeCos Class Reference	385
4.173.1 Detailed Description	386
4.173.2 Member Function Documentation	386
4.173.2.1 getTokenName	386
4.173.2.2 calculateFunction	386
4.173.2.3 cloneExprNode	387
4.173.2.4 constructADTape	387
4.174OSnLNodeDivide Class Reference	387

4.174.1 Detailed Description	388
4.174.2 Member Function Documentation	388
4.174.2.1 getTokenName	388
4.174.2.2 calculateFunction	388
4.174.2.3 cloneExprNode	389
4.174.2.4 constructADTape	389
4.175OSnLNodeE Class Reference	389
4.175.1 Detailed Description	390
4.175.2 Member Function Documentation	390
4.175.2.1 getTokenNumber	390
4.175.2.2 getTokenName	390
4.175.2.3 getNonlinearExpressionInXML	391
4.175.2.4 calculateFunction	391
4.175.2.5 cloneExprNode	391
4.175.2.6 constructADTape	391
4.176OSnLNodeErf Class Reference	391
4.176.1 Detailed Description	392
4.176.2 Member Function Documentation	392
4.176.2.1 getTokenName	392
4.176.2.2 calculateFunction	393
4.176.2.3 cloneExprNode	393
4.176.2.4 constructADTape	393
4.177OSnLNodeExp Class Reference	393
4.177.1 Detailed Description	394
4.177.2 Member Function Documentation	394
4.177.2.1 getTokenName	394
4.177.2.2 calculateFunction	394
4.177.2.3 constructADTape	395
4.177.2.4 cloneExprNode	395
4.178OSnLNodeIf Class Reference	395
4.178.1 Detailed Description	396
4.178.2 Member Function Documentation	396
4.178.2.1 getTokenName	396
4.178.2.2 calculateFunction	396
4.178.2.3 cloneExprNode	397
4.178.2.4 constructADTape	397
4.179OSnLNodeLn Class Reference	397

4.179.1 Detailed Description	398
4.179.2 Member Function Documentation	398
4.179.2.1 getTokenName	398
4.179.2.2 calculateFunction	398
4.179.2.3 cloneExprNode	399
4.179.2.4 constructADTape	399
4.180OSnLNodeMatrixDeterminant Class Reference	399
4.180.1 Detailed Description	400
4.180.2 Member Function Documentation	400
4.180.2.1 getTokenName	400
4.180.2.2 calculateFunction	400
4.180.2.3 constructADTape	401
4.180.2.4 cloneExprNode	401
4.181OSnLNodeMatrixToScalar Class Reference	401
4.181.1 Detailed Description	401
4.181.2 Member Function Documentation	402
4.181.2.1 getTokenName	402
4.181.2.2 calculateFunction	402
4.181.2.3 constructADTape	402
4.181.2.4 cloneExprNode	403
4.182OSnLNodeMatrixTrace Class Reference	403
4.182.1 Detailed Description	403
4.182.2 Member Function Documentation	404
4.182.2.1 getTokenName	404
4.182.2.2 calculateFunction	404
4.182.2.3 constructADTape	404
4.182.2.4 cloneExprNode	404
4.183OSnLNodeMax Class Reference	405
4.183.1 Detailed Description	405
4.183.2 Member Function Documentation	406
4.183.2.1 getTokenName	406
4.183.2.2 calculateFunction	406
4.183.2.3 cloneExprNode	406
4.183.2.4 constructADTape	406
4.184OSnLNodeMin Class Reference	407
4.184.1 Detailed Description	407
4.184.2 Member Function Documentation	408

4.184.2.1 getTokenName	408
4.184.2.2 calculateFunction	408
4.184.2.3 cloneExprNode	408
4.184.2.4 constructADTape	408
4.185OSnLNodeMinus Class Reference	409
4.185.1 Detailed Description	409
4.185.2 Member Function Documentation	410
4.185.2.1 getTokenName	410
4.185.2.2 calculateFunction	410
4.185.2.3 cloneExprNode	410
4.185.2.4 constructADTape	410
4.186OSnLNodeNegate Class Reference	411
4.186.1 Detailed Description	411
4.186.2 Member Function Documentation	412
4.186.2.1 getTokenName	412
4.186.2.2 calculateFunction	412
4.186.2.3 cloneExprNode	412
4.186.2.4 constructADTape	412
4.187OSnLNodeNumber Class Reference	413
4.187.1 Detailed Description	413
4.187.2 Member Function Documentation	414
4.187.2.1 getTokenName	414
4.187.2.2 getTokenNumber	414
4.187.2.3 getNonlinearExpressionInXML	414
4.187.2.4 calculateFunction	414
4.187.2.5 cloneExprNode	415
4.187.2.6 copyNodeAndDescendants	415
4.187.2.7 constructADTape	415
4.187.3 Member Data Documentation	415
4.187.3.1 id	415
4.188OSnLNodePI Class Reference	415
4.188.1 Detailed Description	416
4.188.2 Member Function Documentation	416
4.188.2.1 getTokenNumber	416
4.188.2.2 getTokenName	417
4.188.2.3 getNonlinearExpressionInXML	417
4.188.2.4 calculateFunction	417

4.188.2.5 cloneExprNode	417
4.188.2.6 constructADTape	417
4.189OSnLNodePlus Class Reference	418
4.189.1 Detailed Description	418
4.189.2 Member Function Documentation	419
4.189.2.1 getTokenName	419
4.189.2.2 calculateFunction	419
4.189.2.3 constructADTape	419
4.189.2.4 cloneExprNode	419
4.190OSnLNodePower Class Reference	419
4.190.1 Detailed Description	420
4.190.2 Member Function Documentation	420
4.190.2.1 getTokenName	420
4.190.2.2 calculateFunction	420
4.190.2.3 cloneExprNode	421
4.190.2.4 constructADTape	421
4.191OSnLNodeProduct Class Reference	421
4.191.1 Detailed Description	422
4.191.2 Member Function Documentation	422
4.191.2.1 getTokenName	422
4.191.2.2 calculateFunction	422
4.191.2.3 cloneExprNode	423
4.191.2.4 constructADTape	423
4.192OSnLNodeSin Class Reference	423
4.192.1 Detailed Description	424
4.192.2 Member Function Documentation	424
4.192.2.1 getTokenName	424
4.192.2.2 calculateFunction	424
4.192.2.3 cloneExprNode	425
4.192.2.4 constructADTape	425
4.193OSnLNodeSqrt Class Reference	425
4.193.1 Detailed Description	426
4.193.2 Member Function Documentation	426
4.193.2.1 getTokenName	426
4.193.2.2 calculateFunction	426
4.193.2.3 cloneExprNode	427
4.193.2.4 constructADTape	427

4.194OSnLNodeSquare Class Reference	427
4.194.1 Detailed Description	428
4.194.2 Member Function Documentation	428
4.194.2.1 getTokenName	428
4.194.2.2 calculateFunction	428
4.194.2.3 cloneExprNode	429
4.194.2.4 constructADTape	429
4.195OSnLNodeSum Class Reference	429
4.195.1 Detailed Description	430
4.195.2 Member Function Documentation	430
4.195.2.1 getTokenName	430
4.195.2.2 calculateFunction	430
4.195.2.3 cloneExprNode	431
4.195.2.4 constructADTape	431
4.196OSnLNodeTimes Class Reference	431
4.196.1 Detailed Description	432
4.196.2 Member Function Documentation	432
4.196.2.1 getTokenName	432
4.196.2.2 calculateFunction	432
4.196.2.3 cloneExprNode	433
4.196.2.4 constructADTape	433
4.197OSnLNodeVariable Class Reference	433
4.197.1 Detailed Description	434
4.197.2 Member Function Documentation	434
4.197.2.1 getVariableIndexMap	434
4.197.2.2 getTokenNumber	435
4.197.2.3 getTokenName	435
4.197.2.4 getNonlinearExpressionInXML	435
4.197.2.5 calculateFunction	435
4.197.2.6 cloneExprNode	435
4.197.2.7 copyNodeAndDescendants	436
4.197.2.8 constructADTape	436
4.198OSnLParserData Class Reference	436
4.198.1 Detailed Description	438
4.198.2 Member Data Documentation	438
4.198.2.1 nlNodePoint	438
4.198.2.2 tmpnlcount	438

4.199OSoLParserData Class Reference	439
4.199.1 Detailed Description	440
4.200OSoLReader Class Reference	440
4.200.1 Detailed Description	441
4.200.2 Member Function Documentation	441
4.200.2.1 readOSoL	441
4.201OSoLWriter Class Reference	441
4.201.1 Detailed Description	442
4.201.2 Member Function Documentation	442
4.201.2.1 writeOSoL	442
4.202OSOption Class Reference	442
4.202.1 Detailed Description	450
4.202.2 Member Function Documentation	451
4.202.2.1 setHeader	451
4.202.2.2 setRandom	452
4.202.2.3 deepCopyFrom	452
4.202.2.4 getJobID	452
4.202.2.5 getNumberOfInitVarValues	452
4.202.2.6 getNumberOfInitVarValuesString	452
4.202.2.7 getNumberOfIntegerVariableBranchingWeights	453
4.202.2.8 getNumberOfSOS	453
4.202.2.9 getNumberOfSOSVarBranchingWeights	453
4.202.2.10getNumberOfOtherVariableOptions	453
4.202.2.11getNumberOfInitObjValues	453
4.202.2.12getNumberOfInitObjBounds	453
4.202.2.13getNumberOfOtherObjectiveOptions	454
4.202.2.14getNumberOfInitConValues	454
4.202.2.15getNumberOfInitDualVarValues	454
4.202.2.16getNumberOfOtherConstraintOptions	454
4.202.2.17getNumberOfSolverOptions	454
4.202.2.18getOtherGeneralOptions	454
4.202.2.19getOtherSystemOptions	455
4.202.2.20getOtherServiceOptions	455
4.202.2.21getOtherJobOptions	455
4.202.2.22getOtherOptions	455
4.202.2.23getAllOtherOptions	455
4.202.2.24getJobDependencies	455

4.202.2.25	getRequiredDirectories	456
4.202.2.26	getRequiredFiles	456
4.202.2.27	getDirectoriesToMake	456
4.202.2.28	getFilesToMake	456
4.202.2.29	getInputDirectoriesToMove	456
4.202.2.30	getInputFilesToMove	456
4.202.2.31	getOutputDirectoriesToMove	457
4.202.2.32	getOutputFilesToMove	457
4.202.2.33	getDirectoriesToDelete	457
4.202.2.34	getFilesToDelete	457
4.202.2.35	getProcessesToKill	457
4.202.2.36	getInitVarValuesSparse	457
4.202.2.37	getInitVarValuesDense	458
4.202.2.38	getInitVarValuesDense	458
4.202.2.39	getInitVarValuesStringSparse	458
4.202.2.40	getInitVarValuesStringDense	458
4.202.2.41	getInitVarValuesStringDense	458
4.202.2.42	getInitBasisStatusSparse	459
4.202.2.43	getInitBasisStatusDense	459
4.202.2.44	getVariableInitialBasisStatusDense	459
4.202.2.45	getNumberOfInitialBasisElements	459
4.202.2.46	getInitialBasisElements	459
4.202.2.47	getIntegerVariableBranchingWeightsSparse	460
4.202.2.48	getIntegerVariableBranchingWeightsDense	460
4.202.2.49	getIntegerVariableBranchingWeightsDense	460
4.202.2.50	getSOSVariableBranchingWeightsSparse	460
4.202.2.51	getOtherVariableOptions	460
4.202.2.52	getOtherVariableOption	461
4.202.2.53	getAllOtherVariableOptions	461
4.202.2.54	getInitObjValuesSparse	461
4.202.2.55	getInitObjValuesDense	461
4.202.2.56	getInitObjValuesDense	461
4.202.2.57	getInitObjBoundsSparse	462
4.202.2.58	getInitObjLowerBoundsDense	462
4.202.2.59	getInitObjLowerBoundsDense	462
4.202.2.60	getInitObjUpperBoundsDense	462
4.202.2.61	getInitObjUpperBoundsDense	462

4.202.2.62	getObjectiveInitialBasisStatusDense	. 463
4.202.2.63	getOtherObjectiveOptions	. 463
4.202.2.64	getOtherObjectiveOption	. 463
4.202.2.65	getAllOtherObjectiveOptions	. 464
4.202.2.66	getInitConValuesSparse	. 464
4.202.2.67	getInitConValuesDense	. 464
4.202.2.68	getInitConValuesDense	. 464
4.202.2.69	getInitDualVarValuesSparse	. 464
4.202.2.70	getInitDualVarLowerBoundsDense	. 464
4.202.2.71	getInitDualVarLowerBoundsDense	. 465
4.202.2.72	getInitDualVarUpperBoundsDense	. 466
4.202.2.73	getInitDualVarUpperBoundsDense	. 466
4.202.2.74	getSlackVariableInitialBasisStatusDense	. 466
4.202.2.75	getOtherConstraintOptions	. 466
4.202.2.76	getOtherConstraintOption	. 467
4.202.2.77	getAllOtherConstraintOptions	. 467
4.202.2.78	getSolverOptions	. 467
4.202.2.79	getSolverOptions	. 467
4.202.2.80	getAllSolverOptions	. 468
4.202.2.81	setOtherGeneralOptions	. 468
4.202.2.82	setAnotherGeneralOption	. 468
4.202.2.83	setMinDiskSpace	. 468
4.202.2.84	setMinMemorySize	. 468
4.202.2.85	setMinCPUSpeed	. 469
4.202.2.86	setMinCPUNumber	. 469
4.202.2.87	setPathPairs	. 469
4.202.2.88	setAnotherInitBasisStatus	. 469
4.202.2.89	setOtherVariableOptionAttributes	. 470
4.202.2.90	setOtherOptionOrResultEnumeration	. 471
4.202.2.91	setOtherVariableOptionVar	. 471
4.202.2.92	setOtherObjectiveOptionAttributes	. 472
4.202.2.93	setOtherObjectiveOptionObj	. 473
4.202.2.94	setOtherConstraintOptionAttributes	. 473
4.202.2.95	setOtherConstraintOptionCon	. 474
4.202.2.96	setSolverOptionContent	. 474
4.202.3	Member Data Documentation	. 474
4.202.3.1	optionHeader	. 474

4.203osOptionsStruc Struct Reference	475
4.203.1 Detailed Description	477
4.203.2 Member Data Documentation	477
4.203.2.1 serviceMethod	477
4.203.2.2 solverName	477
4.203.2.3 browser	477
4.204OSosr12ampl Class Reference	477
4.204.1 Detailed Description	478
4.204.2 Member Function Documentation	478
4.204.2.1 writeSolFile	478
4.205OSOutput Class Reference	478
4.205.1 Detailed Description	479
4.205.2 Member Function Documentation	479
4.205.2.1 OSPrint	479
4.205.2.2 SetPrintLevel	480
4.205.2.3 SetPrintLevel	481
4.205.2.4 AddChannel	481
4.205.2.5 DeleteChannel	481
4.205.2.6 FindChannel	481
4.206OSOutputChannel Class Reference	482
4.206.1 Detailed Description	483
4.206.2 Constructor & Destructor Documentation	483
4.206.2.1 OSOutputChannel	483
4.206.3 Member Function Documentation	483
4.206.3.1 setPrintLevel	483
4.206.3.2 setAllPrintLevels	483
4.206.3.3 setAllPrintLevels	483
4.206.3.4 OSPrintf	484
4.207OSReferencedObject Class Reference	484
4.207.1 Detailed Description	484
4.208OSReferencer Class Reference	486
4.208.1 Detailed Description	486
4.209OSResult Class Reference	486
4.209.1 Detailed Description	498
4.209.2 Member Function Documentation	499
4.209.2.1 setHeader	499
4.209.2.2 setRandom	500

4.209.2.3	getGeneralStatus	500
4.209.2.4	getGeneralStatusType	500
4.209.2.5	getGeneralStatusDescription	500
4.209.2.6	getNumberOfGeneralSubstatuses	500
4.209.2.7	getGeneralSubstatusName	501
4.209.2.8	getGeneralSubstatusDescription	502
4.209.2.9	getGeneralMessage	502
4.209.2.10	getServiceName	502
4.209.2.11	getServiceURI	502
4.209.2.12	getInstanceName	502
4.209.2.13	getJobID	503
4.209.2.14	getSolverInvoked	503
4.209.2.15	getTimeStamp	503
4.209.2.16	getNumberOfOtherGeneralResults	503
4.209.2.17	getOtherGeneralResultName	503
4.209.2.18	getTimeNumber	503
4.209.2.19	getTimeValue	504
4.209.2.20	getVariableNumber	504
4.209.2.21	getObjectiveNumber	504
4.209.2.22	getConstraintNumber	504
4.209.2.23	getSolutionNumber	504
4.209.2.24	getSolutionStatus	504
4.209.2.25	getSolutionStatusType	505
4.209.2.26	getSolutionStatusDescription	505
4.209.2.27	getSolutionWeightedObjectives	505
4.209.2.28	getSolutionMessage	505
4.209.2.29	getOptimalPrimalVariableValues	506
4.209.2.30	getBasisStatusNumberOfEI	506
4.209.2.31	getBasisStatusEI	506
4.209.2.32	getBasisInformationDense	506
4.209.2.33	getNumberOfOtherVariableResults	507
4.209.2.34	getAnotherVariableResultNumberOfVar	507
4.209.2.35	getOtherVariableResultArrayType	507
4.209.2.36	getOtherVariableResultEnumerationValue	507
4.209.2.37	getOtherVariableResultEnumerationDescription	508
4.209.2.38	getOtherVariableResultEnumerationNumberOfEI	508
4.209.2.39	getOtherVariableResultEnumerationEI	508

4.209.2.40	getOtherVariableResultArrayDense	508
4.209.2.41	getOptimalObjValue	509
4.209.2.42	getOtherObjectiveResultArrayType	509
4.209.2.43	getOtherObjectiveResultEnumerationValue	509
4.209.2.44	getOtherObjectiveResultEnumerationDescription	510
4.209.2.45	getOtherObjectiveResultEnumerationNumberOfEl	511
4.209.2.46	getOtherObjectiveResultEnumerationEl	511
4.209.2.47	getOtherObjectiveResultArrayDense	511
4.209.2.48	getOptimalDualVariableValues	512
4.209.2.49	getOtherConstraintResultArrayType	513
4.209.2.50	getOtherConstraintResultEnumerationValue	513
4.209.2.51	getOtherConstraintResultEnumerationDescription	513
4.209.2.52	getOtherConstraintResultEnumerationNumberOfEl	513
4.209.2.53	getOtherConstraintResultEnumerationEl	514
4.209.2.54	getOtherConstraintResultArrayDense	514
4.209.2.55	setGeneralStatus	514
4.209.2.56	setGeneralStatusType	514
4.209.2.57	setNumberOfGeneralSubstatuses	515
4.209.2.58	setGeneralStatusDescription	515
4.209.2.59	setGeneralSubstatusName	515
4.209.2.60	setGeneralSubstatusDescription	515
4.209.2.61	setGeneralMessage	516
4.209.2.62	setServiceName	516
4.209.2.63	setServiceURI	516
4.209.2.64	setInstanceName	516
4.209.2.65	setJobID	517
4.209.2.66	setSolverInvoked	517
4.209.2.67	setTimeStamp	517
4.209.2.68	setNumberOfOtherGeneralResults	517
4.209.2.69	setOtherGeneralResultName	518
4.209.2.70	setOtherGeneralResultValue	518
4.209.2.71	setOtherGeneralResultDescription	518
4.209.2.72	setSystemInformation	518
4.209.2.73	setAvailableDiskSpaceUnit	519
4.209.2.74	setAvailableDiskSpaceDescription	519
4.209.2.75	setAvailableDiskSpaceValue	519
4.209.2.76	setAvailableMemoryUnit	519

4.209.2.77	setAvailableMemoryDescription	520
4.209.2.78	setAvailableMemoryValue	520
4.209.2.79	setAvailableCPUSpeedUnit	520
4.209.2.80	setAvailableCPUSpeedDescription	520
4.209.2.81	setAvailableCPUSpeedValue	521
4.209.2.82	setAvailableCPUNumberDescription	521
4.209.2.83	setAvailableCPUNumberValue	521
4.209.2.84	setNumberOfOtherSystemResults	521
4.209.2.85	setOtherSystemResultName	522
4.209.2.86	setOtherSystemResultValue	522
4.209.2.87	setOtherSystemResultDescription	522
4.209.2.88	setCurrentState	522
4.209.2.89	setCurrentJobCount	523
4.209.2.90	setTotalJobsSoFar	523
4.209.2.91	setTimeServiceStarted	523
4.209.2.92	setServiceUtilization	523
4.209.2.93	setNumberOfOtherServiceResults	524
4.209.2.94	setOtherServiceResultName	524
4.209.2.95	setOtherServiceResultValue	524
4.209.2.96	setOtherServiceResultDescription	524
4.209.2.97	setJobStatus	525
4.209.2.98	setJobSubmitTime	525
4.209.2.99	setScheduledStartTime	525
4.209.2.100	setActualStartTime	525
4.209.2.101	setJobEndTime	526
4.209.2.102	setTime	526
4.209.2.103	addTimingInformation	526
4.209.2.104	setTimingInformation	526
4.209.2.105	setNumberOfTimes	527
4.209.2.106	setTimeNumber	527
4.209.2.107	setUsedDiskSpaceUnit	527
4.209.2.108	setUsedDiskSpaceDescription	527
4.209.2.109	setUsedDiskSpaceValue	528
4.209.2.110	setUsedMemoryUnit	528
4.209.2.111	setUsedMemoryDescription	528
4.209.2.112	setUsedMemoryValue	528
4.209.2.113	setUsedCPUSpeedUnit	529

4.209.2.114	SetUsedCPUSpeedDescription	529
4.209.2.115	SetUsedCPUSpeedValue	529
4.209.2.116	SetUsedCPUNumberDescription	529
4.209.2.117	SetUsedCPUNumberValue	530
4.209.2.118	SetNumberOfOtherJobResults	530
4.209.2.119	SetOtherJobResultName	530
4.209.2.120	SetOtherJobResultValue	530
4.209.2.121	SetOtherJobResultDescription	531
4.209.2.122	SetVariableNumber	531
4.209.2.123	SetObjectiveNumber	531
4.209.2.124	SetConstraintNumber	531
4.209.2.125	SetSolutionNumber	532
4.209.2.126	SetSolutionStatus	532
4.209.2.127	SetSolutionStatusType	532
4.209.2.128	SetNumberOfSolutionSubstatuses	533
4.209.2.129	SetSolutionStatusDescription	533
4.209.2.130	SetSolutionSubstatusType	533
4.209.2.131	SetSolutionSubstatusDescription	533
4.209.2.132	SetSolutionTargetObjectiveIdx	534
4.209.2.133	SetSolutionTargetObjectiveName	534
4.209.2.134	SetSolutionWeightedObjectives	534
4.209.2.135	SetSolutionMessage	535
4.209.2.136	SetNumberOfPrimalVariableValues	535
4.209.2.137	SetPrimalVariableValuesSparse	535
4.209.2.138	SetPrimalVariableValuesDense	536
4.209.2.139	SetNumberOfVarValues	536
4.209.2.140	SetVarValue	537
4.209.2.141	SetNumberOfVarValuesString	538
4.209.2.142	SetVarValueString	538
4.209.2.143	SetBasisStatus	539
4.209.2.144	SetNumberOfOtherVariableResults	539
4.209.2.145	SetAnOtherVariableResultSparse	540
4.209.2.146	SetAnOtherVariableResultSparse	541
4.209.2.147	SetAnOtherVariableResultDense	542
4.209.2.148	SetAnOtherVariableResultDense	542
4.209.2.149	SetOtherVariableResultNumberOfVar	543
4.209.2.150	SetOtherVariableResultNumberOfEnumerations	543

4.209.2.154	SetOtherVariableResultName	543
4.209.2.155	SetOtherVariableResultType	544
4.209.2.156	SetOtherVariableResultVarType	544
4.209.2.157	SetOtherVariableResultEnumType	545
4.209.2.158	SetOtherVariableResultValue	545
4.209.2.159	SetOtherVariableResultDescription	546
4.209.2.160	SetOtherVariableResultSolver	546
4.209.2.161	SetOtherVariableResultCategory	547
4.209.2.162	SetOtherVariableResultVarIdx	548
4.209.2.163	SetOtherVariableResultVarName	548
4.209.2.164	SetOtherVariableResultVar	549
4.209.2.165	SetOtherOptionOrResultEnumeration	549
4.209.2.166	SetNumberOfOtherObjectiveResults	550
4.209.2.167	SetNumberOfObjValues	550
4.209.2.168	SetNumberOfObjectiveValues	551
4.209.2.169	SetObjectiveValuesSparse	551
4.209.2.170	SetObjectiveValuesDense	552
4.209.2.171	SetObjValue	552
4.209.2.172	SetOtherObjectiveResultNumberOfObj	552
4.209.2.173	SetOtherObjectiveResultNumberOfEnumerations	553
4.209.2.174	SetOtherObjectiveResultName	553
4.209.2.175	SetOtherObjectiveResultType	554
4.209.2.176	SetOtherObjectiveResultObjType	554
4.209.2.177	SetOtherObjectiveResultEnumType	555
4.209.2.178	SetOtherObjectiveResultValue	555
4.209.2.179	SetOtherObjectiveResultDescription	556
4.209.2.180	SetOtherObjectiveResultSolver	557
4.209.2.181	SetOtherObjectiveResultCategory	557
4.209.2.182	SetOtherObjectiveResultObjIdx	558
4.209.2.183	SetOtherObjectiveResultObjName	558
4.209.2.184	SetOtherObjectiveResultObj	559
4.209.2.185	SetNumberOfOtherConstraintResults	559
4.209.2.186	SetNumberOfDualValues	560
4.209.2.187	SetNumberOfDualVariableValues	560
4.209.2.188	SetDualVariableValuesSparse	560
4.209.2.189	SetDualVariableValuesDense	561
4.209.2.190	SetConstraintValuesDense	561

4.209.2.189	SetDualValue	562
4.209.2.189	SetOtherConstraintResultNumberOfCon	563
4.209.2.189	SetOtherConstraintResultNumberOfEnumerations	563
4.209.2.189	SetOtherConstraintResultName	564
4.209.2.189	SetOtherConstraintResultType	564
4.209.2.189	SetOtherConstraintResultConType	565
4.209.2.189	SetOtherConstraintResultEnumType	565
4.209.2.189	SetOtherConstraintResultValue	565
4.209.2.189	SetOtherConstraintResultDescription	566
4.209.2.189	SetOtherConstraintResultSolver	566
4.209.2.189	SetOtherConstraintResultCategory	567
4.209.2.189	SetOtherConstraintResultConIdx	567
4.209.2.200	SetOtherConstraintResultConName	568
4.209.2.200	SetOtherConstraintResultCon	568
4.209.2.200	SetMatrixVariableSolution	568
4.209.2.200	SetMatrixVarValuesAttributes	569
4.209.2.200	SetMatrixVarValuesBlockStructure	569
4.209.2.200	SetMatrixVarValuesBlockElements	570
4.209.2.200	SetMatrixVariablesOtherResultGeneralAttributes	571
4.209.2.200	SetMatrixVariablesOtherResultMatrixAttributes	571
4.209.2.200	SetMatrixVariablesOtherResultBlockStructure	572
4.209.2.200	SetMatrixVariablesOtherResultBlockElements	572
4.209.2.210	SetNumberOfOtherSolutionResults	573
4.209.2.210	SetOtherSolutionResultName	574
4.209.2.210	SetOtherSolutionResultValue	574
4.209.2.210	SetOtherSolutionResultCategory	574
4.209.2.210	SetOtherSolutionResultDescription	574
4.209.2.210	SetOtherSolutionResultNumberOfItems	575
4.209.2.210	SetOtherSolutionResultItem	575
4.209.2.210	SetAnOtherSolutionResult	575
4.209.2.210	SetNumberOfSolverOutputs	576
4.209.2.210	SetSolverOutputName	576
4.209.2.220	SetSolverOutputCategory	576
4.209.2.220	SetSolverOutputDescription	577
4.209.2.220	SetSolverOutputNumberOfItems	578
4.209.2.220	SetSolverOutputItem	578
4.210	OSrL2Gams Class Reference	578

4.210.1 Detailed Description	579
4.210.2 Constructor & Destructor Documentation	579
4.210.2.1 OSrL2Gams	579
4.210.3 Member Function Documentation	579
4.210.3.1 writeSolution	579
4.210.3.2 writeSolution	579
4.211 OSrLParserData Class Reference	579
4.211.1 Detailed Description	582
4.211.2 Member Data Documentation	582
4.211.2.1 kounter	582
4.211.2.2 numberAttributePresent	582
4.211.2.3 categoryAttribute	583
4.212 OSrLReader Class Reference	583
4.212.1 Detailed Description	583
4.212.2 Member Function Documentation	584
4.212.2.1 readOSrL	584
4.213 OSrLWriter Class Reference	585
4.213.1 Detailed Description	585
4.213.2 Member Function Documentation	585
4.213.2.1 writeOSrL	585
4.214 OSSmartPtr< T > Class Template Reference	586
4.214.1 Detailed Description	587
4.214.2 Constructor & Destructor Documentation	589
4.214.2.1 ~OSSmartPtr	589
4.214.3 Member Function Documentation	589
4.214.3.1 operator->	589
4.214.3.2 operator*	589
4.214.4 Friends And Related Function Documentation	589
4.214.4.1 operator==	589
4.214.4.2 operator==	590
4.214.4.3 operator"! =	590
4.214.4.4 operator"! =	590
4.214.4.5 GetRawPtr	590
4.214.4.6 IsValid	590
4.214.4.7 IsNull	590
4.215 OSSolverAgent Class Reference	590
4.215.1 Detailed Description	591

4.215.2 Constructor & Destructor Documentation	592
4.215.2.1 OSSolverAgent	592
4.215.3 Member Function Documentation	593
4.215.3.1 solve	593
4.215.3.2 getJobID	593
4.215.3.3 send	593
4.215.3.4 kill	593
4.215.3.5 retrieve	594
4.215.3.6 knock	594
4.215.3.7 fileUpload	594
4.216OtherConOption Class Reference	595
4.216.1 Detailed Description	595
4.216.2 Member Function Documentation	596
4.216.2.1 setRandom	596
4.216.2.2 deepCopyFrom	596
4.217OtherConResult Class Reference	596
4.217.1 Detailed Description	597
4.217.2 Member Function Documentation	597
4.217.2.1 setRandom	597
4.218OtherConstraintOption Class Reference	597
4.218.1 Detailed Description	599
4.218.2 Member Function Documentation	599
4.218.2.1 setRandom	599
4.218.2.2 deepCopyFrom	599
4.218.2.3 setCon	599
4.218.2.4 addCon	600
4.219OtherConstraintResult Class Reference	600
4.219.1 Detailed Description	601
4.219.2 Member Function Documentation	601
4.219.2.1 setRandom	601
4.220OtherMatrixVariableResult Class Reference	602
4.220.1 Detailed Description	602
4.220.2 Member Data Documentation	603
4.220.2.1 name	603
4.220.2.2 enumeration	603
4.221OtherObjectiveOption Class Reference	603
4.221.1 Detailed Description	604

4.221.2 Member Function Documentation	604
4.221.2.1 setRandom	604
4.221.2.2 deepCopyFrom	605
4.221.2.3 setObj	605
4.221.2.4 addObj	605
4.222OtherObjectiveResult Class Reference	605
4.222.1 Detailed Description	606
4.222.2 Member Function Documentation	607
4.222.2.1 setRandom	607
4.223OtherObjOption Class Reference	607
4.223.1 Detailed Description	608
4.223.2 Member Function Documentation	608
4.223.2.1 setRandom	608
4.223.2.2 deepCopyFrom	608
4.224OtherObjResult Class Reference	609
4.224.1 Detailed Description	609
4.224.2 Member Function Documentation	610
4.224.2.1 setRandom	610
4.225OtherOption Class Reference	610
4.225.1 Detailed Description	611
4.225.2 Member Function Documentation	611
4.225.2.1 setRandom	611
4.225.2.2 deepCopyFrom	611
4.226OtherOptionOrResultEnumeration Class Reference	612
4.226.1 Detailed Description	612
4.226.2 Member Function Documentation	613
4.226.2.1 setRandom	613
4.226.2.2 deepCopyFrom	614
4.226.2.3 setOtherOptionOrResultEnumeration	614
4.227OtherOptions Class Reference	614
4.227.1 Detailed Description	615
4.227.2 Member Function Documentation	615
4.227.2.1 setRandom	615
4.227.2.2 deepCopyFrom	616
4.227.2.3 setOther	616
4.227.2.4 addOther	616
4.228OtherResult Class Reference	616

4.228.1 Detailed Description	617
4.228.2 Member Function Documentation	617
4.228.2.1 setRandom	617
4.229OtherResults Class Reference	618
4.229.1 Detailed Description	618
4.229.2 Member Function Documentation	619
4.229.2.1 setRandom	619
4.230OtherSolutionResult Class Reference	619
4.230.1 Detailed Description	620
4.230.2 Member Function Documentation	620
4.230.2.1 setRandom	620
4.231OtherSolutionResults Class Reference	620
4.231.1 Detailed Description	621
4.231.2 Member Function Documentation	621
4.231.2.1 setRandom	621
4.232OtherSolverOutput Class Reference	622
4.232.1 Detailed Description	622
4.232.2 Member Function Documentation	623
4.232.2.1 setRandom	623
4.233OtherVariableOption Class Reference	623
4.233.1 Detailed Description	624
4.233.2 Member Function Documentation	624
4.233.2.1 setRandom	624
4.233.2.2 deepCopyFrom	624
4.233.2.3 setVar	625
4.233.2.4 addVar	625
4.234OtherVariableResult Class Reference	625
4.234.1 Detailed Description	626
4.234.2 Member Function Documentation	626
4.234.2.1 setRandom	626
4.235OtherVariableResultStruct Struct Reference	627
4.235.1 Detailed Description	627
4.235.2 Member Data Documentation	627
4.235.2.1 otherVarText	627
4.235.2.2 otherVarIndex	628
4.236OtherVarOption Class Reference	628
4.236.1 Detailed Description	629

4.236.2 Member Function Documentation	629
4.236.2.1 setRandom	629
4.236.2.2 deepCopyFrom	629
4.237OtherVarResult Class Reference	630
4.237.1 Detailed Description	630
4.237.2 Member Function Documentation	631
4.237.2.1 setRandom	631
4.238PathPair Class Reference	632
4.238.1 Detailed Description	632
4.238.2 Member Function Documentation	633
4.238.2.1 setRandom	633
4.238.2.2 deepCopyFrom	633
4.239PathPairs Class Reference	633
4.239.1 Detailed Description	634
4.239.2 Member Function Documentation	635
4.239.2.1 setRandom	635
4.239.2.2 deepCopyFrom	636
4.239.2.3 setPathPair	636
4.239.2.4 setPathPair	636
4.239.2.5 addPathPair	636
4.240PolarCone Class Reference	637
4.240.1 Detailed Description	637
4.240.2 Member Function Documentation	638
4.240.2.1 getConeName	638
4.240.2.2 setRandom	638
4.240.2.3 deepCopyFrom	638
4.240.3 Member Data Documentation	638
4.240.3.1 numberOfOtherIndexes	638
4.241PolyhedralCone Class Reference	638
4.241.1 Detailed Description	639
4.241.2 Member Function Documentation	639
4.241.2.1 getConeName	639
4.241.2.2 getConeInXML	640
4.241.2.3 setRandom	640
4.241.2.4 deepCopyFrom	640
4.241.3 Member Data Documentation	640
4.241.3.1 numberOfOtherIndexes	640

4.242Processes Class Reference	640
4.242.1 Detailed Description	641
4.242.2 Member Function Documentation	642
4.242.2.1 setRandom	642
4.242.2.2 deepCopyFrom	643
4.242.2.3 setProcess	643
4.242.2.4 addProcess	643
4.243ProductCone Class Reference	643
4.243.1 Detailed Description	644
4.243.2 Member Function Documentation	644
4.243.2.1 getConeName	644
4.243.2.2 getConeInXML	644
4.243.2.3 setRandom	645
4.243.2.4 deepCopyFrom	645
4.243.3 Member Data Documentation	645
4.243.3.1 numberOfOtherIndexes	645
4.244QuadraticCoefficients Class Reference	645
4.244.1 Detailed Description	646
4.245QuadraticCone Class Reference	646
4.245.1 Detailed Description	647
4.245.2 Member Function Documentation	647
4.245.2.1 getConeName	647
4.245.2.2 getConeInXML	647
4.245.2.3 setRandom	647
4.245.2.4 deepCopyFrom	648
4.245.3 Member Data Documentation	648
4.245.3.1 numberOfOtherIndexes	648
4.245.3.2 normScaleFactor	648
4.245.3.3 axisDirection	648
4.246QuadraticTerm Class Reference	649
4.246.1 Detailed Description	649
4.247QuadraticTerms Class Reference	649
4.247.1 Detailed Description	650
4.247.2 Member Data Documentation	650
4.247.2.1 rowIndexes	650
4.248RotatedQuadraticCone Class Reference	650
4.248.1 Detailed Description	651

4.248.2 Member Function Documentation	651
4.248.2.1 getConeName	651
4.248.2.2 getConeInXML	652
4.248.2.3 setRandom	652
4.248.2.4 deepCopyFrom	652
4.248.3 Member Data Documentation	652
4.248.3.1 numberOfOtherIndexes	652
4.248.3.2 normScaleFactor	652
4.248.3.3 firstAxisDirection	653
4.249ScalarExpressionTree Class Reference	653
4.249.1 Detailed Description	654
4.249.2 Member Function Documentation	654
4.249.2.1 getPrefixFromExpressionTree	654
4.249.2.2 getPostfixFromExpressionTree	654
4.249.2.3 getVariableIndicesMap	654
4.249.2.4 calculateFunction	654
4.250SemidefiniteCone Class Reference	655
4.250.1 Detailed Description	655
4.250.2 Member Function Documentation	656
4.250.2.1 getConeName	656
4.250.2.2 getConeInXML	656
4.250.2.3 setRandom	656
4.250.2.4 deepCopyFrom	656
4.250.3 Member Data Documentation	656
4.250.3.1 numberOfOtherIndexes	656
4.251ServiceOption Class Reference	657
4.251.1 Detailed Description	657
4.251.2 Member Function Documentation	658
4.251.2.1 setRandom	658
4.251.2.2 deepCopyFrom	658
4.252ServiceResult Class Reference	658
4.252.1 Detailed Description	659
4.252.2 Member Function Documentation	659
4.252.2.1 setRandom	659
4.253SolverOption Class Reference	660
4.253.1 Detailed Description	661
4.253.2 Member Function Documentation	661

4.253.2.1 setRandom	661
4.253.2.2 deepCopyFrom	661
4.254 SolverOptions Class Reference	662
4.254.1 Detailed Description	662
4.254.2 Member Function Documentation	663
4.254.2.1 setRandom	663
4.254.2.2 deepCopyFrom	663
4.254.2.3 setSolverOptions	663
4.254.2.4 addSolverOption	663
4.255 SolverOutput Class Reference	664
4.255.1 Detailed Description	664
4.255.2 Member Function Documentation	665
4.255.2.1 setRandom	665
4.256 SOSVariableBranchingWeights Class Reference	665
4.256.1 Detailed Description	666
4.256.2 Member Function Documentation	666
4.256.2.1 setRandom	666
4.256.2.2 deepCopyFrom	666
4.256.2.3 setSOS	667
4.256.2.4 addSOS	667
4.257 SOSWeights Class Reference	667
4.257.1 Detailed Description	668
4.257.2 Member Function Documentation	668
4.257.2.1 setRandom	668
4.257.2.2 deepCopyFrom	669
4.257.2.3 setVar	669
4.257.2.4 addVar	669
4.258 SparseHessianMatrix Class Reference	669
4.258.1 Detailed Description	670
4.258.2 Constructor & Destructor Documentation	670
4.258.2.1 SparseHessianMatrix	670
4.259 SparseIntVector Class Reference	670
4.259.1 Detailed Description	671
4.259.2 Constructor & Destructor Documentation	671
4.259.2.1 SparseIntVector	671
4.259.3 Member Data Documentation	671
4.259.3.1 indexes	671

4.260	SparseJacobianMatrix Class Reference	672
4.260.1	Detailed Description	672
4.260.2	Constructor & Destructor Documentation	672
4.260.2.1	SparseJacobianMatrix	672
4.261	SparseMatrix Class Reference	673
4.261.1	Detailed Description	673
4.261.2	Constructor & Destructor Documentation	674
4.261.2.1	SparseMatrix	674
4.261.3	Member Function Documentation	674
4.261.3.1	display	674
4.261.4	Member Data Documentation	674
4.261.4.1	isColumnMajor	674
4.261.4.2	indexes	674
4.262	SparseVector Class Reference	674
4.262.1	Detailed Description	675
4.262.2	Constructor & Destructor Documentation	675
4.262.2.1	SparseVector	675
4.263	StorageCapacity Class Reference	675
4.263.1	Detailed Description	676
4.263.2	Member Function Documentation	676
4.263.2.1	setRandom	676
4.263.2.2	deepCopyFrom	676
4.264	SystemOption Class Reference	677
4.264.1	Detailed Description	678
4.264.2	Member Function Documentation	678
4.264.2.1	setRandom	678
4.264.2.2	deepCopyFrom	678
4.265	SystemResult Class Reference	679
4.265.1	Detailed Description	679
4.265.2	Member Function Documentation	680
4.265.2.1	setRandom	680
4.266	TimeDomain Class Reference	680
4.266.1	Detailed Description	680
4.267	TimeDomainInterval Class Reference	681
4.267.1	Detailed Description	681
4.268	TimeDomainStage Class Reference	681
4.268.1	Detailed Description	682

4.269TimeDomainStageCon Class Reference	682
4.269.1 Detailed Description	682
4.270TimeDomainStageConstraints Class Reference	682
4.270.1 Detailed Description	683
4.271TimeDomainStageObj Class Reference	683
4.271.1 Detailed Description	683
4.272TimeDomainStageObjectives Class Reference	684
4.272.1 Detailed Description	684
4.273TimeDomainStages Class Reference	684
4.273.1 Detailed Description	685
4.274TimeDomainStageVar Class Reference	685
4.274.1 Detailed Description	685
4.275TimeDomainStageVariables Class Reference	686
4.275.1 Detailed Description	686
4.276TimeMeasurement Class Reference	686
4.276.1 Detailed Description	687
4.276.2 Member Function Documentation	687
4.276.2.1 setRandom	687
4.277TimeSpan Class Reference	687
4.277.1 Detailed Description	688
4.277.2 Member Function Documentation	688
4.277.2.1 setRandom	688
4.277.2.2 deepCopyFrom	689
4.278TimingInformation Class Reference	689
4.278.1 Detailed Description	690
4.278.2 Member Function Documentation	690
4.278.2.1 setRandom	690
4.279Variable Class Reference	690
4.279.1 Detailed Description	691
4.279.2 Member Data Documentation	691
4.279.2.1 lb	691
4.279.2.2 ub	691
4.279.2.3 type	691
4.280VariableOption Class Reference	691
4.280.1 Detailed Description	692
4.280.2 Member Function Documentation	693
4.280.2.1 setRandom	693

4.280.2.2 deepCopyFrom	693
4.280.2.3 setOther	693
4.280.2.4 addOther	693
4.281 Variables Class Reference	694
4.281.1 Detailed Description	694
4.282 VariableSolution Class Reference	694
4.282.1 Detailed Description	695
4.282.2 Member Function Documentation	695
4.282.2.1 setRandom	695
4.283 VariableValues Class Reference	696
4.283.1 Detailed Description	696
4.283.2 Member Function Documentation	697
4.283.2.1 setRandom	697
4.284 VariableValuesString Class Reference	697
4.284.1 Detailed Description	698
4.284.2 Member Function Documentation	698
4.284.2.1 setRandom	698
4.285 VarReferenceMatrixElements Class Reference	698
4.285.1 Detailed Description	699
4.285.2 Member Function Documentation	699
4.285.2.1 getNodeType	699
4.285.2.2 getMatrixType	699
4.285.2.3 getNodeName	699
4.285.2.4 getMatrixNodeInXML	700
4.285.2.5 alignsOnBlockBoundary	700
4.285.2.6 cloneMatrixNode	700
4.285.2.7 setRandom	700
4.285.2.8 deepCopyFrom	700
4.286 VarReferenceMatrixValues Class Reference	701
4.286.1 Detailed Description	701
4.286.2 Member Function Documentation	701
4.286.2.1 setRandom	701
4.286.2.2 deepCopyFrom	702
4.287 VarValue Class Reference	702
4.287.1 Detailed Description	703
4.287.2 Member Function Documentation	703
4.287.2.1 setRandom	703

4.288VarValueString Class Reference	703
4.288.1 Detailed Description	704
4.288.2 Member Function Documentation	704
4.288.2.1 setRandom	704
4.289WSUtil Class Reference	705
4.289.1 Detailed Description	705
4.289.2 Constructor & Destructor Documentation	706
4.289.2.1 WSUtil	706
4.289.3 Member Function Documentation	707
4.289.3.1 sendSOAPMessage	707
4.289.3.2 SOAPify	707
4.289.3.3 deSOAPify	707
4.289.3.4 createSOAPMessage	707
4.289.3.5 createFormDataUpload	708
4.289.3.6 getOSxL	708
4.290YYLTYPE Struct Reference	709
4.290.1 Detailed Description	709
4.291YYSTYPE Union Reference	709
4.291.1 Detailed Description	709
5 File Documentation	711
5.1 OSCommandLine.h File Reference	711
5.2 OSCommandLineReader.h File Reference	711
5.2.1 Detailed Description	711
5.3 OSCouenneSolver.h File Reference	712
5.3.1 Detailed Description	712
5.4 OSCsdpSolver.h File Reference	712
5.4.1 Detailed Description	713
5.5 OSExpressionTree.h File Reference	713
5.5.1 Detailed Description	714
5.6 OSGeneral.h File Reference	714
5.6.1 Detailed Description	715
5.7 OSgLParseData.h File Reference	715
5.7.1 Detailed Description	716
5.8 OSgLWriter.h File Reference	716
5.8.1 Detailed Description	716
5.8.2 Function Documentation	717

5.8.2.1	writeIntVectorData	. 717
5.8.2.2	writeGeneralFileHeader	. 717
5.8.2.3	writeOtherOptionOrResultEnumeration	. 718
5.8.2.4	writeDblVectorData	. 718
5.8.2.5	writeBasisStatus	. 718
5.9	OShL.h File Reference	. 718
5.9.1	Detailed Description	. 719
5.10	OSiLParserData.h File Reference	. 719
5.10.1	Detailed Description	. 719
5.11	OSiLReader.h File Reference	. 719
5.11.1	Detailed Description	. 720
5.12	OSiLWriter.h File Reference	. 720
5.12.1	Detailed Description	. 720
5.13	OSInstance.h File Reference	. 720
5.13.1	Detailed Description	. 723
5.14	OSMatrix.h File Reference	. 723
5.14.1	Detailed Description	. 725
5.15	OSmps2OS.h File Reference	. 725
5.15.1	Detailed Description	. 726
5.16	OSmps2osil.h File Reference	. 726
5.16.1	Detailed Description	. 726
5.17	OSnLNode.h File Reference	. 726
5.17.1	Detailed Description	. 729
5.18	OSnLParserData.h File Reference	. 729
5.18.1	Detailed Description	. 729
5.19	OSoLParserData.h File Reference	. 729
5.19.1	Detailed Description	. 730
5.20	OSoLReader.h File Reference	. 730
5.20.1	Detailed Description	. 730
5.21	OSoLWriter.h File Reference	. 730
5.21.1	Detailed Description	. 731
5.22	OSOption.h File Reference	. 731
5.22.1	Detailed Description	. 733
5.23	OSOptionsStruc.h File Reference	. 733
5.23.1	Detailed Description	. 734
5.24	OSosrl2ampl.h File Reference	. 734
5.24.1	Detailed Description	. 734

5.25 OSOutput.h File Reference	734
5.25.1 Detailed Description	735
5.26 OSParameters.h File Reference	735
5.26.1 Detailed Description	736
5.26.2 Enumeration Type Documentation	736
5.26.2.1 ENUM_OUTPUT_LEVEL	736
5.26.2.2 ENUM_OUTPUT_AREA	736
5.26.2.3 ENUM_BASIS_STATUS	736
5.27 OSResult.h File Reference	737
5.27.1 Detailed Description	739
5.28 OSrLParserData.h File Reference	739
5.28.1 Detailed Description	739
5.29 OSrLReader.h File Reference	739
5.29.1 Detailed Description	740
5.30 OSrLWriter.h File Reference	740
5.30.1 Detailed Description	740
5.31 OSRunSolver.h File Reference	740
5.31.1 Detailed Description	741
5.31.2 Function Documentation	741
5.31.2.1 runSolver	741
5.31.2.2 runSolver	741
5.31.2.3 runSolver	742
5.31.2.4 runSolver	742
5.31.2.5 selectSolver	742
5.32 OSSolverAgent.h File Reference	743
5.32.1 Detailed Description	743
5.33 OSStringUtil.h File Reference	743
5.33.1 Detailed Description	743
5.33.2 Function Documentation	744
5.33.2.1 writeStringData	744
5.34 OSWSUtil.h File Reference	745
5.34.1 Detailed Description	745
Index	747

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

```
_EKKfactinfo [external]
AbcDualRowPivot [external]
    AbcDualRowDantzig [external]
    AbcDualRowSteepest [external]
AbcMatrix [external]
AbcMatrix2 [external]
AbcMatrix3 [external]
AbcNonLinearCost [external]
AbcPrimalColumnPivot [external]
    AbcPrimalColumnDantzig [external]
    AbcPrimalColumnSteepest [external]
AbcSimplexFactorization [external]
AbcTolerancesEtc [external]
AbcWarmStartOrganizer [external]
forcing_constraint_action::action [external]
doubleton_action::action [external]
tripleton_action::action [external]
remove_fixed_action::action [external]
std::allocator< T >
ampl_info [external]
OsiSolverInterface::ApplyCutsReturnCode [external]
std::array< T >
std::auto_ptr< T >
auxiliary_graph [external]
CbcGenCtlBlk::babState_struct [external]
Base64 ..... 41
std::basic_string< Char >
    std::string
    std::wstring
std::basic_string< char >
std::basic_string< wchar_t >
BasisStatus ..... 45
BCP_buffer [external]
BCP_cg_par [external]
```

```

    BCP_parameter_set< BCP_cg_par >[external]
BCP_fatal_error[external]
BCP_internal_brobj[external]
BCP_lp_branching_object[external]
BCP_lp_node[external]
BCP_lp_par[external]
    BCP_parameter_set< BCP_lp_par >[external]
BCP_lp_parent[external]
BCP_lp_result[external]
BCP_lp_statistics[external]
BCP_lp_waiting_col[external]
BCP_lp_waiting_row[external]
BCP_MemPool[external]
BCP_message_environment[external]
    BCP_single_environment[external]
BCP_node_storage_in_tm[external]
BCP_obj_change[external]
BCP_obj_set_change[external]
BCP_parameter[external]
BCP_parameter_set< Par >[external]
BCP_presolved_lp_brobj[external]
BCP_problem_core[external]
BCP_problem_core_change[external]
BCP_process[external]
    BCP_cg_prob[external]
    BCP_lp_prob[external]
    BCP_tm_prob[external]
    BCP_ts_prob[external]
    BCP_vg_prob[external]
BCP_scheduler[external]
BCP_slave_params[external]
BCP_solution[external]
    BCP_solution_generic[external]
BCP_string[external]
BCP_tm_flags[external]
BCP_tm_node_data[external]
BCP_tm_node_to_send[external]
BCP_tm_par[external]
    BCP_parameter_set< BCP_tm_par >[external]
BCP_tm_stat[external]
BCP_tree[external]
BCP_ts_node_data[external]
BCP_ts_par[external]
    BCP_parameter_set< BCP_ts_par >[external]
BCP_user_class[external]
    BCP_cg_user[external]
    BCP_lp_user[external]
    BCP_tm_user[external]
    BCP_ts_user[external]
    BCP_vg_user[external]
BCP_user_pack[external]
BCP_vec< T >[external]
BCP_vec< BCP_child_action >[external]
BCP_vec< BCP_col * >[external]

```

```

BCP_vec< BCP_cut * >[external]
    BCP_cut_set[external]
BCP_vec< BCP_cut_core * >[external]
BCP_vec< BCP_lp_result * >[external]
BCP_vec< BCP_lp_waiting_col * >[external]
    BCP_lp_var_pool[external]
BCP_vec< BCP_lp_waiting_row * >[external]
    BCP_lp_cut_pool[external]
BCP_vec< BCP_obj_change >[external]
BCP_vec< BCP_row * >[external]
BCP_vec< BCP_string >[external]
BCP_vec< BCP_tm_node * >[external]
BCP_vec< BCP_user_data * >[external]
BCP_vec< BCP_var * >[external]
    BCP_var_set[external]
BCP_vec< BCP_var_core * >[external]
BCP_vec< BCP_vec< BCP_cut * > >[external]
BCP_vec< BCP_vec< BCP_row * > >[external]
BCP_vec< char >[external]
BCP_vec< Coin::SmartPtr< BCP_cut > >[external]
BCP_vec< Coin::SmartPtr< BCP_var > >[external]
BCP_vec< double >[external]
BCP_vec< int >[external]
BCP_vec< std::pair< BCP_string, BCP_parameter > >[external]
BCP_vec< std::pair< int, int > >[external]
BCP_vec_change< T >[external]
BCP_vec_change< char >[external]
BCP_vec_change< double >[external]
BCP_vg_par[external]
    BCP_parameter_set< BCP_vg_par >[external]
BCP_warmstart[external]
    BCP_warmstart_basis[external]
    BCP_warmstart_dual[external]
    BCP_warmstart_primaldual[external]
std::bitset< Bits >
BitVector128[external]
blockStruct[external]
blockStruct3[external]

BranchingWeight ..... 52
ClpNode::branchState[external]
Ipopt::CachedResults< T >[external]
CachedResults< Ipopt::SmartPtr< const Ipopt::Matrix > >[external]
CachedResults< Ipopt::SmartPtr< const Ipopt::SymMatrix > >[external]
CachedResults< Ipopt::SmartPtr< const Ipopt::Vector > >[external]
CachedResults< Ipopt::SmartPtr< Ipopt::Vector > >[external]
CachedResults< Number >[external]
CachedResults< void * >[external]
CbcBaseModel[external]
CbcBranchDecision[external]
    CbcBranchDefaultDecision[external]
    CbcBranchDynamicDecision[external]
CbcCompare[external]
CbcCompareBase[external]
    CbcCompareDefault[external]

```

- CbcCompareDepth [external]
- CbcCompareEstimate [external]
- CbcCompareObjective [external]
- CbcConsequence [external]
- CbcFixVariable [external]
- CbcCutGenerator [external]
- CbcCutModifier [external]
 - CbcCutSubsetModifier [external]
- CbcEventHandler [external]
- CbcFathom [external]
 - CbcFathomDynamicProgramming [external]
- CbcFeasibilityBase [external]
- CbcGenCtlBlk [external]
- CbcHeuristic [external]
 - CbcHeuristicCrossover [external]
 - CbcHeuristicDINS [external]
 - CbcHeuristicDive [external]
 - CbcHeuristicDiveCoefficient [external]
 - CbcHeuristicDiveFractional [external]
 - CbcHeuristicDiveGuided [external]
 - CbcHeuristicDiveLineSearch [external]
 - CbcHeuristicDivePseudoCost [external]
 - CbcHeuristicDiveVectorLength [external]
 - CbcHeuristicDW [external]
 - CbcHeuristicDynamic3 [external]
 - CbcHeuristicFPump [external]
 - CbcHeuristicGreedyCover [external]
 - CbcHeuristicGreedyEquality [external]
 - CbcHeuristicGreedySOS [external]
 - CbcHeuristicJustOne [external]
 - CbcHeuristicLocal [external]
 - CbcHeuristicNaive [external]
 - CbcHeuristicPartial [external]
 - CbcHeuristicPivotAndFix [external]
 - CbcHeuristicProximity [external]
 - CbcHeuristicRandRound [external]
 - CbcHeuristicRENS [external]
 - CbcHeuristicRINS [external]
 - CbcHeuristicVND [external]
 - CbcRounding [external]
 - CbcSerendipity [external]
- CbcHeuristicNode [external]
- CbcHeuristicNodeList [external]
- CbcModel [external]
- CbcNauty [external]
- CbcNodeInfo [external]
 - CbcFullNodeInfo [external]
 - CbcPartialNodeInfo [external]
- CbcObjectUpdateData [external]
- CbcOrClpParam [external]
- CbcParam [external]
- CbcGenCtlBlk::cbcParamsInfo_struct [external]
- CbcRowCuts [external]
- CbcSolver [external]

- CbcSolverUsefulData [external]
- CbcSolverUsefulData2 [external]
- CbcStatistics [external]
- CbcStopNow [external]
- CbcStrategy [external]
 - CbcStrategyDefault [external]
 - CbcStrategyDefaultSubTree [external]
 - CbcStrategyNull [external]
- CbcStrongInfo [external]
- CbcSymmetry [external]
- CbcThread [external]
- CbcTree [external]
 - CbcTreeLocal [external]
 - CbcTreeVariable [external]
- CbcUser [external]
- Cgl012Cut [external]
- cgl_arc [external]
- cgl_graph [external]
- cgl_node [external]
- CglBK [external]
- CglCutGenerator [external]
 - CglAllDifferent [external]
 - CglClique [external]
 - CglFakeClique [external]
 - CglDuplicateRow [external]
 - CglFlowCover [external]
 - CglGMI [external]
 - CglGomory [external]
 - CglImplication [external]
 - CglKnapsackCover [external]
 - CglLandP [external]
 - CglLiftAndProject [external]
 - CglMixedIntegerRounding [external]
 - CglMixedIntegerRounding2 [external]
 - CglOddHole [external]
 - CglProbing [external]
 - CglRedSplit [external]
 - CglRedSplit2 [external]
 - CglResidualCapacity [external]
 - CglSimpleRounding [external]
 - CglStored [external]
 - CglTemporary [external]
 - CglTwomir [external]
 - CglZeroHalf [external]
- CglFlowVUB [external]
- CglHashLink [external]
- LAP::CglLandPSimplex [external]
- CglMixIntRoundVUB [external]
- CglMixIntRoundVUB2 [external]
- CglParam [external]
 - CglGMIParam [external]
 - CglLandP::Parameters [external]
 - CglRedSplit2Param [external]
 - CglRedSplitParam [external]

- CglPreProcess [external]
- CglTreeInfo [external]
 - CglTreeProbingInfo [external]
- CglUniqueRowCuts [external]
- CbcGenCtlBlk::chooseStrongCtl_struct [external]
- CliqueEntry [external]
- CglProbing::CliqueType [external]
- ClpCholeskyBase [external]
 - ClpCholeskyDense [external]
 - ClpCholeskyMumps [external]
 - ClpCholeskyTaucs [external]
 - ClpCholeskyUfl [external]
 - ClpCholeskyWssmp [external]
 - ClpCholeskyWssmpKKT [external]
- ClpCholeskyDenseC [external]
- ClpConstraint [external]
 - ClpConstraintAmpl [external]
 - ClpConstraintLinear [external]
 - ClpConstraintQuadratic [external]
- ClpDataSave [external]
- ClpDisasterHandler [external]
 - OsiClpDisasterHandler [external]
- ClpDualRowPivot [external]
 - ClpDualRowDantzig [external]
 - ClpDualRowSteepest [external]
- ClpEventHandler [external]
 - MyEventHandler [external]
- ClpFactorization [external]
- ClpHashValue [external]
- ClpLsq [external]
- ClpMatrixBase [external]
 - ClpDummyMatrix [external]
 - ClpNetworkMatrix [external]
 - ClpPackedMatrix [external]
 - ClpDynamicMatrix [external]
 - ClpDynamicExampleMatrix [external]
 - ClpGubMatrix [external]
 - ClpGubDynamicMatrix [external]
 - ClpPlusMinusOneMatrix [external]
- ClpModel [external]
 - ClpInterior [external]
 - ClpPdco [external]
 - ClpPredictorCorrector [external]
 - ClpSimplex [external]
 - AbcSimplex [external]
 - AbcSimplexDual [external]
 - AbcSimplexPrimal [external]
 - ClpSimplexDual [external]
 - ClpSimplexOther [external]
 - ClpSimplexPrimal [external]
 - ClpSimplexNonlinear [external]
- ClpNetworkBasis [external]
- ClpNode [external]
- ClpNodeStuff [external]

```

ClpNonLinearCost [external]
ClpObjective [external]
    ClpAmplObjective [external]
    ClpLinearObjective [external]
    ClpQuadraticObjective [external]
ClpPackedMatrix2 [external]
ClpPackedMatrix3 [external]
ClpPdcoBase [external]
ClpPresolve [external]
ClpPrimalColumnPivot [external]
    ClpPrimalColumnDantzig [external]
    ClpPrimalColumnSteepest [external]
    ClpPrimalQuadraticDantzig [external]
ClpSimplexProgress [external]
ClpSolve [external]
ClpTrustedData [external]
CoinAbcAnyFactorization [external]
    CoinAbcDenseFactorization [external]
    CoinAbcTypeFactorization [external]
CoinAbcStack [external]
CoinAbcStatistics [external]
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]
    CgILandP::NoBasisError [external]
    CgILandP::SimplexInterfaceError [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]
- ClpHashValue::CoinHashLink[external]
- CoinLpIO::CoinHashLink[external]
- CoinHashLink[external]
- CoinMpsIO::CoinHashLink[external]
- CoinIndexedVector[external]
 - CoinPartitionedVector[external]
 - LAP::TabRow[external]
- CoinLpIO[external]
- CoinMessageHandler[external]
 - MyMessageHandler[external]
- CoinMessages[external]
 - CbcMessage[external]
 - CglMessage[external]
 - ClpMessage[external]
 - CoinMessage[external]
 - LAP::LandPMessages[external]
 - LAP::LapMessages[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]
 - CoinOsiFactorization[external]
 - CoinSimpFactorization[external]
- CoinPackedMatrix[external]
 - BCP_lp_relax[external]
- CoinPackedVectorBase[external]
 - CoinPackedVector[external]
 - BCP_col[external]
 - BCP_row[external]
 - CoinShallowPackedVector[external]
- CoinPair< S, T >[external]
- CoinParam[external]
 - CbcCbcParam[external]
 - CbcGenParam[external]
 - CbcOsiParam[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]


```

duprow3_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]
    BCP_tm_node[external]
    CbcNode[external]
CoinTreeSiblings[external]
CoinTriple< S, T, U >[external]
CoinWarmStart[external]
    CoinWarmStartBasis[external]
        AbcWarmStart[external]
    CoinWarmStartDual[external]
    CoinWarmStartPrimalDual[external]
    CoinWarmStartVector< T >[external]
    CoinWarmStartVector< double >[external]
    CoinWarmStartVector< U >[external]
    CoinWarmStartVectorPair< T, U >[external]
CoinWarmStartDiff[external]
    CoinWarmStartBasisDiff[external]
    CoinWarmStartDualDiff[external]
    CoinWarmStartPrimalDualDiff[external]
    CoinWarmStartVectorDiff< T >[external]
    CoinWarmStartVectorDiff< double >[external]
    CoinWarmStartVectorDiff< U >[external]
    CoinWarmStartVectorPairDiff< T, U >[external]
CoinYacc[external]
std::complex

```

Cone	58
CompletelyPositiveMatricesCone	56
CopositiveMatricesCone	78
DualCone	90
IntersectionCone	158
NonnegativeCone	239
NonpositiveCone	240
OrthantCone	262
PolarCone	637
PolyhedralCone	638
ProductCone	643
QuadraticCone	646
RotatedQuadraticCone	650
SemidefiniteCone	655
Cones	60
ConReferenceMatrixElement	62
OsiCuts::const_iterator[external]	
std::basic_string< Char >::const_iterator	
std::string::const_iterator	
std::wstring::const_iterator	
std::deque< T >::const_iterator	
std::list< T >::const_iterator	
std::forward_list< T >::const_iterator	
std::map< K, T >::const_iterator	
std::unordered_map< K, T >::const_iterator	
std::multimap< K, T >::const_iterator	
std::unordered_multimap< K, T >::const_iterator	
std::set< K >::const_iterator	
std::unordered_set< K >::const_iterator	
std::multiset< K >::const_iterator	
std::unordered_multiset< K >::const_iterator	
std::vector< T >::const_iterator	
std::basic_string< Char >::const_reverse_iterator	
std::string::const_reverse_iterator	
std::wstring::const_reverse_iterator	
std::deque< T >::const_reverse_iterator	
std::list< T >::const_reverse_iterator	
std::forward_list< T >::const_reverse_iterator	
std::map< K, T >::const_reverse_iterator	
std::unordered_map< K, T >::const_reverse_iterator	
std::multimap< K, T >::const_reverse_iterator	
std::unordered_multimap< K, T >::const_reverse_iterator	
std::set< K >::const_reverse_iterator	
std::unordered_set< K >::const_reverse_iterator	
std::multiset< K >::const_reverse_iterator	
std::unordered_multiset< K >::const_reverse_iterator	
std::vector< T >::const_reverse_iterator	
Constraint	72
ConstraintOption	73
Constraints	75
ConstraintSolution	76
ContactOption	77
CPUNumber	81
CPU Speed	83

cut[external]	
cut_list[external]	
cutParams[external]	
LAP::Cuts[external]	
cycle[external]	
cycle_list[external]	
CbcGenCtlBlk::debugSolInfo_struct[external]	
DefaultSolver	86
BonminSolver	51
CoinSolver	54
CouenneSolver	80
CsdpSolver	85
IpoptSolver	165
KnitroSolver	173
LindoSolver	175
std::deque< T >	
std::deque< StdVectorDouble >	
DGG_constraint_t[external]	
DGG_data_t[external]	
DGG_list_t[external]	
DirectoriesAndFiles	88
disaggregationAction[external]	
CbcGenCtlBlk::djFixCtl_struct[external]	
DoubleVector	89
dropped_zero[external]	
dualColumnResult[external]	
DualVariableValues	92
DualVarValue	93
edge[external]	
EKKHlink[external]	
std::error_category	
std::error_code	
std::error_condition	
ErrorClass	94
std::exception	
std::bad_alloc	
std::bad_cast	
std::bad_exception	
std::bad_typeid	
std::ios_base::failure	
std::logic_error	
std::domain_error	
std::invalid_argument	
std::length_error	
std::out_of_range	
std::runtime_error	
std::overflow_error	
std::range_error	
std::underflow_error	
ExpandedMatrixBlocks	95
ExprNode	98
OSnLMNode	352
OSnLMNodeDiagonalMatrixFromVector	355
OSnLMNodeIdentityMatrix	356

OSnLMNodeMatrixCon	357
OSnLMNodeMatrixDiagonal	358
OSnLMNodeMatrixDotTimes	359
OSnLMNodeMatrixInverse	360
OSnLMNodeMatrixLowerTriangle	361
OSnLMNodeMatrixMinus	362
OSnLMNodeMatrixNegate	363
OSnLMNodeMatrixObj	364
OSnLMNodeMatrixPlus	366
OSnLMNodeMatrixProduct	366
OSnLMNodeMatrixReference	368
OSnLMNodeMatrixScalarTimes	369
OSnLMNodeMatrixSubmatrixAt	370
OSnLMNodeMatrixSum	371
OSnLMNodeMatrixTimes	372
OSnLMNodeMatrixTranspose	373
OSnLMNodeMatrixUpperTriangle	374
OSnLMNodeMatrixVar	375
OSnLNode	377
OSnLNodeAbs	381
OSnLNodeAllDiff	383
OSnLNodeCos	385
OSnLNodeDivide	387
OSnLNodeE	389
OSnLNodeErf	391
OSnLNodeExp	393
OSnLNodeIf	395
OSnLNodeLn	397
OSnLNodeMatrixDeterminant	399
OSnLNodeMatrixToScalar	401
OSnLNodeMatrixTrace	403
OSnLNodeMax	405
OSnLNodeMin	407
OSnLNodeMinus	409
OSnLNodeNegate	411
OSnLNodeNumber	413
OSnLNodePI	415
OSnLNodePlus	418
OSnLNodePower	419
OSnLNodeProduct	421
OSnLNodeSin	423
OSnLNodeSqrt	425
OSnLNodeSquare	427
OSnLNodeSum	429
OSnLNodeTimes	431
OSnLNodeVariable	433
FactorPointers[external]	
FileUtil	102
Ipopt::Filter[external]	
Ipopt::FilterEntry[external]	
std::forward_list< T >	
GeneralFileHeader	104
GeneralOption	110
GeneralResult	112

GeneralSparseMatrix	113
GeneralStatus	116
GeneralSubstatus	117
CbcGenCtIBlk::genParamsInfo_struct[external]	
glp_prob[external]	
Idiot[external]	
IdiotResult[external]	
ilp[external]	
IndexStringPair	118
IndexValuePair	119
Info[external]	
info_weak[external]	
InitBasStatus	119
InitConstraintValues	121
InitConValue	124
InitDualVariableValues	125
InitDualVarValue	128
InitialBasisStatus	130
InitObjBound	132
InitObjectiveBounds	134
InitObjectiveValues	136
InitObjValue	140
InitVariableValues	142
InitVariableValuesString	144
InitVarValue	148
InitVarValueString	150
InstanceData	151
InstanceLocationOption	153
IntegerVariableBranchingWeights	154
Interval	161
IntVector	161
OtherOptionOrResultEnumeration	612
std::ios_base	
basic_ios< char >	
basic_ios< wchar_t >	
std::basic_ios	
basic_istream< char >	
basic_istream< wchar_t >	
basic_ostream< char >	
basic_ostream< wchar_t >	
std::basic_istream	
basic_ifstream< char >	
basic_ifstream< wchar_t >	
basic_iostream< char >	
basic_iostream< wchar_t >	
basic_istreamstream< char >	
basic_istreamstream< wchar_t >	
std::basic_ifstream	
std::ifstream	
std::wifstream	
std::basic_iostream	
basic_fstream< char >	
basic_fstream< wchar_t >	
basic_stringstream< char >	

basic_stringstream< wchar_t >	
std::basic_fstream	
std::fstream	
std::wfstream	
std::basic_stringstream	
std::stringstream	
std::wstringstream	
std::basic_istream	
std::istream	
std::wistream	
std::basic_ostream	
basic_iostream< char >	
basic_iostream< wchar_t >	
basic_ofstream< char >	
basic_ofstream< wchar_t >	
basic_ostringstream< char >	
basic_ostringstream< wchar_t >	
std::basic_iostream	
std::basic_ofstream	
std::ofstream	
std::wofstream	
std::basic_ostringstream	
std::ostringstream	
std::wostringstream	
std::ostream	
std::wostream	
std::ios	
std::wios	
Ipopt::IpoptException [external]	
OsiCuts::iterator [external]	
std::unordered_multimap< K, T >::iterator	
std::unordered_set< K >::iterator	
std::basic_string< Char >::iterator	
std::string::iterator	
std::wstring::iterator	
std::deque< T >::iterator	
std::list< T >::iterator	
std::map< K, T >::iterator	
std::unordered_map< K, T >::iterator	
std::multimap< K, T >::iterator	
std::set< K >::iterator	
std::multiset< K >::iterator	
std::forward_list< T >::iterator	
std::unordered_multiset< K >::iterator	
std::vector< T >::iterator	
JobDependencies	166
JobOption	169
JobResult	171
LinearConstraintCoefficients	178
LinearMatrixElement	179
LinearMatrixElementTerm	183
std::list< T >	

log_var[external]	
ma77_control_d[external]	
ma77_info_d[external]	
ma86_control_d[external]	
ma86_info_d[external]	
ma97_control_d[external]	
ma97_info[external]	
std::map< K, T >	
std::map< int, BCP_tm_node * >	
std::map< int, int >	
std::map< int, MatrixExpressionTree * >	
std::map< int, ScalarExpressionTree * >	
std::map< int, std::vector< OSnLNode * > >	
MathUtil	185
Matrices	187
MatrixCon	196
MatrixConstraints	197
MatrixConstraintSolution	197
MatrixElementValues	200
ConReferenceMatrixValues	66
ConstantMatrixValues	70
GeneralMatrixValues	108
LinearMatrixValues	184
ObjReferenceMatrixValues	251
VarReferenceMatrixValues	701
MatrixExpression	201
MatrixExpressions	202
MatrixNode	204
MatrixConstructor	198
BaseMatrix	42
MatrixBlocks	192
MatrixElements	199
ConReferenceMatrixElements	63
ConstantMatrixElements	67
GeneralMatrixElements	106
LinearMatrixElements	180
MixedRowReferenceMatrixElements	234
ObjReferenceMatrixElements	249
VarReferenceMatrixElements	698
MatrixTransformation	214
MatrixType	217
MatrixBlock	188
OSMatrix	337
OSMatrixWithMatrixConIdx	341
OSMatrixWithMatrixObjIdx	342
OSMatrixWithMatrixVarIdx	344
MatrixObj	208
MatrixObjectives	209
MatrixObjectiveSolution	210
MatrixProgramming	211
MatrixProgrammingSolution	213
MatrixVar	225
MatrixVariables	227

MatrixVariableSolution	227
MatrixVariableValues	228
MaxTime	229
mc68_control[external]	
mc68_info[external]	
MinCPUNumber	230
MinCPUSpeed	231
MinDiskSpace	232
MinMemorySize	233
std::multimap< K, T >	
std::multiset< K >	
Nl	237
NlpProblemDef	
KnitroProblem	173
NonlinearExpressions	238
ObjCoef	242
Objective	243
ObjectiveOption	244
Objectives	246
ObjectiveSolution	246
ObjectiveValues	248
ObjValue	253
Ipopt::Observer[external]	
DependentResult< Ipopt::SmartPtr< const Ipopt::Matrix > >[external]	
DependentResult< Ipopt::SmartPtr< const Ipopt::SymMatrix > >[external]	
DependentResult< Ipopt::SmartPtr< const Ipopt::Vector > >[external]	
DependentResult< Ipopt::SmartPtr< Ipopt::Vector > >[external]	
DependentResult< Number >[external]	
DependentResult< void * >[external]	
Ipopt::DependentResult< T >[external]	
OptimizationOption	255
OptimizationResult	257
OptimizationSolution	258
OptimizationSolutionStatus	260
OptimizationSolutionSubstatus	261
Options[external]	
OS_AMPL_SUFFIX	264
OSCommandLine	264
OSCommandLineReader	268
OSExpressionTree	270
MatrixExpressionTree	203
ScalarExpressionTree	653
OSgams2osil	271
OSGeneral	272
OSgLParserData	272
OShL	274
OSSolverAgent	590
OsiAuxInfo[external]	
OsiBabSolver[external]	
OsiBranchingInformation[external]	
OsiBranchingObject[external]	
CbcBranchingObject[external]	
CbcCliquesBranchingObject[external]	
CbcCutBranchingObject[external]	

CbcDummyBranchingObject [external]	
CbcFixingBranchingObject [external]	
CbcIntegerBranchingObject [external]	
CbcDynamicPseudoCostBranchingObject [external]	
CbcIntegerPseudoCostBranchingObject [external]	
CbcLongCliqueBranchingObject [external]	
CbcLotsizeBranchingObject [external]	
CbcNWayBranchingObject [external]	
CbcOrbitalBranchingObject [external]	
CbcSOSBranchingObject [external]	
OsiTwoWayBranchingObject [external]	
OsiBiLinearBranchingObject [external]	
OsiIntegerBranchingObject [external]	
BCP_lp_integer_branching_object [external]	
OsiLinkBranchingObject [external]	
OsiLotsizeBranchingObject [external]	
OsiSOSBranchingObject [external]	
BCP_lp_sos_branching_object [external]	
OsiOldLinkBranchingObject [external]	
OsiChooseVariable [external]	
OsiChooseStrong [external]	
OsiChooseStrongSubset [external]	
OsiCut [external]	
OsiColCut [external]	
OsiRowCut [external]	
CbcCountRowCut [external]	
OsiRowCut2 [external]	
OsiCuts [external]	
OsiHotInfo [external]	
OsiLinkedBound [external]	
OSILParserData	277
OSILReader	280
OSILWriter	281
OSInstance	282
OsiObject [external]	
CbcObject [external]	
CbcBranchCut [external]	
CbcBranchAllDifferent [external]	
CbcBranchToFixLots [external]	
CbcClique [external]	
CbcFollowOn [external]	
CbcGeneral [external]	
CbcIdiotBranch [external]	
CbcLotsize [external]	
CbcNWay [external]	
CbcSimpleInteger [external]	
CbcSimpleIntegerDynamicPseudoCost [external]	
CbcSimpleIntegerPseudoCost [external]	
CbcSOS [external]	
OsiObject2 [external]	
OsiBiLinear [external]	
OsiBiLinearEquality [external]	
OsiLotsize [external]	
OsiSimpleInteger [external]	

OsiSimpleFixedInteger[external]	
OsiUsesBiLinear[external]	
OsiSOS[external]	
OsiLink[external]	
OsiOldLink[external]	
OsiOneLink[external]	
CbcGenCtlBlk::osiParamsInfo_struct[external]	
OsiPresolve[external]	
OsiPseudoCosts[external]	
OsiRowCutDebugger[external]	
OsiSolverBranch[external]	
OsiSolverInterface[external]	
OsiCbcSolverInterface[external]	
OsiClpSolverInterface[external]	
CbcOsiSolver[external]	
OsiSolverLink[external]	
OsiSolverLinearizedQuadratic[external]	
OsiCpxSolverInterface[external]	
OsiGlpkSolverInterface[external]	
OsiGrbSolverInterface[external]	
OsiMskSolverInterface[external]	
OsiSpxSolverInterface[external]	
OsiVolSolverInterface[external]	
OsiXprSolverInterface[external]	
OsiSolverResult[external]	
OSMatlab	334
OSmps2OS	345
OSmps2osil	347
OSnl2OS	348
OSnLPParserData	436
OSoLPParserData	439
OSoLReader	440
OSoLWriter	441
OSOption	442
osOptionsStruc	475
OSosl2ampl	477
OSOutputChannel	482
OSReferencedObject	484
OSOutput	478
OSReferencer	486
OSSmartPtr< T >	586
OSResult	486
OSrL2Gams	578
OSrLPParserData	579
OSrLReader	583
OSrLWriter	585
OtherConOption	595
OtherConResult	596
OtherConstraintOption	597
OtherConstraintResult	600
OtherMatrixVariableResult	602
OtherObjectiveOption	603
OtherObjectiveResult	605
OtherObjOption	607

OtherObjResult	609
OtherOption	610
OtherOptions	614
OtherResult	616
OtherResults	618
OtherSolutionResult	619
OtherSolutionResults	620
OtherSolverOutput	622
OtherVariableOption	623
OtherVariableResult	625
OtherVariableResultStruct	627
OtherVarOption	628
OtherVarResult	630
Outfo[external]	
ClpSimplexOther::parametricsData[external]	
parity_ilp[external]	
PathPair	632
PathPairs	633
Ipopt::PiecewisePenalty[external]	
Ipopt::PiecewisePenEntry[external]	
AbcSimplexPrimal::pivotStruct[external]	
pool_cut[external]	
pool_cut_list[external]	
presolvehlink[external]	
std::priority_queue< T >	
CbcHeuristicDive::PriorityType[external]	
Ipopt::AmplOptionsList::PrivatInfo[external]	
Processes	640
PseudoReducedCost[external]	
QuadraticCoefficients	645
QuadraticTerm	649
QuadraticTerms	649
std::queue< T >	
Coin::ReferencedObject[external]	
BCP_cut[external]	
BCP_cut_algo[external]	
BCP_cut_core[external]	
BCP_node_change[external]	
BCP_user_data[external]	
BCP_var[external]	
BCP_var_algo[external]	
BCP_var_core[external]	
Ipopt::ReferencedObject[external]	
Ipopt::AlgorithmBuilder[external]	
Ipopt::InexactAlgorithmBuilder[external]	
Ipopt::AlgorithmStrategyObject[external]	
Ipopt::AugSystemSolver[external]	
Ipopt::AugRestoSystemSolver[external]	
Ipopt::GenAugSystemSolver[external]	
Ipopt::LowRankAugSystemSolver[external]	
Ipopt::LowRankSSAugSystemSolver[external]	
Ipopt::StdAugSystemSolver[external]	
Ipopt::BacktrackingLSAcceptor[external]	
Ipopt::CGPenaltyLSAcceptor[external]	

- Ipopt::FilterLSAcceptor [external]
- Ipopt::InexactLSAcceptor [external]
- Ipopt::PenaltyLSAcceptor [external]
- Ipopt::ConvergenceCheck [external]
 - Ipopt::OptimalityErrorConvergenceCheck [external]
 - Ipopt::RestoConvergenceCheck [external]
 - Ipopt::RestoFilterConvergenceCheck [external]
 - Ipopt::RestoPenaltyConvergenceCheck [external]
- Ipopt::EqMultiplierCalculator [external]
 - Ipopt::LeastSquareMultipliers [external]
- Ipopt::GenKKTSolverInterface [external]
- Ipopt::HessianUpdater [external]
 - Ipopt::ExactHessianUpdater [external]
 - Ipopt::LimMemQuasiNewtonUpdater [external]
- Ipopt::InexactNewtonNormalStep [external]
- Ipopt::InexactNormalStepCalculator [external]
 - Ipopt::InexactDoglegNormalStep [external]
- Ipopt::InexactPDSolver [external]
- Ipopt::IpoptAlgorithm [external]
- Ipopt::IterateInitializer [external]
 - Ipopt::DefaultIterateInitializer [external]
 - Ipopt::RestoIterateInitializer [external]
 - Ipopt::WarmStartIterateInitializer [external]
- Ipopt::IterationOutput [external]
 - Ipopt::OrigIterationOutput [external]
 - Ipopt::RestoIterationOutput [external]
- Ipopt::IterativeSolverTerminationTester [external]
 - Ipopt::InexactNormalTerminationTester [external]
 - Ipopt::InexactPDTerminationTester [external]
- Ipopt::LineSearch [external]
 - Ipopt::BacktrackingLineSearch [external]
- Ipopt::MuOracle [external]
 - Ipopt::LoqoMuOracle [external]
 - Ipopt::ProbingMuOracle [external]
 - Ipopt::QualityFunctionMuOracle [external]
- Ipopt::MuUpdate [external]
 - Ipopt::AdaptiveMuUpdate [external]
 - Ipopt::MonotoneMuUpdate [external]
- Ipopt::PDPerturbationHandler [external]
 - Ipopt::CGPerturbationHandler [external]
- Ipopt::PDSolver [external]
 - Ipopt::PDFullSpaceSolver [external]
- Ipopt::RestorationPhase [external]
 - Ipopt::MinC_1NrmRestorationPhase [external]
 - Ipopt::RestoRestorationPhase [external]
- Ipopt::SearchDirectionCalculator [external]
 - Ipopt::CGSearchDirCalculator [external]
 - Ipopt::InexactSearchDirCalculator [external]
 - Ipopt::PDSearchDirCalculator [external]
- Ipopt::SparseSymLinearSolverInterface [external]
 - Ipopt::IterativePardisoSolverInterface [external]
 - Ipopt::IterativeWsmvSolverInterface [external]
 - Ipopt::Ma27TSolverInterface [external]
 - Ipopt::Ma57TSolverInterface [external]

```

    Ipopt::Ma77SolverInterface [external]
    Ipopt::Ma86SolverInterface [external]
    Ipopt::Ma97SolverInterface [external]
    Ipopt::MumpsSolverInterface [external]
    Ipopt::PardisoSolverInterface [external]
    Ipopt::WsmvSolverInterface [external]
    Ipopt::SymLinearSolver [external]
    Ipopt::TSymLinearSolver [external]
    Ipopt::TDependencyDetector [external]
    Ipopt::Ma28TDependencyDetector [external]
    Ipopt::TSymDependencyDetector [external]
    Ipopt::TSymScalingMethod [external]
    Ipopt::InexactTSymScalingMethod [external]
    Ipopt::Mc19TSymScalingMethod [external]
    Ipopt::SlackBasedTSymScalingMethod [external]
    Ipopt::AmplOptionsList [external]
    Ipopt::AmplOptionsList::AmplOption [external]
    Ipopt::AmplSuffixHandler [external]
    Ipopt::IpoptAdditionalCq [external]
    Ipopt::CGPenaltyCq [external]
    Ipopt::InexactCq [external]
    Ipopt::IpoptAdditionalData [external]
    Ipopt::CGPenaltyData [external]
    Ipopt::InexactData [external]
    Ipopt::IpoptApplication [external]
    Ipopt::IpoptCalculatedQuantities [external]
    Ipopt::IpoptData [external]
    Ipopt::IpoptNLP [external]
    Ipopt::OrigIpoptNLP [external]
    Ipopt::RestIpoptNLP [external]
    Ipopt::Journal [external]
    Ipopt::FileJournal [external]
    Ipopt::StreamJournal [external]
    Ipopt::Journalist [external]
    Ipopt::MatrixSpace [external]
    Ipopt::CompoundMatrixSpace [external]
    Ipopt::DenseGenMatrixSpace [external]
    Ipopt::ExpandedMultiVectorMatrixSpace [external]
    Ipopt::ExpansionMatrixSpace [external]
    Ipopt::GenTMatrixSpace [external]
    Ipopt::MultiVectorMatrixSpace [external]
    Ipopt::ScaledMatrixSpace [external]
    Ipopt::SumMatrixSpace [external]
    Ipopt::SymMatrixSpace [external]
    Ipopt::CompoundSymMatrixSpace [external]
    Ipopt::DenseSymMatrixSpace [external]
    Ipopt::DiagMatrixSpace [external]
    Ipopt::IdentityMatrixSpace [external]
    Ipopt::LowRankUpdateSymMatrixSpace [external]
    Ipopt::SumSymMatrixSpace [external]
    Ipopt::SymScaledMatrixSpace [external]
    Ipopt::SymTMatrixSpace [external]
    Ipopt::ZeroSymMatrixSpace [external]
    Ipopt::TransposeMatrixSpace [external]

```

- Ipopt::ZeroMatrixSpace [external]
- Ipopt::NLP [external]
 - Ipopt::NLPBoundsRemover [external]
 - Ipopt::TNLPAdapter [external]
- Ipopt::NLPScalingObject [external]
 - Ipopt::StandardScalingBase [external]
 - Ipopt::EquilibrationScaling [external]
 - Ipopt::GradientScaling [external]
 - Ipopt::NoNLPScalingObject [external]
 - Ipopt::UserScaling [external]
- Ipopt::OptionsList [external]
- Ipopt::PointPerturber [external]
- Ipopt::RegisteredOption [external]
- Ipopt::RegisteredOptions [external]
- Ipopt::SolveStatistics [external]
- Ipopt::TaggedObject [external]
 - Ipopt::Matrix [external]
 - Ipopt::CompoundMatrix [external]
 - Ipopt::DenseGenMatrix [external]
 - Ipopt::ExpandedMultiVectorMatrix [external]
 - Ipopt::ExpansionMatrix [external]
 - Ipopt::GenTMatrix [external]
 - Ipopt::MultiVectorMatrix [external]
 - Ipopt::ScaledMatrix [external]
 - Ipopt::SumMatrix [external]
 - Ipopt::SymMatrix [external]
 - Ipopt::CompoundSymMatrix [external]
 - Ipopt::DenseSymMatrix [external]
 - Ipopt::DiagMatrix [external]
 - Ipopt::IdentityMatrix [external]
 - Ipopt::LowRankUpdateSymMatrix [external]
 - Ipopt::SumSymMatrix [external]
 - Ipopt::SymScaledMatrix [external]
 - Ipopt::SymTMatrix [external]
 - Ipopt::ZeroSymMatrix [external]
 - Ipopt::TransposeMatrix [external]
 - Ipopt::ZeroMatrix [external]
- Ipopt::Vector [external]
 - Ipopt::CompoundVector [external]
 - Ipopt::IteratesVector [external]
 - Ipopt::DenseVector [external]
- Ipopt::TimingStatistics [external]
- Ipopt::TNLP [external]
 - Ipopt::AmplTNLP [external]
 - Ipopt::StdInterfaceTNLP [external]
 - Ipopt::TNLPReducer [external]
- IpoptProblem 163
- Ipopt::TripletToCSRConverter [external]
- Ipopt::VectorSpace [external]
 - Ipopt::CompoundVectorSpace [external]
 - Ipopt::IteratesVectorSpace [external]
 - Ipopt::DenseVectorSpace [external]
- Ipopt::Referencer [external]
- Ipopt::SmartPtr< T > [external]

```

Ipopt::SmartPtr< Bonmin::TNLPSolver > [external]
Ipopt::SmartPtr< BonminProblem > [external]
SmartPtr< const Ipopt::AmplOptionsList::AmplOption > [external]
SmartPtr< const Ipopt::CompoundVectorSpace > [external]
SmartPtr< const Ipopt::ExpansionMatrix > [external]
SmartPtr< const Ipopt::IteratesVector > [external]
SmartPtr< const Ipopt::Journalist > [external]
SmartPtr< const Ipopt::LowRankUpdateSymMatrixSpace > [external]
SmartPtr< const Ipopt::Matrix > [external]
SmartPtr< const Ipopt::MatrixSpace > [external]
SmartPtr< const Ipopt::MultiVectorMatrix > [external]
SmartPtr< const Ipopt::NLP > [external]
SmartPtr< const Ipopt::ScaledMatrixSpace > [external]
SmartPtr< const Ipopt::SymMatrix > [external]
SmartPtr< const Ipopt::SymMatrixSpace > [external]
SmartPtr< const Ipopt::SymScaledMatrixSpace > [external]
SmartPtr< const Ipopt::Vector > [external]
SmartPtr< const Ipopt::VectorSpace > [external]
SmartPtr< Ipopt::AmplSuffixHandler > [external]
SmartPtr< Ipopt::AugSystemSolver > [external]
SmartPtr< Ipopt::BacktrackingLSAcceptor > [external]
SmartPtr< Ipopt::CompoundMatrix > [external]
SmartPtr< Ipopt::CompoundMatrixSpace > [external]
SmartPtr< Ipopt::CompoundSymMatrix > [external]
SmartPtr< Ipopt::CompoundSymMatrixSpace > [external]
SmartPtr< Ipopt::CompoundVector > [external]
SmartPtr< Ipopt::CompoundVectorSpace > [external]
SmartPtr< Ipopt::ConvergenceCheck > [external]
SmartPtr< Ipopt::DenseGenMatrix > [external]
SmartPtr< Ipopt::DenseSymMatrix > [external]
SmartPtr< Ipopt::DenseVector > [external]
SmartPtr< Ipopt::DiagMatrix > [external]
SmartPtr< Ipopt::DiagMatrixSpace > [external]
SmartPtr< Ipopt::EqMultiplierCalculator > [external]
SmartPtr< Ipopt::ExpandedMultiVectorMatrix > [external]
SmartPtr< Ipopt::ExpansionMatrix > [external]
SmartPtr< Ipopt::ExpansionMatrixSpace > [external]
SmartPtr< Ipopt::GenKKTSolverInterface > [external]
SmartPtr< Ipopt::HessianUpdater > [external]
SmartPtr< Ipopt::IdentityMatrixSpace > [external]
SmartPtr< Ipopt::InexactNewtonNormalStep > [external]
SmartPtr< Ipopt::InexactNormalStepCalculator > [external]
SmartPtr< Ipopt::InexactNormalTerminationTester > [external]
SmartPtr< Ipopt::InexactPDSolver > [external]
SmartPtr< Ipopt::IpoptAdditionalCq > [external]
SmartPtr< Ipopt::IpoptAdditionalData > [external]
SmartPtr< Ipopt::IpoptAlgorithm > [external]
Ipopt::SmartPtr< Ipopt::IpoptApplication > [external]
SmartPtr< Ipopt::IpoptCalculatedQuantities > [external]
SmartPtr< Ipopt::IpoptData > [external]
SmartPtr< Ipopt::IpoptNLP > [external]
SmartPtr< Ipopt::IterateInitializer > [external]
SmartPtr< Ipopt::IteratesVectorSpace > [external]
SmartPtr< Ipopt::IterationOutput > [external]

```

```

SmartPtr< Ipopt::IterativeSolverTerminationTester > [external]
SmartPtr< Ipopt::Journal > [external]
SmartPtr< Ipopt::Journalist > [external]
SmartPtr< Ipopt::LineSearch > [external]
SmartPtr< Ipopt::Matrix > [external]
SmartPtr< Ipopt::MultiVectorMatrix > [external]
SmartPtr< Ipopt::MuOracle > [external]
SmartPtr< Ipopt::MuUpdate > [external]
SmartPtr< Ipopt::NLP > [external]
SmartPtr< Ipopt::NLPScalingObject > [external]
SmartPtr< Ipopt::OptionsList > [external]
SmartPtr< Ipopt::OrigIterationOutput > [external]
SmartPtr< Ipopt::PDPerturbationHandler > [external]
SmartPtr< Ipopt::PDSysSystemSolver > [external]
SmartPtr< Ipopt::RegisteredOption > [external]
SmartPtr< Ipopt::RegisteredOptions > [external]
SmartPtr< Ipopt::RestorationPhase > [external]
SmartPtr< Ipopt::ScaledMatrixSpace > [external]
SmartPtr< Ipopt::SearchDirectionCalculator > [external]
SmartPtr< Ipopt::SolveStatistics > [external]
SmartPtr< Ipopt::SparseSymLinearSolverInterface > [external]
SmartPtr< Ipopt::SumSymMatrixSpace > [external]
SmartPtr< Ipopt::SymLinearSolver > [external]
SmartPtr< Ipopt::SymMatrix > [external]
SmartPtr< Ipopt::SymScaledMatrixSpace > [external]
SmartPtr< Ipopt::TDependencyDetector > [external]
Ipopt::SmartPtr< Ipopt::TNLP > [external]
Ipopt::SmartPtr< Ipopt::TNLP > [external]
SmartPtr< Ipopt::TripletToCSRConverter > [external]
SmartPtr< Ipopt::TSymLinearSolver > [external]
SmartPtr< Ipopt::TSymScalingMethod > [external]
SmartPtr< Ipopt::Vector > [external]
std::string::reverse_iterator
std::multimap< K, T >::reverse_iterator
std::wstring::reverse_iterator
std::deque< T >::reverse_iterator
std::vector< T >::reverse_iterator
std::multiset< K >::reverse_iterator
std::unordered_multiset< K >::reverse_iterator
std::unordered_map< K, T >::reverse_iterator
std::unordered_set< K >::reverse_iterator
std::set< K >::reverse_iterator
std::unordered_multimap< K, T >::reverse_iterator
std::list< T >::reverse_iterator
std::forward_list< T >::reverse_iterator
std::map< K, T >::reverse_iterator
std::basic_string< Char >::reverse_iterator
scatterStruct [external]
select_cut [external]
separation_graph [external]
ServiceOption . . . . . 657
ServiceResult . . . . . 658
std::set< K >
short_path_node [external]

```


std::smart_ptr< T >	
Coin::SmartPtr< T >[external]	
SmartPtr< Ipopt::TNLP >[external]	
SolverOption	660
SolverOptions	662
SolverOutput	664
SOSVariableBranchingWeights	665
SOSWeights	667
SparseHessianMatrix	669
SparselntVector	670
SparseJacobianMatrix	672
SparseMatrix	673
SparseVector	674
std::stack< T >	
StorageCapacity	675
Ipopt::RegisteredOption::string_entry[external]	
Ipopt::Subject[external]	
Ipopt::TaggedObject[external]	
symrec[external]	
std::system_error	
SystemOption	677
SystemResult	679
OsiUnitTest::TestOutcome[external]	
OsiUnitTest::TestOutcomes[external]	
std::thread	
TimeDomain	680
TimeDomainInterval	681
TimeDomainStage	681
TimeDomainStageCon	682
TimeDomainStageConstraints	682
TimeDomainStageObj	683
TimeDomainStageObjectives	684
TimeDomainStages	684
TimeDomainStageVar	685
TimeDomainStageVariables	686
Ipopt::TimedTask[external]	
TimeSpan	687
TimeMeasurement	686
TimingInformation	689
TMINLP	
BonminProblem	49
Ipopt::TripletHelper[external]	
std::unique_ptr< T >	
std::unordered_map< K, T >	
std::unordered_multimap< K, T >	
std::unordered_multiset< K >	
std::unordered_set< K >	
USER_initialize[external]	
std::valarray< T >	
LAP::Validator[external]	
Variable	690
VariableOption	691
Variables	694
VariableSolution	694

VariableValues	696
VariableValuesString	697
VarValue	702
VarValueString	703
std::vector< T >	
std::vector< bool >	
std::vector< CbcNode * >	
std::vector< ColumnSelectionStrategy >	
std::vector< double >	
std::vector< ExpandedMatrixBlocks * >	
std::vector< ExprNode * >	
std::vector< IndexValuePair * >	
std::vector< int * >	
std::vector< int >	
std::vector< MatrixNode * >	
std::vector< OsiObject * >	
std::vector< OSnLNode * >	
std::vector< OtherVariableResultStruct * >	
std::vector< RowSelectionStrategy >	
std::vector< std::string >	
VOL_alpha_factor[external]	
VOL_dual[external]	
VOL_dvector[external]	
VOL_indc[external]	
VOL_ivector[external]	
VOL_parms[external]	
VOL_primal[external]	
VOL_problem[external]	
VOL_swing[external]	
VOL_user_hooks[external]	
OsiVolSolverInterface[external]	
VOL_vh[external]	
std::weak_ptr< T >	
WSUtil	705
YYLTYPE	709
YYSTYPE	709
K	
S	
T	
U	

Chapter 2

Class Index

2.1 Class List

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

Base64	Use this class to read and write data in base64	41
BaseMatrix	Data structure to represent a point of departure for constructing a matrix by modifying parts of a previously defined matrix	42
BasisStatus	Data structure to represent an LP basis on both input and output	45
BonminProblem	49
BonminSolver	Solves problems using lpopt	51
BranchingWeight	BranchingWeight class	52
CoinSolver	Implements a solve method for the Coin solvers	54
CompletelyPositiveMatricesCone	The CompletelyPositiveMatricesCone Class	56
Cone	The in-memory representation of a generic cone Specific cone types are derived from this generic class	58
Cones	The in-memory representation of the <cones> element	60
ConReferenceMatrixElement	Data structure to represent an entry in a conReferenceMatrix element, which consists of a constraint reference as well as a value type	62
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	63
ConReferenceMatrixValues	Data structure to represent the nonzeros in a conReferenceMatrix element	66
ConstantMatrixElements	Data structure to represent the constant elements in a MatrixType object	67
ConstantMatrixValues	To represent the nonzeros in a constantMatrix element	70

Constraint	
The in-memory representation of the <con> element	72
ConstraintOption	
ConstraintOption class	73
Constraints	
The in-memory representation of the <constraints> element	75
ConstraintSolution	
The ConstraintSolution Class	76
ContactOption	
ContactOption class	77
CopositiveMatricesCone	
The CopositiveMatricesCone Class	78
CouenneSolver	
Solves problems using Ipopt	80
CPUNumber	
CPUNumber class	81
CPUSpeed	
CPUSpeed class	83
CsdpSolver	
Solves problems using Csdp	85
DefaultSolver	
The Default Solver Class	86
DirectoriesAndFiles	
DirectoriesAndFiles class	88
DoubleVector	
Double vector data structure	89
DualCone	
The in-memory representation of a dual cone	90
DualVariableValues	
The DualVariableValues Class	92
DualVarValue	
The DualVarValue Class	93
ErrorClass	
Used for throwing exceptions	94
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	95
ExprNode	
A generic class from which we derive both OSnLNode and OSnLMNode	98
FileUtil	
Class used to make it easy to read and write files	102
GeneralFileHeader	
Data structure that holds general information about files that conform to one of the OSxL schemas	104
GeneralMatrixElements	
Data structure to represent the nonzero values in a generalMatrix element	106
GeneralMatrixValues	
Data structure to represent the nonzeros in a generalMatrix element	108
GeneralOption	
The GeneralOption Class	110
GeneralResult	
The GeneralResult Class	112
GeneralSparseMatrix	
Sparse matrix data structure for matrices that can hold nonconstant values	113

GeneralStatus	
The GeneralStatus Class	116
GeneralSubstatus	
The GeneralSubstatus Class	117
IndexStringPair	
A commonly used structure holding an index-string pair This definition is based on the definition of IndexValuePair in OSGeneral.h	118
IndexValuePair	
A commonly used structure holding an index-value pair	119
InitBasStatus	
InitBasStatus class	119
InitConstraintValues	
InitConstraintValues class	121
InitConValue	
InitConValue class	124
InitDualVariableValues	
InitDualVariableValues class	125
InitDualVarValue	
InitDualVarValue class	128
InitialBasisStatus	
InitialBasisStatus class	130
InitObjBound	
InitObjBound class	132
InitObjectiveBounds	
InitObjectiveBounds class	134
InitObjectiveValues	
InitObjectiveValues class	136
InitObjValue	
InitObjValue class	140
InitVariableValues	
InitVariableValues class	142
InitVariableValuesString	
InitVariableValuesString class	144
InitVarValue	
InitVarValue class	148
InitVarValueString	
InitVarValueString class	150
InstanceData	
The in-memory representation of the <instanceData> element	151
InstanceLocationOption	
InstanceLocationOption class	153
IntegerVariableBranchingWeights	
IntegerVariableBranchingWeights class	154
IntersectionCone	
The in-memory representation of an intersection cone	158
Interval	
The in-memory representation of the <interval> element	161
IntVector	
Integer Vector data structure	161
IpoptProblem	163
IpoptSolver	
Solves problems using Ipopt	165
JobDependencies	
JobDependencies class	166

JobOption	
JobOption class	169
JobResult	
The JobResult Class	171
KnitroProblem	173
KnitroSolver	
KnitroSolver class solves problems using Knitro	173
LindoSolver	
LindoSolver class solves problems using Lindo	175
LinearConstraintCoefficients	
The in-memory representation of the <code><linearConstraintCoefficients></code> element	178
LinearMatrixElement	
Data structure to represent an expression in a linearMatrix element A LinearMatrixElement is a (finite) sum of LinearMatrixElementTerms , with an optional additive constant	179
LinearMatrixElements	
Data structure to represent the nonzero values in a linearMatrix element	180
LinearMatrixElementTerm	
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	183
LinearMatrixValues	
Data structure to represent the linear expressions in a LinearMatrixElement object	184
MathUtil	
This class has routines for linear algebra	185
Matrices	
The in-memory representation of the <code><matrices></code> element	187
MatrixBlock	
Data structure to represent a MatrixBlock object (derived from MatrixType)	188
MatrixBlocks	
Data structure to represent the nonzeros of a matrix in a blockwise fashion	192
MatrixCon	
The in-memory representation of the <code><matrixCon></code> element	196
MatrixConstraints	
The in-memory representation of the <code><matrixConstraints></code> element	197
MatrixConstraintSolution	
The in-memory representation of the <code><MatrixConstraintSolution></code> element	197
MatrixConstructor	
Data structure to describe one step in the construction of a matrix	198
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	199
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	200
MatrixExpression	
The in-memory representation of the <code><expr></code> 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	201
MatrixExpressions	
The in-memory representation of the <code><matrixExpressions></code> element	202
MatrixExpressionTree	
Used to hold the instance in memory	203

MatrixNode	Generic class from which we derive matrix constructors (BaseMatrix , MatrixElements , MatrixTransformation and MatrixBlocks) as well as matrix types (OSMatrix and MatrixBlock)	204
MatrixObj	The in-memory representation of the <code><matrixObj></code> element	208
MatrixObjectives	The in-memory representation of the <code><matrixObjectives></code> element	209
MatrixObjectiveSolution	The in-memory representation of the <code><MatrixVariableSolution></code> element	210
MatrixProgramming	The in-memory representation of the <code><matrixProgramming></code> element	211
MatrixProgrammingSolution	The in-memory representation of the <code><MatrixProgrammingSolution></code> element	213
MatrixTransformation	Data structure to represent the nonzeros of a matrix by transformation from other (previously defined) matrices	214
MatrixType	Data structure to represent a MatrixType object (from which we derive OSMatrix and MatrixBlock)	217
MatrixVar	The in-memory representation of the <code><matrixVar></code> element	225
MatrixVariables	The in-memory representation of the <code><matrixVariables></code> element	227
MatrixVariableSolution	The in-memory representation of the <code><MatrixVariableSolution></code> element	227
MatrixVariableValues	The in-memory representation of the <code><matrixVariables></code> element	228
MaxTime	MaxTime class	229
MinCPUNumber	MinCPUNumber class	230
MinCPUSpeed	MinCPUSpeed class	231
MinDiskSpace	MinDiskSpace class	232
MinMemorySize	MinMemorySize class	233
MixedRowReferenceMatrixElements	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	234
NI	The in-memory representation of the <code><nl></code> element	237
NonlinearExpressions	The in-memory representation of the <code><nonlinearExpressions></code> element	238
NonnegativeCone	The NonnegativeCone Class	239
NonpositiveCone	The NonpositiveCone Class	240
ObjCoef	The in-memory representation of the objective function <code><coef></code> element	242
Objective	The in-memory representation of the <code><obj></code> element	243
ObjectiveOption	ObjectiveOption class	244

Objectives	
The in-memory representation of the <objectives> element	246
ObjectiveSolution	
The ObjectiveSolution Class	246
ObjectiveValues	
The ObjectiveValues Class	248
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)	249
ObjReferenceMatrixValues	
To represent the nonzeros in an objReferenceMatrix element	251
ObjValue	
The ObjValue Class	253
OptimizationOption	
OptimizationOption class	255
OptimizationResult	
The OptimizationResult Class	257
OptimizationSolution	
The OptimizationSolution Class	258
OptimizationSolutionStatus	
The OptimizationSolutionStatus Class	260
OptimizationSolutionSubstatus	
The OptimizationSolutionSubstatus Class	261
OrthantCone	
The OrthantCone Class	262
OS_AMPL_SUFFIX	264
OSCommandLine	
This class is used to store command line options for the OSSolverService executable and to provide methods to manipulate them	264
OSCommandLineReader	
The OSCommandLineReader Class	268
OSExpressionTree	
Used to hold the instance in memory	270
OSgams2osil	
Creating a OSInstance from a GAMS model given as GAMS Modeling Object (GMO)	271
OSGeneral	272
OSgLParserData	
The OSgLParserData Class	272
OShL	
An interface that specified virtual methods to be implemented by agents	274
OSiLParserData	
The OSiLParserData Class, used to store parser data	277
OSiLReader	
Used to read an OSiL string	280
OSiLWriter	
Take an OSInstance object and write a string that validates against the OSiL schema	281
OSInstance	
The in-memory representation of an OSiL instance	282
OSMatlab	
The OSMatlab Class	334
OSMatrix	
Data structure to represent a matrix object (derived from MatrixType)	337
OSMatrixWithMatrixConIdx	
This class extends OSMatrix for use, e.g., in the matrixCon section of OSoL and OSrL	341

OSMatrixWithMatrixObjIdx	
This class extends OSMatrix for use, e.g., in the matrixObj section of OSoL and OSrL	342
OSMatrixWithMatrixVarIdx	
This class extends OSMatrix for use, e.g., in the matrixVar section of OSoL and OSrL	344
OSmps2OS	
The OSmps2OS Class	345
OSmps2osil	
The OSmps2osil Class	347
OSnl2OS	
The OSnl2OS Class	348
OSnLMNode	
The OSnLMNode Class for nonlinear expressions involving matrices	352
OSnLMNodeDiagonalMatrixFromVector	355
OSnLMNodeIdentityMatrix	356
OSnLMNodeMatrixCon	357
OSnLMNodeMatrixDiagonal	358
OSnLMNodeMatrixDotTimes	359
OSnLMNodeMatrixInverse	360
OSnLMNodeMatrixLowerTriangle	361
OSnLMNodeMatrixMinus	362
OSnLMNodeMatrixNegate	363
OSnLMNodeMatrixObj	364
OSnLMNodeMatrixPlus	366
OSnLMNodeMatrixProduct	
The OSnLMNodeMatrixProduct Class	366
OSnLMNodeMatrixReference	368
OSnLMNodeMatrixScalarTimes	369
OSnLMNodeMatrixSubmatrixAt	370
OSnLMNodeMatrixSum	371
OSnLMNodeMatrixTimes	372
OSnLMNodeMatrixTranspose	373
OSnLMNodeMatrixUpperTriangle	374
OSnLMNodeMatrixVar	375
OSnLNode	
The OSnLNode Class for nonlinear expressions	377
OSnLNodeAbs	
The OSnLNodeAbs Class	381
OSnLNodeAllDiff	
The OSnLNodeAllDiff Class	383
OSnLNodeCos	
The OSnLNodeCos Class	385
OSnLNodeDivide	
The OSnLNodeDivide Class	387
OSnLNodeE	
The OSnLNodeE Class	389
OSnLNodeErf	
The OSnLNodeErf Class	391
OSnLNodeExp	
The OSnLNodeExp Class	393
OSnLNodeIf	
The OSnLNodeIf Class	395
OSnLNodeLn	
The OSnLNodeLn Class	397

OSnLNodeMatrixDeterminant	
The next few nodes evaluate to a scalar even though one or more of its arguments are matrices . . .	399
OSnLNodeMatrixToScalar	
The OSnLNodeMatrixTrace Class	401
OSnLNodeMatrixTrace	
The OSnLNodeMatrixTrace Class	403
OSnLNodeMax	
The OSnLNodeMax Class	405
OSnLNodeMin	
The OSnLNodeMin Class	407
OSnLNodeMinus	
The OSnLNodeMinus Class	409
OSnLNodeNegate	
The OSnLNodeNegate Class	411
OSnLNodeNumber	
The OSnLNodeNumber Class	413
OSnLNodePI	
The OSnLNodePI Class	415
OSnLNodePlus	
The OSnLNodePlus Class	418
OSnLNodePower	
The OSnLNodePower Class	419
OSnLNodeProduct	
The OSnLNodeProduct Class	421
OSnLNodeSin	
The OSnLNodeSin Class	423
OSnLNodeSqrt	
The OSnLNodeSqrt Class	425
OSnLNodeSquare	
The OSnLNodeSquare Class	427
OSnLNodeSum	
The OSnLNodeSum Class	429
OSnLNodeTimes	
The OSnLNodeTimes Class	431
OSnLNodeVariable	
The OSnLNodeVariable Class	433
OSnLParserData	
The OSnLParserData Class	436
OSoLParserData	
The OSoLParserData Class	439
OSoLReader	
Used to read an OSoL string	440
OSoLWriter	
Take an OSOption object and write a string that validates against the OSoL schema	441
OSOption	
The Option Class	442
osOptionsStruc	
This structure is used to store options for the OSSolverService executable	475
OSosrl2ampl	
The OSosrl2ampl Class	477
OSOutput	
This class handles all the output from OSSolverService, OSAmplClient and other executables derived from them	478

OSOutputChannel	
Class that holds information about one output channel (file, device, stream, peripheral, etc.)	482
OSReferencedObject	
ReferencedObject class	484
OSReferencer	
Pseudo-class, from which everything has to inherit that wants to use be registered as a Referencer for a ReferencedObject	486
OSResult	
The Result Class	486
OSrL2Gams	
Reads an optimization result and stores result and solution in a Gams Modeling Object	578
OSrLParserData	
The OSrLParserData Class	579
OSrLReader	
The OSrLReader Class	583
OSrLWriter	
Take an OSResult object and write a string that validates against OSrL	585
OSSmartPtr< T >	
Template class for Smart Pointers	586
OSSolverAgent	
Used by a client to invoke a remote solver	590
OtherConOption	
OtherConOption class	595
OtherConResult	
The OtherConResult Class	596
OtherConstraintOption	
OtherConstraintOption class	597
OtherConstraintResult	
The OtherConstraintResult Class	600
OtherMatrixVariableResult	
The in-memory representation of the <matrixVariables> <other> element	602
OtherObjectiveOption	
OtherObjectiveOption class	603
OtherObjectiveResult	
The OtherObjectiveResult Class	605
OtherObjOption	
OtherObjOption class	607
OtherObjResult	
The OtherObjResult Class	609
OtherOption	
OtherOption class	610
OtherOptionOrResultEnumeration	
Brief an integer vector data structure used in OSOption and OSResult	612
OtherOptions	
OtherOptions class	614
OtherResult	
The OtherResult Class	616
OtherResults	
The OtherResults Class	618
OtherSolutionResult	
The OtherSolutionResult Class	619
OtherSolutionResults	
The OtherSolutionResults Class	620

OtherSolverOutput	
The OtherSolverOutput Class	622
OtherVariableOption	
OtherVariableOption class	623
OtherVariableResult	
The OtherVariableResult Class	625
OtherVariableResultStruct	
A structure to information about an OtherVariableResult element	627
OtherVarOption	
OtherVarOption class	628
OtherVarResult	
OtherVarResult Class	630
PathPair	
PathPair class	632
PathPairs	
PathPairs class	633
PolarCone	
The in-memory representation of a polar cone	637
PolyhedralCone	
The in-memory representation of a polyhedral cone	638
Processes	
Processes class	640
ProductCone	
The in-memory representation of a product cone	643
QuadraticCoefficients	
The in-memory representation of the <quadraticCoefficients> element	645
QuadraticCone	
The in-memory representation of a quadratic cone	646
QuadraticTerm	
The in-memory representation of the <qTerm> element	649
QuadraticTerms	
Data structure for holding quadratic terms	649
RotatedQuadraticCone	
The in-memory representation of a rotated quadratic cone	650
ScalarExpressionTree	
Used to hold part of the instance in memory	653
SemidefiniteCone	
The in-memory representation of a cone of semidefinite matrices	655
ServiceOption	
ServiceOption class	657
ServiceResult	
The ServiceResult Class	658
SolverOption	
SolverOption class	660
SolverOptions	
SolverOptions class	662
SolverOutput	
The SolverOutput Class	664
SOSVariableBranchingWeights	
SOSVariableBranchingWeights class	665
SOSWeights	
SOSWeights class	667
SparseHessianMatrix	
The in-memory representation of a SparseHessianMatrix	669

SparseIntVector	
Sparse vector data structure for integer vectors	670
SparseJacobianMatrix	
Sparse Jacobian matrix data structure	672
SparseMatrix	
Sparse matrix data structure	673
SparseVector	
Sparse vector data structure	674
StorageCapacity	
StorageCapacity class	675
SystemOption	
SystemOption class	677
SystemResult	
The SystemResult Class	679
TimeDomain	
The in-memory representation of the <timeDomain> element	680
TimeDomainInterval	681
TimeDomainStage	
The in-memory representation of the <stage> element	681
TimeDomainStageCon	
The in-memory representation of the <con> element	682
TimeDomainStageConstraints	
The in-memory representation of the <constraints> child of the <stage> element	682
TimeDomainStageObj	
The in-memory representation of the <obj> element	683
TimeDomainStageObjectives	
The in-memory representation of the <objectives> child of the <stage> element	684
TimeDomainStages	
The in-memory representation of the <stages> element	684
TimeDomainStageVar	
The in-memory representation of the element	685
TimeDomainStageVariables	
The in-memory representation of the <variables> child of the <stage> element	686
TimeMeasurement	
The TimeMeasurement Class	686
TimeSpan	
TimeSpan class	687
TimingInformation	
The TimingInformation Class	689
Variable	
The in-memory representation of the variable element	690
VariableOption	
VariableOption class	691
Variables	
The in-memory representation of the variables element	694
VariableSolution	
The VariableSolution Class	694
VariableValues	
The VariableValues Class	696
VariableValuesString	
The VariableValuesString Class	697
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	698

VarReferenceMatrixValues	
A concrete class that is used to store a specific type of matrix values, references to variable indexes defined in the core section	701
VarValue	
VarValue Class	702
VarValueString	
VarValueString Class	703
WSUtil	
Used by OSSolverAgent client for help in invoking a remote solver	705
YYLTYPE	709
YYSTYPE	709

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

config_default.h	??
config_os_default.h	??
OSBase64.h	??
OSBonminSolver.h	??
OSCoinSolver.h	??
OSCommandLine.h	711
OSCommandLineReader.h	711
OSConfig.h	??
OSCouenneSolver.h	712
OSCsdpSolver.h	712
OSDataStructures.h	??
OSDefaultSolver.h	??
OSdtoa.h	??
OSErrorClass.h	??
OSExpressionTree.h	713
OSFileUtil.h	??
OSgams2osil.hpp	??
OSGeneral.h	714
OSgLParserData.h	715
OSgLWriter.h	716
OShL.h	718
OSiLParserData.h	719
OSiLReader.h	719
OSiLWriter.h	720
OSInstance.h	
This file defines the OSInstance class along with its supporting classes	720
OSIpoptSolver.h	??
OSKnitroSolver.h	??
OSLindoSolver.h	??
OSMathUtil.h	??
OSMatlabSolver.h	??
OSMatrix.h	723
OSmps2OS.h	725
OSmps2osil.h	726

OSnl2OS.h	??
OSnLNode.h	
This file defines the OSnLNode class along with its derived classes	726
OSnLParserData.h	729
OSoLParserData.h	729
OSoLReader.h	730
OSoLWriter.h	730
OSOption.h	731
OSOptionsStruc.h	733
OSosrl2ampl.h	734
OSosrl2gams.hpp	??
OSOutput.h	734
OSParameters.h	735
OSParseosil.tab.hpp	??
OSParseosol.tab.hpp	??
OSParseosrl.tab.hpp	??
OSReferenced.hpp	??
OSResult.h	737
OSrLParserData.h	739
OSrLReader.h	739
OSrLWriter.h	740
OSRunSolver.h	740
OSSmartPtr.hpp	??
OSSolverAgent.h	743
OSStringUtil.h	743
OSWSUtil.h	745

Chapter 4

Class Documentation

4.1 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

4.1.1 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.

4.1.2 Member Function Documentation

4.1.2.1 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.

4.1.2.2 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:

- OSBase64.h

4.2 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.
- 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` [baseMatrixStartRow](#)
to select the position of the upper left corner of the portion of the base matrix that is to be selected
- `int` [baseMatrixEndRow](#)
to select the position of the lower right corner of the portion of the base matrix that is to be selected
- `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

4.2.1 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 1536 of file `OSMatrix.h`.

4.2.2 Member Function Documentation

4.2.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE BaseMatrix::getNodeType ()` `[virtual]`

Returns

the value of `nType`

Reimplemented from [MatrixNode](#).

4.2.2.2 `virtual std::string BaseMatrix::getNodeName () [virtual]`

Returns

the name of the operator

Implements [MatrixNode](#).

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

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.2.2.4 `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](#).

4.2.2.5 `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](#).

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

The implementation of the virtual functions.

Returns

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

Implements [MatrixNode](#).

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

- [OSMatrix.h](#)

4.3 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

- 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.

4.3.1 Detailed Description

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

Definition at line 645 of file OSGeneral.h.

4.3.2 Member Function Documentation

4.3.2.1 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 "numberOf← XXX" 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

4.3.2.2 `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

4.3.2.3 `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_STAT↵ US - see below)
<i>i</i>	contains the array of indices
<i>ni</i>	contains the number of elements in i

4.3.2.4 `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_STAT↵ US - see below)
<i>idx</i>	contains the value of the index

4.3.2.5 `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

4.3.2.6 int BasisStatus::getEl (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

4.3.2.7 `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_ST↵ ATUS — 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.

4.3.2.8 `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 <i>resultArray</i>
<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.

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

- [OSGeneral.h](#)

4.4 BonminProblem Class Reference

Inheritance diagram for BonminProblem:

Collaboration diagram for BonminProblem:

Public Member Functions

- **BonminProblem** (*OSInstance* *osinstance_, *OSOption* *osoption_)
the *BonminProblem* class constructor
- virtual **~BonminProblem** ()
the *BonminProblem* class destructor

Overloaded functions specific to a TMINLP.

now for some pure Bonmin methods

- virtual bool **get_variables_types** (Ipopt::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_, lpopt::Index n, const lpopt::Number *x, lpopt::Number obj_value)
*Method called by **lpopt** at the end of optimization.*

4.4.1 Detailed Description

Definition at line 53 of file OSBonminSolver.h.

4.4.2 Member Function Documentation

4.4.2.1 virtual bool BonminProblem::get_variables_types (lpopt::Index n, VariableType * var_types) [virtual]

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

Parameters

<i>n</i>	size of var_types (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).

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

Pass info about linear and nonlinear variables.

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

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

Parameters

<i>m</i>	size of const_types (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).

4.4.2.4 virtual bool BonminProblem::get_nlp_info (lpopt::Index & n, lpopt::Index & m, lpopt::Index & nnz_jac_g, lpopt::Index & nnz_h_lag, lpopt::TNLP::IndexStyleEnum & index_style) [virtual]

Method to pass the main dimensions of the problem to **lpopt**.

Parameters

<i>n</i>	number of variables in problem.
<i>m</i>	number of constraints.
<i>nnz_jac_g</i>	number of nonzeros in Jacobian of constraints system.

<i>nnz_h_lag</i>	number of nonzeros 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.

4.4.2.5 `virtual bool BonminProblem::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]`

Bonmin specific methods for defining the nlp problem.

Method to return the bounds for my problem

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

Method called by **Ipopt** at the end of optimization.

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

- OSBonminSolver.h

4.5 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

- [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

4.5.1 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.

4.5.2 Member Function Documentation

4.5.2.1 void BonminSolver::setSolverOptions () throw **ErrorClass** [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

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

- OSBonminSolver.h

4.6 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

4.6.1 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.

4.6.2 Member Function Documentation

4.6.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.6.2.2 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

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

- [OSOption.h](#)

4.7 CoinSolver Class Reference

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

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.

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 osoption from an osol string if needed

4.7.1 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.

4.7.2 Member Function Documentation

4.7.2.1 void CoinSolver::solve () throw **ErrorClass** [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

4.7.2.2 void CoinSolver::buildSolverInstance () throw **ErrorClass** [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

4.7.2.3 void CoinSolver::setSolverOptions () throw **ErrorClass** [virtual]

The implementation of the corresponding virtual function.

Returns

void.

Implements [DefaultSolver](#).

4.7.2.4 `bool CoinSolver::setCoinPackedMatrix ()`

Create a **CoinPackedMatrix**.

Returns

true if a **CoinPackedMatrix** successfully created.

4.7.2.5 `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.

4.7.2.6 `string CoinSolver::dataEchoCheck ()`

Print out problem parameters.

Returns

void

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

- OSCoinSolver.h

4.8 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

4.8.1 Detailed Description

The [CompletelyPositiveMatricesCone](#) Class.

Remarks

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

Definition at line 1186 of file OSInstance.h.

4.8.2 Member Function Documentation

4.8.2.1 virtual std::string [CompletelyPositiveMatricesCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.8.2.2 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](#).

4.8.2.3 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 "numberOf↵XXX" 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

4.8.2.4 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:

- [OSInstance.h](#)

4.9 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.
- virtual [~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 [numberOfOtherIndexes](#)
[Cones](#) can also be formed by Multidimensional tensors.
- 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

4.9.1 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.

4.9.2 Member Function Documentation

4.9.2.1 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](#).

4.9.2.2 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](#).

4.9.2.3 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 "numberOf↵XXX" 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

4.9.2.4 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

4.9.3 Member Data Documentation

4.9.3.1 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.

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

- [OSInstance.h](#)

4.10 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

4.10.1 Detailed Description

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

Definition at line 1532 of file OSInstance.h.

4.10.2 Member Function Documentation

4.10.2.1 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 "numberOfXXX" 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

4.10.2.2 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

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

- [OSInstance.h](#)

4.11 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

- 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.

4.11.1 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 MixedRowReferenceMatrix elements. A MixedRowReferenceMatrix is obtained by combining ObjReferenceMatrix elements and ConReferenceMatrix elements into a single matrix constructor.

Definition at line 453 of file OSMatrix.h.

4.11.2 Member Function Documentation

4.11.2.1 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 "numberOfXXX" 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

4.11.2.2 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

4.11.3 Member Data Documentation

4.11.3.1 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.

4.11.3.2 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.

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

- [OSMatrix.h](#)

4.12 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

- virtual `ENUM_MATRIX_CONSTRUCTOR_TYPE` [getNode](#)()
- 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 *` [value](#)
The constraint references (indexes of core constraints and value types) of the elements.

4.12.1 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 1177 of file OSMatrix.h.

4.12.2 Member Function Documentation

4.12.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE` `ConReferenceMatrixElements::getNode`() `[virtual]`

Returns

the constructor type

Reimplemented from [MatrixNode](#).

4.12.2.2 `virtual ENUM_MATRIX_TYPE` `ConReferenceMatrixElements::getMatrixType`() `[virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.12.2.3 `virtual std::string ConReferenceMatrixElements::getNodeName () [virtual]`

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.12.2.4 `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](#).

4.12.2.5 `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](#).

4.12.2.6 `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](#).

4.12.2.7 `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 "numberOf↵ XXX" 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

4.12.2.8 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

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

- [OSMatrix.h](#)

4.13 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

- 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.
- virtual 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.

4.13.1 Detailed Description

a data structure to represent the nonzeros in a conReferenceMatrix element

Definition at line 711 of file OSMatrix.h.

4.13.2 Member Function Documentation

4.13.2.1 `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 "numberOf↵XXX" 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

4.13.2.2 `virtual bool ConReferenceMatrixValues::deepCopyFrom (ConReferenceMatrixValues * that) [virtual]`

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:

- [OSMatrix.h](#)

4.14 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

- virtual `ENUM_MATRIX_CONSTRUCTOR_TYPE` [getNode](#)`Type` ()
- virtual `std::string` [getNode](#)`Name` ()
- virtual `ENUM_MATRIX_TYPE` [getMatrix](#)`Type` ()
- 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](#) * value

The value array of the (nonzero) constant elements.

4.14.1 Detailed Description

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

Definition at line 749 of file OSMatrix.h.

4.14.2 Member Function Documentation

4.14.2.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [ConstantMatrixElements::getNodeType](#) () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.14.2.2 virtual std::string [ConstantMatrixElements::getNodeName](#) () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.14.2.3 virtual ENUM_MATRIX_TYPE [ConstantMatrixElements::getMatrixType](#) () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.14.2.4 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](#).

4.14.2.5 `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](#).

4.14.2.6 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](#).

4.14.2.7 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 "numberOf↵XXX" 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

4.14.2.8 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

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

- [OSMatrix.h](#)

4.15 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

- 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.

Additional Inherited Members

4.15.1 Detailed Description

to represent the nonzeros in a constantMatrix element

Definition at line 501 of file OSMatrix.h.

4.15.2 Member Function Documentation

4.15.2.1 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.

4.15.2.2 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 "numberOf↵ XXX" 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

4.15.2.3 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

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

- [OSMatrix.h](#)

4.16 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

4.16.1 Detailed Description

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

Definition at line 218 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.17 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

4.17.1 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.

4.17.2 Member Function Documentation

4.17.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.17.2.2 `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

4.17.2.3 `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

4.17.2.4 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
--------------	--

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

- [OSOption.h](#)

4.18 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

4.18.1 Detailed Description

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

Definition at line 251 of file OSInstance.h.

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

- [OSInstance.h](#)

4.19 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

4.19.1 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 1860 of file OSResult.h.

4.19.2 Member Function Documentation

4.19.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.20 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

4.20.1 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.

4.20.2 Member Function Documentation**4.20.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.20.2.2 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

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

- [OSOption.h](#)

4.21 CopositiveMatricesCone Class Reference

The [CopositiveMatricesCone](#) Class.

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

Inheritance diagram for CopositiveMatricesCone:

Collaboration diagram for CopositiveMatricesCone:

Public Member Functions

- [CpositiveMatricesCone](#) ()
default constructor.
- [~CpositiveMatricesCone](#) ()
default destructor.
- virtual std::string [getConeName](#) ()
- virtual std::string [getConeInXML](#) ()
Write a [CpositiveMatricesCone](#) object in XML format.
- bool [IsEqual](#) ([CpositiveMatricesCone](#) *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](#) ([CpositiveMatricesCone](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.21.1 Detailed Description

The [CpositiveMatricesCone](#) Class.

Remarks

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

Definition at line 1127 of file OSInstance.h.

4.21.2 Member Function Documentation

4.21.2.1 virtual std::string [CpositiveMatricesCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.21.2.2 virtual std::string [CpositiveMatricesCone::getConeInXML](#) () [virtual]

Write a [CpositiveMatricesCone](#) 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](#).

4.21.2.3 bool [CpositiveMatricesCone::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 "numberOf↵XXX" 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

4.21.2.4 `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:

- [OSInstance.h](#)

4.22 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 osoption from an osol string if needed

4.22.1 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.

4.22.2 Member Function Documentation

4.22.2.1 `void CouenneSolver::setSolverOptions () throw ErrorClass` `[virtual]`

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

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

- [OSCouenneSolver.h](#)

4.23 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

4.23.1 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.

4.23.2 Member Function Documentation

4.23.2.1 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 "numberOfXXX" attributes and <XXX> children)

4.23.2.2 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

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

- [OSGeneral.h](#)

4.24 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)

4.24.1 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.

4.24.2 Member Function Documentation

4.24.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.24.2.2 `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

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

- [OSGeneral.h](#)

4.25 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
- virtual [~CsdpSolver](#) ()
the [CsdpSolver](#) class destructor
- virtual void [solve](#) () throw (ErrorClass)
solve results in an instance being read into the Csdp data structures and optimized
- 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

- [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

4.25.1 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 72 of file OSCsdpSolver.h.

4.25.2 Member Function Documentation

4.25.2.1 void CsdpSolver::buildSolverInstance () throw **ErrorClass**) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

4.25.2.2 void CsdpSolver::setSolverOptions () throw **ErrorClass**) [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

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

- [OSCsdpSolver.h](#)

4.26 DefaultSolver Class Reference

The Default Solver Class.

Inheritance diagram for DefaultSolver:

Collaboration diagram for DefaultSolver:

Public Member Functions

- 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

4.26.1 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.

4.26.2 Member Data Documentation

4.26.2.1 `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.

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

- OSDefaultSolver.h

4.27 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

4.27.1 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.

4.27.2 Member Function Documentation

4.27.2.1 `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 "numberOfXXX" attributes and <XXX> children)

4.27.2.2 `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

4.27.2.3 `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

4.27.2.4 `bool DirectoriesAndFiles::addPath (std::string path)`

A function to add a <path> element.

Parameters

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

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

- [OSOption.h](#)

4.28 DoubleVector Class Reference

a double vector data structure

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

Public Attributes

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

4.28.1 Detailed Description

a double vector data structure

Definition at line 609 of file `OSGeneral.h`.

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

- [OSGeneral.h](#)

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

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

4.29.1 Detailed Description

The in-memory representation of a dual cone.

Definition at line 1400 of file OSInstance.h.

4.29.2 Member Function Documentation

4.29.2.1 virtual std::string DualCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.29.2.2 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 "numberOfXXX" 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

4.29.2.3 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

4.29.3 Member Data Documentation

4.29.3.1 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 1421 of file OSInstance.h.

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

- [OSInstance.h](#)

4.30 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.

4.30.1 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 1654 of file OSResult.h.

4.30.2 Member Function Documentation

4.30.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.31 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

4.31.1 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 1598 of file OSResult.h.

4.31.2 Member Function Documentation**4.31.2.1** `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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.32 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

4.32.1 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.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 ErrorClass::ErrorClass (std::string errmsg_)

the class constructor

Parameters

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

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

- OSErrClass.h

4.33 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.
- [GeneralSparseMatrix](#) * [getBlock](#) (int rowIdx, int colIdx)
a method to retrieve a particular block from a collection
- bool [isBlockDiagonal](#) ()
a method to determine whether the collection is blockDiagonal

Public Attributes

- bool [bDeleteArrays](#)
bDeleteArrays is true if we delete the arrays in garbage collection set to true by default
- [ENUM_MATRIX_TYPE](#) [vType](#)
vType holds the type of all (nonzero) values in the collection of blocks contained in this set of blocks.
- bool [isRowMajor](#)
isRowMajor holds whether the (nonzero) values holding the data are stored by columnrow.
- int [blockNumber](#)
blockNumber gives the number of blocks (which is the size of the blockRows and blockColumns arrays).
- int * [rowOffset](#)
rowOffset 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 * [colOffset](#)
colOffset 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 [rowOffsetSize](#)
These two parameters give the size of the rowOffset and colOffset arrays, respectively.
- 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.

4.33.1 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 1768 of file OSMatrix.h.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 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.

4.33.3 Member Function Documentation

4.33.3.1 `bool ExpandedMatrixBlocks::display (int secondaryDim)`

This method displays data structure in the matrix format.

Returns

4.33.3.2 `GeneralSparseMatrix* ExpandedMatrixBlocks::getBlock (int rowIdx, int colIdx)`

a method to retrieve a particular block from a collection

Parameters

<i>rowIdx</i>	is the row index of the block to be retrieved
<i>colIdx</i>	is the column index of the block to be retrieved

Returns

a pointer to the block (as a [GeneralSparseMatrix](#))

4.33.3.3 `bool ExpandedMatrixBlocks::isBlockDiagonal ()`

a method to determine whether the collection is blockDiagonal

Returns

whether the collection is blockDiagonal or not

4.33.4 Member Data Documentation

4.33.4.1 `bool ExpandedMatrixBlocks::isRowMajor`

isRowMajor holds whether the (nonzero) values holding the data are stored by columnrow.

If false, the matrix is stored by column.

Definition at line 1787 of file OSMatrix.h.

4.33.4.2 `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 1823 of file `OSMatrix.h`.

4.33.4.3 `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 1831 of file `OSMatrix.h`.

4.33.4.4 `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 1838 of file `OSMatrix.h`.

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

- [OSMatrix.h](#)

4.34 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 [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.

- unsigned int [inumberOfMatrixChildren](#)

inumberOfMatrixChildren 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.

4.34.1 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](#).

4.34.2 Member Function Documentation

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

Returns

the value of [inodeInt](#)

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

4.34.2.2 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](#).

4.34.2.3 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](#).

4.34.2.4 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](#).

4.34.2.5 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 prefixVector to a vector of pointers of ExprNodes
----------	---

Returns

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

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

4.34.2.6 `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](#).

4.34.2.7 `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

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

Returns

a vector of pointers to ExprNodes in postfix.

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

4.34.2.8 `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](#), [OSnLMNodeMatrix↔Var](#), [OSnLMNodeMatrixReference](#), [OSnLMNodeMatrixSubmatrixAt](#), [OSnLMNodeDiagonalMatrixFromVector](#), [OSnLMNodeMatrixDiagonal](#), [OSnLMNodeMatrixUpperTriangle](#), [OSnLMNodeMatrixLowerTriangle](#), [OSnLMNodeIdentity↔Matrix](#), [OSnLMNodeMatrixDotTimes](#), [OSnLMNodeMatrixScalarTimes](#), [OSnLMNodeMatrixTranspose](#), [OSnLMNode↔MatrixInverse](#), [OSnLMNodeMatrixTimes](#), [OSnLMNodeMatrixNegate](#), [OSnLMNodeMatrixMinus](#), [OSnLMNodeMatrix↔Sum](#), [OSnLMNodeMatrixPlus](#), [OSnLNodeMatrixToScalar](#), [OSnLNodeMatrixTrace](#), [OSnLNodeMatrixDeterminant](#), [O↔SnLNodeAllDiff](#), [OSnLNodeVariable](#), [OSnLNodePI](#), [OSnLNodeE](#), [OSnLNodeNumber](#), [OSnLNodeErf](#), [O↔SnLNodeAbs](#), [OSnLNodeExp](#), [OSnLNodeSin](#), [OSnLNodeCos](#), [OSnLNodeSquare](#), [OSnLNodeSqrt](#), [OSnLNodeLn](#), [O↔SnLNodeProduct](#), [OSnLNodePower](#), [OSnLNodeDivide](#), [OSnLNodeTimes](#), [OSnLNodeNegate](#), [OSnLNodeMinus](#), [O↔SnLNodeMin](#), [OSnLNodeMax](#), [OSnLNodeSum](#), and [OSnLNodePlus](#).

4.34.3 Member Data Documentation

4.34.3.1 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.

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

- [OSnLNode.h](#)

4.35 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.

4.35.1 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.

4.35.2 Member Function Documentation

4.35.2.1 `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.

4.35.2.2 `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.

4.35.2.3 `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.

4.35.2.4 `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.

4.35.2.5 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:

- OSFileUtil.h

4.36 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

4.36.1 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.

4.36.2 Member Function Documentation

4.36.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.36.2.2 `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

4.36.2.3 `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)
-------------	--

4.36.2.4 `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

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

- [OSGeneral.h](#)

4.37 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

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

4.37.1 Detailed Description

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

Definition at line 1005 of file OSMatrix.h.

4.37.2 Member Function Documentation

4.37.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE GeneralMatrixElements::getNodeType () [virtual]`

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.37.2.2 `virtual ENUM_MATRIX_TYPE GeneralMatrixElements::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.37.2.3 `virtual std::string GeneralMatrixElements::getNodeName () [virtual]`

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.37.2.4 `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](#).

4.37.2.5 `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](#).

4.37.2.6 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](#).

4.37.2.7 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 "numberOfXXX" 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

4.37.2.8 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

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

- [OSMatrix.h](#)

4.38 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

- 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.
- virtual bool [deepCopyFrom](#) ([GeneralMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.38.1 Detailed Description

a data structure to represent the nonzeros in a generalMatrix element

Definition at line 639 of file OSMatrix.h.

4.38.2 Member Function Documentation

4.38.2.1 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 "numberOf↵XXX" 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

4.38.2.2 virtual bool GeneralMatrixValues::deepCopyFrom (GeneralMatrixValues * that) [virtual]

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:

- [OSMatrix.h](#)

4.39 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

4.39.1 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.

4.39.2 Member Function Documentation

4.39.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.39.2.2 `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

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

- [OSOption.h](#)

4.40 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

4.40.1 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 266 of file OSResult.h.

4.40.2 Member Function Documentation

4.40.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.41 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.
- bool [isDiagonal](#) ()
a method to determine whether the matrix is diagonal

Public Attributes

- bool [b_deleteStartArray](#)
b_deleteStartArray is true if we delete the start array in garbage collection — set to true by default
- bool [b_deleteIndexArray](#)
b_deleteIndexArray is true if we delete the index array in garbage collection — set to true by default
- bool [b_deleteValueArray](#)
b_deleteValueArray is true if we delete the value array in garbage collection — set to true by default
- bool [isRowMajor](#)
isRowMajor holds whether the matrix is stored by row.
- ENUM_MATRIX_SYMMETRY [symmetry](#)
To track the type of symmetry present in the matrix or block.
- int [startSize](#)
startSize is the dimension of the starts array
- int [valueSize](#)
valueSize is the dimension of the index and value arrays
- int * [start](#)
start holds an integer array of start elements in the matrix, which points to the start of a column (row) of nonzero elements.
- int * [index](#)
index holds an integer array of rowIdx (or colIdx) elements in coefMatrix (AMatrix).
- ENUM_MATRIX_TYPE [vType](#)
vType holds the type of values found in the value array.
- [MatrixElementValues](#) * [value](#)
value 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.

4.41.1 Detailed Description

a sparse matrix data structure for matrices that can hold nonconstant values

Definition at line 1654 of file OSMatrix.h.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 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).

4.41.3 Member Function Documentation

4.41.3.1 `bool GeneralSparseMatrix::display (int secondaryDim)`

This method displays the data contained in the matrix.

Returns

4.41.3.2 `bool GeneralSparseMatrix::isDiagonal ()`

a method to determine whether the matrix is diagonal

Returns

whether the matrix is diagonal or not

4.41.4 Member Data Documentation

4.41.4.1 `bool GeneralSparseMatrix::isRowMajor`

isRowMajor holds whether the matrix is stored by row.

If false, the matrix is stored by column (which is the default).

Definition at line 1680 of file OSMatrix.h.

4.41.4.2 `ENUM_MATRIX_SYMMETRY GeneralSparseMatrix::symmetry`

To track the type of symmetry present in the matrix or block.

Remarks

for definitions, see [OSParameters.h](#)

Definition at line 1686 of file OSMatrix.h.

4.41.4.3 `int* GeneralSparseMatrix::index`

index 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 1708 of file OSMatrix.h.

4.41.4.4 `ENUM_MATRIX_TYPE GeneralSparseMatrix::vType`

vType holds the type of values found in the value array.

Remarks

See [OSParameters.h](#) for a list of possible types

Definition at line 1714 of file OSMatrix.h.

4.41.4.5 **MatrixElementValues*** **GeneralSparseMatrix::value**

value 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 1723 of file OSMatrix.h.

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

- [OSMatrix.h](#)

4.42 **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 substatures
- std::string [type](#)
the type of status
- std::string [description](#)
the description of the status
- [GeneralSubstatus](#) ** [substatus](#)
the array of substatures

4.42.1 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 105 of file OSResult.h.

4.42.2 Member Function Documentation**4.42.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.43 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

4.43.1 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 54 of file OSResult.h.

4.43.2 Member Function Documentation

4.43.2.1 `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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.44 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

4.44.1 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 29 of file OSResult.h.

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

- [OSResult.h](#)

4.45 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

4.45.1 Detailed Description

A commonly used structure holding an index-value pair.

Definition at line 630 of file OSGeneral.h.

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

- [OSGeneral.h](#)

4.46 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

4.46.1 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.

4.46.2 Member Function Documentation

4.46.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.46.2.2 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

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

- [OSOption.h](#)

4.47 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

4.47.1 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.

4.47.2 Member Function Documentation

4.47.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.47.2.2 `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

4.47.2.3 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

4.47.2.4 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>

4.47.2.5 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

4.47.2.6 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

4.47.2.7 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

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

- [OSOption.h](#)

4.48 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

4.48.1 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.

4.48.2 Member Function Documentation

4.48.2.1 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 "numberOfXXX" attributes and <XXX> children)

4.48.2.2 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

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

- [OSOption.h](#)

4.49 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

4.49.1 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.

4.49.2 Member Function Documentation

4.49.2.1 `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 "numberOfXXX" attributes and <XXX> children)

4.49.2.2 `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

4.49.2.3 `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

4.49.2.4 `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>

4.49.2.5 `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

4.49.2.6 `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

4.49.2.7 `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

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

- [OSOption.h](#)

4.50 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

4.50.1 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.

4.50.2 Member Function Documentation

4.50.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.50.2.2 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

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

- [OSOption.h](#)

4.51 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

4.51.1 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.

4.51.2 Member Function Documentation

4.51.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.51.2.2 `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

4.51.2.3 `bool InitialBasisStatus::setVar (int numberOfVar, InitBasStatus ** 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 to be that are to be set</i>

4.51.2.4 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

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

- [OSOption.h](#)

4.52 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

4.52.1 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.

4.52.2 Member Function Documentation

4.52.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

4.52.2.2 `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

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

- [OSOption.h](#)

4.53 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

4.53.1 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.

4.53.2 Member Function Documentation

4.53.2.1 `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 "numberOf← XXX" attributes and <XXX> children)

4.53.2.2 `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

4.53.2.3 `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

4.53.2.4 `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>

4.53.2.5 `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

4.53.2.6 `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

4.53.2.7 `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

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

- [OSOption.h](#)

4.54 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*

4.54.1 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.

4.54.2 Member Function Documentation

4.54.2.1 `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 "numberOf←XXX" attributes and <XXX> children)

4.54.2.2 `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

4.54.2.3 `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

4.54.2.4 `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>

4.54.2.5 `bool InitObjectiveValues::setObj (int numberOfObj, int * idx, double * value, 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>value</i>	the array of corresponding values
<i>name</i>	the array of objective names

4.54.2.6 `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

4.54.2.7 `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

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

- [OSOption.h](#)

4.55 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

4.55.1 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.

4.55.2 Member Function Documentation

4.55.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.55.2.2 `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

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

- [OSOption.h](#)

4.56 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

4.56.1 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.

4.56.2 Member Function Documentation

4.56.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.56.2.2 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

4.56.2.3 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>	<i>the array of elements that are to be set</i>

4.56.2.4 bool InitVariableValues::setVar (int *numberOfVar*, InitVarValue ** *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>

4.56.2.5 `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 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>

4.56.2.6 `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

4.56.2.7 `bool InitVariableValues::addVar (int numberOfVar, InitVarValue ** 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>

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

- [OSOption.h](#)

4.57 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

4.57.1 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.

4.57.2 Member Function Documentation

4.57.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

4.57.2.2 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

4.57.2.3 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>

4.57.2.4 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>

4.57.2.5 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>
-------------	---

4.57.2.6 `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

4.57.2.7 `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>

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

- [OSOption.h](#)

4.58 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

4.58.1 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.

4.58.2 Member Function Documentation

4.58.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

4.58.2.2 `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

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

- [OSOption.h](#)

4.59 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

4.59.1 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.

4.59.2 Member Function Documentation

4.59.2.1 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 "numberOfXXX" attributes and <XXX> children)

4.59.2.2 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

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

- [OSOption.h](#)

4.60 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

4.60.1 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 2174 of file OSInstance.h.

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

- [OSInstance.h](#)

4.61 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

4.61.1 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.

4.61.2 Member Function Documentation**4.61.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.61.2.2 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

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

- [OSOption.h](#)

4.62 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

4.62.1 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.

4.62.2 Member Function Documentation

4.62.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.62.2.2 `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

4.62.2.3 `bool IntegerVariableBranchingWeights::setVar (int numberOfVar, BranchingWeight ** 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 to be that are to be set</i>

4.62.2.4 `bool IntegerVariableBranchingWeights::setVar (int numberOfVar, BranchingWeight ** 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>

4.62.2.5 `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 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>
-------------	---

4.62.2.6 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

4.62.2.7 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>

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

- [OSOption.h](#)

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

4.63.1 Detailed Description

The in-memory representation of an intersection cone.

Definition at line 1324 of file `OSInstance.h`.

4.63.2 Member Function Documentation

4.63.2.1 `virtual std::string IntersectionCone::getConeName () [virtual]`

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.63.2.2 `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](#).

4.63.2.3 `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 "numberOf← XXX" 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

4.63.2.4 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

4.63.3 Member Data Documentation

4.63.3.1 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 1345 of file OSInstance.h.

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

- [OSInstance.h](#)

4.64 Interval Class Reference

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

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

4.64.1 Detailed Description

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

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

- [OSInstance.h](#)

4.65 IntVector Class Reference

an integer Vector data structure

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

Inheritance diagram for IntVector:

Public Member Functions

- [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

4.65.1 Detailed Description

an integer Vector data structure

Definition at line 469 of file OSGeneral.h.

4.65.2 Member Function Documentation

4.65.2.1 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 "numberOf↵ XXX" 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

4.65.2.2 `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

4.65.2.3 `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

4.65.2.4 `bool IntVector::extendIntVector (int i)`

append a value to an [IntVector](#)

Parameters

<i>i</i>	contains the value to be appended
----------	-----------------------------------

4.65.2.5 `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
----------	---

4.65.2.6 `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.

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

- [OSGeneral.h](#)

4.66 IpoptProblem Class Reference

Inheritance diagram for IpoptProblem:

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)

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.

4.66.1 Detailed Description

Definition at line 52 of file OSIpoptSolver.h.

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

- OSIpoptSolver.h

4.67 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

- [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

4.67.1 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.

4.67.2 Member Function Documentation

4.67.2.1 `void IpoptSolver::buildSolverInstance () throw ErrorClass` [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

4.67.2.2 `void IpoptSolver::setSolverOptions () throw ErrorClass` [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

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

- `OSIpoptSolver.h`

4.68 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

4.68.1 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.

4.68.2 Member Function Documentation

4.68.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.68.2.2 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

4.68.2.3 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

4.68.2.4 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
--------------	---

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

- [OSOption.h](#)

4.69 JobOption Class Reference

the [JobOption](#) class.

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

Collaboration diagram for JobOption:

Public Member Functions

- [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.

Public Attributes

- [TimeSpan](#) * [maxTime](#)
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

4.69.1 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.

4.69.2 Member Function Documentation

4.69.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.69.2.2 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

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

- [OSOption.h](#)

4.70 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
- [CPUSpeed](#) * [usedCPUSpeed](#)
a pointer to the [CPUSpeed](#) class
- [CPUNumber](#) * [usedCPUNumber](#)
a pointer to the [CPUNumber](#) class
- [OtherResults](#) * [otherResults](#)
a pointer to the [OtherResults](#) class

4.70.1 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 659 of file OSResult.h.

4.70.2 Member Function Documentation

4.70.2.1 `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 "numberOfXXX" attributes and <XXX> children)

4.70.3 Member Data Documentation

4.70.3.1 StorageCapacity* JobResult::usedMemory

a pointer to the MemorySize class

Definition at line 688 of file OSResult.h.

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

- [OSResult.h](#)

4.71 KnitroProblem Class Reference

Inheritance diagram for KnitroProblem:

Collaboration diagram for KnitroProblem:

Public Member Functions

- [KnitroProblem](#) ([OSInstance](#) *osinstance_, [OSResult](#) *osresult_)
the IpoptProblemclass constructor
- virtual [~KnitroProblem](#) ()
the IpoptProblem class destructor

4.71.1 Detailed Description

Definition at line 87 of file OSKnitroSolver.h.

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

- OSKnitroSolver.h

4.72 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

4.72.1 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.

4.72.2 Member Function Documentation

4.72.2.1 void KnitroSolver::setSolverOptions () throw ErrorClass [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

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

- OSKnitroSolver.h

4.73 LindoSolver Class Reference

the [LindoSolver](#) class solves problems using Lindo.

```
#include <OSLindoSolver.h>
```

Inheritance diagram for LindoSolver:

Collaboration diagram for LindoSolver:

Public Member Functions

- [LindoSolver](#) ()
the [LindoSolver](#) 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.

4.73.1 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.

4.73.2 Member Function Documentation

4.73.2.1 void LindoSolver::setSolverOptions () throw **ErrorClass** [virtual]

The implementation of the virtual functions.

Returns

void.

Implements [DefaultSolver](#).

4.73.2.2 bool LindoSolver::optimize ()

invoke the Lindo API solver

Returns

true if an exception is not thrown.

4.73.2.3 bool LindoSolver::processVariables ()

read the OSiL instance variables and put these into the LINDO API variables

Returns

true if an exception is not thrown.

4.73.2.4 `bool LindoSolver::processConstraints ()`

read the OSiL instance constraints and put these into the LINDO API constraints

Returns

true if an exception is not thrown.

4.73.2.5 `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.

4.73.2.6 `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.

4.73.2.7 `bool LindoSolver::processQuadraticTerms ()`

read the quadratic terms in the model

Returns

true if an exception is not thrown.

4.73.2.8 `bool LindoSolver::processNonlinearExpressions ()`

read the nonlinear terms in the model

Returns

true if an exception is not thrown.

4.73.2.9 `void LindoSolver::lindoAPIErrorCheck (std::string errmsg)` `[protected]`

Lindo's generalized error Reporting function.

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

- OSLindoSolver.h

4.74 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>`.

4.74.1 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.

4.74.2 Member Data Documentation

4.74.2.1 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:

- [OSInstance.h](#)

4.75 LinearMatrixElement Class Reference

a data structure to represent an expression in a linearMatrix element A [LinearMatrixElement](#) is a (finite) sum of Linear↔MatrixElementTerms, with an optional additive constant

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

Collaboration diagram for LinearMatrixElement:

Public Member Functions

- 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.

4.75.1 Detailed Description

a data structure to represent an expression in a linearMatrix element A [LinearMatrixElement](#) is a (finite) sum of Linear↔MatrixElementTerms, with an optional additive constant

Parameters

<i>numberOfVarIdx</i>	gives the number of terms in the expression
-----------------------	---

Definition at line 411 of file OSMatrix.h.

4.75.2 Member Function Documentation

4.75.2.1 bool LinearMatrixElement::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 "numberOf↵XXX" 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

4.75.2.2 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

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

- [OSMatrix.h](#)

4.76 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

- virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [getNode](#)Type ()
- virtual ENUM_MATRIX_TYPE [getMatrix](#)Type ()
- virtual std::string [getNode](#)Name ()
- 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](#) * [value](#)

*The values are expressions of the form $a_0 + a_1 x_{\{i_1\}} * a_2 x_{\{i_2\}} + \dots$*

4.76.1 Detailed Description

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

Definition at line 917 of file OSMatrix.h.

4.76.2 Member Function Documentation

4.76.2.1 virtual `ENUM_MATRIX_CONSTRUCTOR_TYPE` `LinearMatrixElements::getNodeType ()` [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.76.2.2 virtual `ENUM_MATRIX_TYPE` `LinearMatrixElements::getMatrixType ()` [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.76.2.3 virtual `std::string` `LinearMatrixElements::getNodeName ()` [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.76.2.4 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](#).

4.76.2.5 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](#).

4.76.2.6 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](#).

4.76.2.7 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 "numberOf↵XXX" 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

4.76.2.8 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

4.76.3 Member Data Documentation

4.76.3.1 LinearMatrixValues* LinearMatrixElements::value

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 925 of file OSMatrix.h.

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

- [OSMatrix.h](#)

4.77 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

- 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.

4.77.1 Detailed Description

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

Definition at line 373 of file OSMatrix.h.

4.77.2 Member Function Documentation

4.77.2.1 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 "numberOf↵ XXX" 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

4.77.2.2 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

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

- [OSMatrix.h](#)

4.78 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

- 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.
- virtual bool [deepCopyFrom](#) ([LinearMatrixValues](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.78.1 Detailed Description

a data structure to represent the linear expressions in a [LinearMatrixElement](#) object

Definition at line 602 of file OSMatrix.h.

4.78.2 Member Function Documentation

4.78.2.1 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 "numberOf↵ XXX" 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

4.78.2.2 `virtual bool LinearMatrixValues::deepCopyFrom (LinearMatrixValues * that) [virtual]`

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:

- [OSMatrix.h](#)

4.79 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 start↵
Size, int valueSize, int *start, int *index, double *value, int dimension)
Round a double number to the precision specified.

4.79.1 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.

4.79.2 Member Function Documentation

4.79.2.1 `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>x</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>startSize</i>	holds the size of the start array
<i>valueSize</i>	holds the size of the index and value arrays
<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.

4.79.2.2 std::string MathUtil::format_os_dtoa (double x)

Parameters

<i>x</i>	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:

- OSMathUtil.h

4.80 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 `<nI>` elements in the `<matrices>` element.
- [OSMatrix](#) ** [matrix](#)
matrix is a pointer to an array of [OSMatrix](#) object pointers

4.80.1 Detailed Description

The in-memory representation of the `<matrices>` element.

Definition at line 482 of file `OSInstance.h`.

4.80.2 Member Function Documentation

4.80.2.1 `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 "numberOfXXX" 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

4.80.2.2 `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

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

- [OSInstance.h](#)

4.81 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

- 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 bool [expandElements](#) (bool rowMajor)
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).
- 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.

Additional Inherited Members

4.81.1 Detailed Description

a data structure to represent a [MatrixBlock](#) object (derived from [MatrixType](#))

Definition at line 2500 of file OSMatrix.h.

4.81.2 Member Function Documentation

4.81.2.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [MatrixBlock::getNodeTypes](#) () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.81.2.2 virtual std::string [MatrixBlock::getNodeName](#) () [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

4.81.2.3 virtual ENUM_MATRIX_TYPE [MatrixBlock::getMatrixType](#) () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.81.2.4 `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](#).

4.81.2.5 `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](#).

4.81.2.6 `virtual bool MatrixBlock::expandElements (bool rowMajor) [virtual]`

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.

Reimplemented from [MatrixType](#).

4.81.2.7 `MatrixBlock * MatrixBlock::cloneMatrixNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [MatrixNode](#).

4.81.2.8 `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 "numberOf↵XXX" 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

4.81.2.9 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

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

- [OSMatrix.h](#)

4.82 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

- 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](#) * `colOffset`
An array listing the leftmost column of each block within the larger matrix. It is assumed that the blocks are neatly "stacked".
- [IntVector](#) * `rowOffset`
An array listing the top row of each block within the larger matrix.

4.82.1 Detailed Description

a data structure to represent the nonzeros 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 1449 of file OSMatrix.h.

4.82.2 Member Function Documentation

4.82.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixBlocks::getNodeType () [virtual]`

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.82.2.2 `virtual std::string MatrixBlocks::getNodeName () [virtual]`

Returns

the name of the operator

Implements [MatrixNode](#).

4.82.2.3 `virtual ENUM_MATRIX_TYPE MatrixBlocks::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.82.2.4 `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](#).

4.82.2.5 `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](#).

4.82.2.6 **MatrixBlocks * MatrixBlocks::cloneMatrixNode ()** [virtual]

The implementation of the virtual functions.

Returns

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

Implements [MatrixNode](#).

4.82.2.7 **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 "numberOf↵XXX" 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

4.82.2.8 **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

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

- [OSMatrix.h](#)

4.83 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` [lbConIdx](#)
lbConIdx gives a cone that must contain matrixCon - lbMatrix
- `int` [ubMatrixIdx](#)
ubMatrixIdx gives an upper bound for this matrixCon
- `int` [ubConIdx](#)
ubConIdx gives a cone that must contain ubMatrix - matrixCon
- `std::string` [name](#)
an optional name to this [MatrixCon](#)

4.83.1 Detailed Description

The in-memory representation of the `<matrixCon>` element.

Definition at line 1736 of file OSInstance.h.

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

- [OSInstance.h](#)

4.84 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

4.84.1 Detailed Description

The in-memory representation of the `<matrixConstraints>` element.

Definition at line 1787 of file OSInstance.h.

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

- [OSInstance.h](#)

4.85 MatrixConstraintSolution Class Reference

The in-memory representation of the `<MatrixConstraintSolution>` element.

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

Collaboration diagram for MatrixConstraintSolution:

Public Member Functions

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

Public Attributes

- int [numberOfOtherMatrixConstraintResults](#)
numberOfOtherMatrixConstraintResults gives the number of <other> children
- [OSMatrixWithMatrixConIdx](#) ** [matrixCon](#)
matrixCon is an array of pointers to the <matrixCon> children

4.85.1 Detailed Description

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

Definition at line 2047 of file OSResult.h.

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

- [OSResult.h](#)

4.86 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

4.86.1 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.

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

- [OSMatrix.h](#)

4.87 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

- 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](#) * [index](#)
The index array of the (nonzero) elements.

4.87.1 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 248 of file OSMatrix.h.

4.87.2 Member Function Documentation

4.87.2.1 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.

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

- [OSMatrix.h](#)

4.88 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

- virtual bool [deepCopyFrom](#) ([MatrixElementValues](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [numberOfEl](#)
each type of value is stored as an array named "el".

4.88.1 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.

4.88.2 Member Function Documentation

4.88.2.1 virtual bool [MatrixElementValues::deepCopyFrom](#) ([MatrixElementValues](#) * *that*) [virtual]

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. 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 "numberOf↵XXX" 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 A function to make a deep copy of an instance of this class

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

Returns

whether the copy was created successfully

4.88.3 Member Data Documentation**4.88.3.1 int MatrixElementValues::numberOfEI**

each type of value is stored as an array named "ei".

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:

- [OSMatrix.h](#)

4.89 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 [O↔SInstance](#) class created a map of the expression trees, this should be false since the [osExpressionTree](#) is deleted by the [OSInstance](#) object

4.89.1 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 1816 of file `OSInstance.h`.

4.89.2 Member Data Documentation

4.89.2.1 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 1826 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.90 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

4.90.1 Detailed Description

The in-memory representation of the `<matrixExpressions>` element.

Definition at line 1855 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.91 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.

4.91.1 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 208 of file OSExpressionTree.h.

4.91.2 Member Function Documentation

4.91.2.1 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.

4.91.2.2 `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.

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

- [OSExpressionTree.h](#)

4.92 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](#) * > [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` [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
- [MatrixNode](#) ** `m_mChildren`
m_mChildren holds all the children, that is, nodes used in the definition or construction of the current node.

4.92.1 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](#).

4.92.2 Member Function Documentation

4.92.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MatrixNode::getNodeType ()` [virtual]

Returns

the value of `nType`

Reimplemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [MixedRowReference](#)↔[MatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrix](#)↔[Elements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.2 `virtual ENUM_MATRIX_TYPE MatrixNode::getMatrixType ()` [pure virtual]

Returns

the type of the matrix elements

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [MixedRowReference](#)↔[MatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrix](#)↔[Elements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.3 `virtual std::string MatrixNode::getNodeName ()` [pure virtual]

Returns

the name of the matrix constructor

Implemented in [MatrixBlock](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [MixedRowReference](#)↔[MatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrix](#)↔[Elements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.4 `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](#), [OSMatrixWithMatrixConIdx](#), [OSMatrixWithMatrixObjIdx](#), [OSMatrixWithMatrixVarIdx](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [MixedRowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.5 `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.

4.92.2.6 `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.

4.92.2.7 `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.

4.92.2.8 `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](#), [OSMatrixWithMatrixConIdx](#), [OSMatrixWithMatrixObjIdx](#), [OSMatrixWithMatrixVarIdx](#), [OSMatrix](#), [BaseMatrix](#), [MatrixBlocks](#), [MatrixTransformation](#), [MixedRowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.9 `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](#), [MixedRowReferenceMatrixElements](#), [ConReferenceMatrixElements](#), [ObjReferenceMatrixElements](#), [GeneralMatrixElements](#), [LinearMatrixElements](#), [VarReferenceMatrixElements](#), and [ConstantMatrixElements](#).

4.92.2.10 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 "numberOfXXX" 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

4.92.2.11 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

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

- [OSMatrix.h](#)

4.93 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 [orderConelIdx](#)
orderConelIdx 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

4.93.1 Detailed Description

The in-memory representation of the `<matrixObj>` element.

Definition at line 1661 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.94 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

4.94.1 Detailed Description

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

Definition at line 1708 of file OSInstance.h.

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

- [OSInstance.h](#)

4.95 MatrixObjectiveSolution Class Reference

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

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

Collaboration diagram for MatrixObjectiveSolution:

Public Member Functions

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

Public Attributes

- int [numberOfOtherMatrixObjectiveResults](#)
numberOfOtherMatrixObjectiveResults gives the number of <other> children
- [OSMatrixWithMatrixObjIdx](#) ** [matrixObj](#)
matrixObj is an array of pointers to the <matrixObj> children

4.95.1 Detailed Description

The in-memory representation of the `<MatrixVariableSolution>` element.

Definition at line 2021 of file OSResult.h.

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

- [OSResult.h](#)

4.96 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

4.96.1 Detailed Description

The in-memory representation of the `<matrixProgramming>` element.

Definition at line 1882 of file OSInstance.h.

4.96.2 Member Function Documentation

4.96.2.1 `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 "numberOf↵XXX" 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

4.96.2.2 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

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

- [OSInstance.h](#)

4.97 MatrixProgrammingSolution Class Reference

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

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

Collaboration diagram for MatrixProgrammingSolution:

Public Member Functions

- [MatrixProgrammingSolution](#) ()
The MatrixProgramming class constructor.
- [~MatrixProgrammingSolution](#) ()
The MatrixProgramming class destructor.
- bool [isEqual](#) (MatrixProgrammingSolution *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](#) (MatrixProgrammingSolution *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- int [numberOfOtherMatrixProgrammingResults](#)
the number of <other> child elements>

- [OtherSolutionResult](#) ** *other*
a pointer to the array of <other> children
- [MatrixVariableSolution](#) * *matrixVariables*
a pointer to the *matrixVariables* object
- [MatrixObjectiveSolution](#) * *matrixObjectives*
a pointer to the *matrixObjectives* object
- [MatrixConstraintSolution](#) * *matrixConstraints*
a pointer to the *matrixConstraints* object

4.97.1 Detailed Description

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

Definition at line 2075 of file OSResult.h.

4.97.2 Member Function Documentation

4.97.2.1 `bool MatrixProgrammingSolution::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 "numberOf↵XXX" 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

4.97.2.2 `bool MatrixProgrammingSolution::deepCopyFrom (MatrixProgrammingSolution * 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:

- [OSResult.h](#)

4.98 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

- virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [getNode](#)Type ()
- virtual std::string [getNode](#)Name ()
- virtual ENUM_MATRIX_TYPE [getMatrix](#)Type ()
- 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.

4.98.1 Detailed Description

a data structure to represent the nonzeros of a matrix by transformation from other (previously defined) matrices

Definition at line 1358 of file OSMatrix.h.

4.98.2 Member Function Documentation

4.98.2.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [MatrixTransformation::getNode](#)Type () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.98.2.2 virtual std::string [MatrixTransformation::getNode](#)Name () [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

4.98.2.3 `virtual ENUM_MATRIX_TYPE MatrixTransformation::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.98.2.4 `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](#).

4.98.2.5 `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](#).

4.98.2.6 `MatrixTransformation * MatrixTransformation::cloneMatrixNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [MatrixNode](#).

4.98.2.7 `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 "numberOf↵XXX" 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

4.98.2.8 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

4.98.3 Member Data Documentation

4.98.3.1 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 1370 of file OSMatrix.h.

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

- [OSMatrix.h](#)

4.99 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

- 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 [printExpandedMatrix](#) (bool rowMajor)
a utility routine to print the expanded matrix or block.
- 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
- virtual bool [expandElements](#) (bool rowMajor)

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](#) * [convertToOtherMajor](#) (bool isColumnMajor)

A method to convert a matrix to the other major.
- bool [processBlockPartition](#) ()

A method to determine the block structure of a matrixType as defined by the <blocks> element or elements.
- virtual bool [processBlocks](#) (bool rowMajor, ENUM_MATRIX_SYMMETRY [symmetry](#))

A method to process a matrixType into a block structure defined by the <blocks> element or elements.
- virtual bool [processBlocks](#) (int *rowOffset, int rowOffsetSize, int *colOffset, int colOffsetSize, bool rowMajor, ENUM_MATRIX_SYMMETRY [symmetry](#))

A method to process a matrixType into a specific block structure.
- [GeneralSparseMatrix](#) * [extractBlock](#) (int firstrow, int firstcol, int lastrow, int lastcol, 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).
- [ExpandedMatrixBlocks](#) * [getBlocks](#) (int *rowPartition, int rowPartitionSize, int *colPartition, int colPartitionSize, bool rowMajor, bool appendToBlockArray)

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).
- [ExpandedMatrixBlocks](#) * [disassembleMatrix](#) (int *rowPartition, int rowPartitionSize, int *colPartition, int colPartitionSize, bool rowMajor, ENUM_MATRIX_SYMMETRY [symmetry](#))

A method to disassemble a [MatrixType](#) into individual blocks of specific structure.
- 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.
- [GeneralSparseMatrix](#) * [ExpandedMatrixInRowMajorForm](#)

The matrix can be held in expanded form by rows or by columns and in a number of ways stored by blocks.

4.99.1 Detailed Description

a data structure to represent a [MatrixType](#) object (from which we derive [OSMatrix](#) and [MatrixBlock](#))

Definition at line 1889 of file `OSMatrix.h`.

4.99.2 Member Function Documentation

4.99.2.1 `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](#).

4.99.2.2 `bool MatrixType::printExpandedMatrix (bool rowMajor)`

a utility routine to print the expanded matrix or block.

Parameters

<i>rowMajor</i>	controls whether the matrix should be printed in row or column major.
-----------------	---

Returns

whether the operation was successful

Remarks

if the expanded matrix does not exist, return false

4.99.2.3 `int MatrixType::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)

4.99.2.4 `int* MatrixType::getRowPartition ()`

get the row partition of the matrix

Returns

a vector of int corresponding to the partition points of the rows of this matrix

4.99.2.5 `int MatrixType::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)

4.99.2.6 `int* MatrixType::getColumnPartition ()`

get the column partition of the matrix

Returns

a vector of int corresponding to the partition points of the columns of this matrix

4.99.2.7 `virtual bool MatrixType::expandElements (bool rowMajor)` `[virtual]`

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.

Reimplemented in [MatrixBlock](#), and [OSMatrix](#).

4.99.2.8 `GeneralSparseMatrix* MatrixType::convertToOtherMajor (bool isColumnMajor)`

A method to convert a matrix to the other major.

Parameters

<i>isColumnMajor</i>	holds whether the matrix is stored by column. If true, the matrix is converted to row major form. If false, the matrix is stored by row and is converted to column major.
----------------------	---

Returns

A pointer to the matrix in the other major. Return null if input matrix not valid.

4.99.2.9 `bool MatrixType::processBlockPartition ()`

A method to determine the block structure of a matrixType as defined by the <blocks> element or elements.

If multiple partitions are found, they are consolidated into the coarsest partition (intersection of all partition sets).

Returns

whether the operation was successful

Remarks

This method sets `m_iRowPartition`, `m_iRowPartitionSize`, `m_iColumnPartition` and `m_iColumnPartitionSize`, but does not expand any of the blocks themselves.

4.99.2.10 `virtual bool MatrixType::processBlocks (bool rowMajor, ENUM_MATRIX_SYMMETRY symmetry) [virtual]`

A method to process a matrixType into a block structure defined by the <blocks> element or elements.

Parameters

<i>rowMajor</i>	indicates whether the blocks should be stored in row major (if true) or column major.
<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 `extractBlocks` (see below). It is possible (though probably not advisable) to maintain multiple decompositions with different row and column partitions (see next method)

4.99.2.11 `virtual bool MatrixType::processBlocks (int * rowOffset, int rowOffsetSize, int * colOffset, int colOffsetSize, bool rowMajor, ENUM_MATRIX_SYMMETRY symmetry) [virtual]`

A method to process a matrixType into a specific block structure.

Parameters

<i>rowOffset</i>	defines a partition of the matrix rows into the blocks
<i>rowOffsetSize</i>	gives the number of elements in the rowOffset array
<i>colOffset</i>	defines a partition of the matrix columns into the blocks
<i>colOffsetSize</i>	gives the number of elements in the colOffset array
<i>rowMajor</i>	controls whether the blocks are stored by row or by column
<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

4.99.2.12 `GeneralSparseMatrix* MatrixType::extractBlock (int firstrow, int firstcol, int lastrow, int lastcol, 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>lastrow</i>	gives the last row of the block
<i>lastcol</i>	gives the last column of 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

Remarks

Before extracting a block it is necessary to call [processBlocks\(\)](#) in order to make sure that a block partition conformal to the block dimensions and positions has been defined or prepared.

4.99.2.13 `ExpandedMatrixBlocks* MatrixType::getBlocks (int * rowPartition, int rowPartitionSize, int * colPartition, int colPartitionSize, bool rowMajor, bool appendToBlockArray)`

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>rowPartition</i>	defines the partition of the set of rows into the blocks
<i>rowPartitionSize</i>	gives the size of the rowPartition array
<i>colPartition</i>	defines the partition of the set of columns into the blocks
<i>colPartitionSize</i>	gives the size of the colPartition array
<i>rowMajor</i>	indicates whether the blocks are stored in row major form or not.
<i>appendToBlock↔ Array</i>	determines whether the blocks should be created if not found.

Returns

the blocks as an [ExpandedMatrixBlocks](#) object, which is essentially an array of general sparse matrices.

Remarks

If blocks corresponding to the indicated partition do not exist, this method can try to create them. This can be quite storage-intensive and is controlled by the parameter *appendToBlockArray*. If no blocks found (and appending is inhibited) return NULL.

4.99.2.14 `ExpandedMatrixBlocks* MatrixType::disassembleMatrix (int * rowPartition, int rowPartitionSize, int * colPartition, int colPartitionSize, bool rowMajor, ENUM_MATRIX_SYMMETRY symmetry)`

A method to disassemble a [MatrixType](#) into individual blocks of specific structure.

Parameters

<i>rowPartition</i>	defines the partition of the set of rows into the blocks
<i>rowPartitionSize</i>	gives the size of the rowPartition array
<i>colPartition</i>	defines the partition of the set of columns into the blocks
<i>colPartitionSize</i>	gives the size of the colPartition array
<i>rowMajor</i>	indicates whether the blocks are stored in row major form or not.
<i>symmetry</i>	determines what kind of symmetry to use in representing the blocks.

Returns

the blocks as an [ExpandedMatrixBlocks](#) object, which is essentially an array of general sparse matrices.

4.99.2.15 `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 "numberOf↔XXX" 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

4.99.2.16 `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

4.99.3 Member Data Documentation

4.99.3.1 `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 1896 of file `OSMatrix.h`.

4.99.3.2 `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 1902 of file `OSMatrix.h`.

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

- [OSMatrix.h](#)

4.100 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 [lbConIdx](#)
lbConIdx gives a cone that must contain matrixVar - lbMatrix
- int [ubMatrixIdx](#)
ubMatrixIdx gives an upper bound for this matrixVar
- int [ubConIdx](#)
ubConIdx 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).

4.100.1 Detailed Description

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

Definition at line 1579 of file OSInstance.h.

4.100.2 Member Data Documentation

4.100.2.1 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 1616 of file OSInstance.h.

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

- [OSInstance.h](#)

4.101 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

4.101.1 Detailed Description

The in-memory representation of the `<matrixVariables>` element.

Definition at line 1635 of file OSInstance.h.

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

- [OSInstance.h](#)

4.102 MatrixVariableSolution Class Reference

The in-memory representation of the `<MatrixVariableSolution>` element.

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

Collaboration diagram for MatrixVariableSolution:

Public Member Functions

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

Public Attributes

- int [numberOfOtherMatrixVariableResults](#)
numberOfOtherMatrixVariableResults gives the number of <other> children
- [MatrixVariableValues](#) * [values](#)
values is pointer to the <values> child
- [OtherMatrixVariableResult](#) ** [other](#)
other is a pointer to an array of <other> children

4.102.1 Detailed Description

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

Definition at line 1992 of file OSResult.h.

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

- [OSResult.h](#)

4.103 MatrixVariableValues Class Reference

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

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

Collaboration diagram for MatrixVariableValues:

Public Member Functions

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

Public Attributes

- int [numberOfMatrixVar](#)
numberOfMatrixVar gives the number of <matrixVar> children
- [OSMatrixWithMatrixVarIdx](#) ** [matrixVar](#)
matrixVar is an array of pointers to the <matrixVar> children

4.103.1 Detailed Description

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

Definition at line 1914 of file OSResult.h.

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

- [OSResult.h](#)

4.104 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

4.104.1 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 OOption.h.

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

- [OOption.h](#)

4.105 MinCPUNumber Class Reference

the [MinCPUNumber](#) class.

```
#include <OOption.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

4.105.1 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](#).

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

- [OSOption.h](#)

4.106 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

4.106.1 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](#).

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

- [OSOption.h](#)

4.107 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

4.107.1 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](#).

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

- [OSOption.h](#)

4.108 MinMemorySize Class Reference

the [MinMemorySize](#) class.

```
#include <OSOption.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

4.108.1 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](#).

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

- [OSOption.h](#)

4.109 MixedRowReferenceMatrixElements 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 [MixedRowReferenceMatrixElements](#):

Collaboration diagram for [MixedRowReferenceMatrixElements](#):

Public Member Functions

- 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 [MixedRowReferenceMatrixElements](#) * [cloneMatrixNode](#) ()
- bool [isEqual](#) ([MixedRowReferenceMatrixElements](#) *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](#) ([MixedRowReferenceMatrixElements](#) *that)
A function to make a deep copy of an instance of this class.

Public Attributes

- [ConReferenceMatrixValues](#) * [value](#)

The row references (indexes of core rows plus value type) of the elements.

4.109.1 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 1264 of file OSMatrix.h.

4.109.2 Member Function Documentation

4.109.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE MixedRowReferenceMatrixElements::getNodeType () [virtual]`

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.109.2.2 `virtual ENUM_MATRIX_TYPE MixedRowReferenceMatrixElements::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.109.2.3 `virtual std::string MixedRowReferenceMatrixElements::getNodeName () [virtual]`

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.109.2.4 `virtual std::string MixedRowReferenceMatrixElements::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](#).

4.109.2.5 **virtual bool MixedRowReferenceMatrixElements::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](#).

4.109.2.6 **virtual MixedRowReferenceMatrixElements* MixedRowReferenceMatrixElements::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](#).

4.109.2.7 **bool MixedRowReferenceMatrixElements::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 "numberOf↔XXX" 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

4.109.2.8 **bool MixedRowReferenceMatrixElements::deepCopyFrom (MixedRowReferenceMatrixElements * *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

4.109.3 Member Data Documentation

4.109.3.1 ConReferenceMatrixValues* MixedRowReferenceMatrixElements::value

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 1275 of file OSMatrix.h.

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

- [OSMatrix.h](#)

4.110 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](#)

4.110.1 Detailed Description

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

Definition at line 410 of file OSInstance.h.

4.110.2 Member Data Documentation

4.110.2.1 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.

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

- [OSInstance.h](#)

4.111 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

4.111.1 Detailed Description

The in-memory representation of the `<nonlinearExpressions>` element.

Definition at line 452 of file OSInstance.h.

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

- [OSInstance.h](#)

4.112 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

4.112.1 Detailed Description

The [NonnegativeCone](#) Class.

Remarks

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

Definition at line 609 of file OSInstance.h.

4.112.2 Member Function Documentation

4.112.2.1 virtual std::string [NonnegativeCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.112.2.2 `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](#).

4.112.2.3 `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 "numberOfXXX" 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

4.112.2.4 `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:

- [OSInstance.h](#)

4.113 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

4.113.1 Detailed Description

The [NonpositiveCone](#) Class.

Remarks

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

Definition at line 667 of file OSInstance.h.

4.113.2 Member Function Documentation

4.113.2.1 virtual std::string NonpositiveCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.113.2.2 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](#).

4.113.2.3 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 "numberOf↵XXX" 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

4.113.2.4 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:

- [OSInstance.h](#)

4.114 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

4.114.1 Detailed Description

The in-memory representation of the objective function `<coef>` element.

Definition at line 110 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.115 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

4.115.1 Detailed Description

The in-memory representation of the `<obj>` element.

Definition at line 141 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.116 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

4.116.1 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.

4.116.2 Member Function Documentation

4.116.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.116.2.2 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

4.116.2.3 bool ObjectiveOption::setOther (int *numberOfOptions*, OtherObjectiveOption ** *other*)

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

Parameters

<i>numberOf↵ Options</i>	number of <other> elements to be set
<i>other</i>	the array of <other> elements that are to be set

4.116.2.4 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
--------------	--

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

- [OSOption.h](#)

4.117 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

4.117.1 Detailed Description

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

Definition at line 188 of file OSInstance.h.

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

- [OSInstance.h](#)

4.118 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

4.118.1 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 1537 of file OSResult.h.

4.118.2 Member Function Documentation

4.118.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.119 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.

4.119.1 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 1332 of file OSResult.h.

4.119.2 Member Function Documentation

4.119.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.120 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

- 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 [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

- [ObjReferenceMatrixValues](#) * *value*

The objective references (indexes of core objectives) of the elements.

4.120.1 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 1091 of file OSMatrix.h.

4.120.2 Member Function Documentation

4.120.2.1 `virtual ENUM_MATRIX_CONSTRUCTOR_TYPE ObjReferenceMatrixElements::getNodeType () [virtual]`

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.120.2.2 `virtual ENUM_MATRIX_TYPE ObjReferenceMatrixElements::getMatrixType () [virtual]`

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.120.2.3 `virtual std::string ObjReferenceMatrixElements::getNodeName () [virtual]`

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.120.2.4 `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](#).

4.120.2.5 `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](#).

4.120.2.6 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](#).

4.120.2.7 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 "numberOf←XXX" 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

4.120.2.8 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

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

- [OSMatrix.h](#)

4.121 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

- 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.
- virtual bool [deepCopyFrom](#) (ObjReferenceMatrixValues *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.121.1 Detailed Description

to represent the nonzeros in an objReferenceMatrix element

Definition at line 675 of file OSMatrix.h.

4.121.2 Member Function Documentation

4.121.2.1 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 "numberOfXXX" 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

4.121.2.2 virtual bool ObjReferenceMatrixValues::deepCopyFrom (ObjReferenceMatrixValues * *that*) [virtual]

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:

- [OSMatrix.h](#)

4.122 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

4.122.1 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 1281 of file OSResult.h.

4.122.2 Member Function Documentation

4.122.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.123 OptimizationOption Class Reference

the [OptimizationOption](#) class.

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

Collaboration diagram for OptimizationOption:

Public Member Functions

- [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.

Public Attributes

- 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

4.123.1 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.

4.123.2 Member Function Documentation

4.123.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.123.2.2 `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

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

- [OSOption.h](#)

4.124 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

4.124.1 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 2473 of file OSResult.h.

4.124.2 Member Function Documentation**4.124.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.125 OptimizationSolution Class Reference

The [OptimizationSolution](#) Class.

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

Collaboration diagram for OptimizationSolution:

Public Member Functions

- [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.

Public Attributes

- 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

- [ConstraintSolution](#) * `constraints`

constraints holds the solution information for the constraints

- [ObjectiveSolution](#) * `objectives`

objectives holds the solution information for the objectives

- [MatrixProgrammingSolution](#) * `matrixProgramming`

matrixProgramming holds the solution information for the matrix programming components: matrix variables, matrix objectives and matrix constraints

- [OtherSolutionResults](#) * `otherSolutionResults`

otherSolutionResults is a pointer to an [OtherSolutionResults](#) object that is associated with this optimization solution

4.125.1 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 2263 of file OSResult.h.

4.125.2 Member Function Documentation

4.125.2.1 `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 "numberOfXXX" attributes and <XXX> children)
-------------------	--

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

- [OSResult.h](#)

4.126 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 [numberOfSubstatus](#)
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

4.126.1 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 792 of file OSResult.h.

4.126.2 Member Function Documentation

4.126.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.127 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

4.127.1 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 743 of file OSResult.h.

4.127.2 Member Function Documentation

4.127.2.1 `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 "numberOf↔ XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.128 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,.

4.128.1 Detailed Description

The [OrthantCone](#) Class.

Remarks

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

Definition at line 726 of file OSInstance.h.

4.128.2 Member Function Documentation

4.128.2.1 virtual std::string [OrthantCone::getConeName](#) () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.128.2.2 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](#).

4.128.2.3 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 "numberOf↵XXX" 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

4.128.2.4 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

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

- [OSInstance.h](#)

4.129 OS_AMPL_SUFFIX Struct Reference

Collaboration diagram for OS_AMPL_SUFFIX:

4.129.1 Detailed Description

Definition at line 70 of file OSnI2OS.h.

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

- OSnI2OS.h

4.130 OSCommandLine Class Reference

This class is used to store command line options for the OSSolverService executable and to provide methods to manipulate them.

```
#include <OSCommandLine.h>
```

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

4.130.1 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.

4.130.2 Member Function Documentation

4.130.2.1 void OSCommandLine::convertSolverNameToUpperCase ()

...

or to upper case

4.130.3 Member Data Documentation

4.130.3.1 OSInstance* OSCommandLine::osinstance

osinstance is a representation of the instance in [OSInstance](#) format

Definition at line 42 of file OSCommandLine.h.

4.130.3.2 OSOption* OSCommandLine::osoption

osoption is a representation of the solver options in [OSOption](#) format

Definition at line 47 of file OSCommandLine.h.

4.130.3.3 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.

4.130.3.4 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.

4.130.3.5 `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`.

4.130.3.6 `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`.

4.130.3.7 `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`.

4.130.3.8 `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:

- [OSCommandLine.h](#)

4.131 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.

4.131.1 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.

4.131.2 Constructor & Destructor Documentation

4.131.2.1 OSCommandLineReader::OSCommandLineReader ()

[OSCommandLineReader](#) class constructor.

Parameters

<code>osss</code>	is the command line to be parsed
-------------------	----------------------------------

4.131.3 Member Function Documentation

4.131.3.1 OSCommandLine* OSCommandLineReader::readCommandLine (const std::string & osss) throw ErrorClass)

Get an [OSCommandLine](#) object from a command line string.

Parameters

<code>osss</code>	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)

4.131.3.2 OSCommandLine* OSCommandLineReader::parseString (const std::string & osss) throw ErrorClass)

Parse a string and store it into an [OSCommandLine](#) object.

Parameters

<code>osss</code>	a command line string.
-------------------	------------------------

Returns

the [OSCommandLine](#) object corresponding to the command line string.

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

- [OSCommandLineReader.h](#)

4.132 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

4.132.1 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](#).

4.132.2 Member Data Documentation

4.132.2.1 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 getVariableIndicesMap() has been called

Definition at line 70 of file OSExpressionTree.h.

4.132.2.2 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.

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

- [OSExpressionTree.h](#)

4.133 OSgams2osil Class Reference

Creating an [OSInstance](#) from a GAMS model given as GAMS Modeling Object (GMO).

```
#include <OSgams2osil.hpp>
```

Collaboration diagram for OSgams2osil:

Public Member Functions

- 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.

4.133.1 Detailed Description

Creating an [OSInstance](#) from a GAMS model given as GAMS Modeling Object (GMO).

Definition at line 22 of file OSgams2osil.hpp.

4.133.2 Member Function Documentation

4.133.2.1 bool OSgams2osil::createOSInstance ()

Creates an [OSInstance](#) from the GAMS smag instance representation.

Returns

whether the instance is created successfully.

4.133.2.2 OSInstance* OSgams2osil::takeOverOSInstance ()

Gives [OSInstance](#) and ownership to calling function.

This object forgets about the created instance.

4.133.2.3 OSInstance* OSgams2osil::getOSInstance () [inline]

Gives OSInstances but keeps ownership.

Destruction will destruct [OSInstance](#).

Definition at line 55 of file OSgams2osil.hpp.

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

- OSgams2osil.hpp

4.134 OSGeneral Class Reference**4.134.1 Detailed Description**

Definition at line 969 of file OSGeneral.h.

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

- [OSGeneral.h](#)

4.135 OSgLParserData Class Reference

The [OSgLParserData](#) Class.

```
#include <OSgLParserData.h>
```

Collaboration diagram for OSgLParserData:

Public Member Functions

- [OSgLParserData](#) ()
the [OSgLParserData](#) class constructor

Public Attributes

- int * [osglIntArray](#)
data structure to process an [IntVector](#) and hold the data temporarily
- std::string [fileName](#)

- data structure to process a [GeneralFileHeader](#) and hold the data temporarily*

 - void * [scanner](#)

scanner is used to store data in a reentrant lexer we use this to pass an [OSgLParseData](#) 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
 - [OSMatrix](#) ** [matrix](#)

We need to hold an array of <matrix> elements temporarily.
 - [OSMatrixWithMatrixVarIdx](#) ** [matrixWithMatrixVarIdx](#)

There are also other variants of these ...
 - int * [matrixVarIndexes](#)

In order to use synergies in the parser, we store matrixXXXIdx in a separate vector.
 - int [matrixCounter](#)

We also need to keep track locally of the number of matrices.
 - int [nonzeroCounter](#)

Linear matrices need a counter to count the number of terms within each element.
 - [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< int * > [rowOffsets](#)

Vectors to hold rowOffset and colOffset arrays in a place where they are easily accessible while the <block> children are processed.
 - bool [numberOfBlocksPresent](#)

Data elements for parsing number-valued attributes and elements.
 - bool [baseTransposePresent](#)

Data elements for parsing string-valued attributes and text elements.

4.135.1 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](#).

4.135.2 Member Data Documentation

4.135.2.1 [OSMatrixWithMatrixVarIdx](#)** [OSgLParseData::matrixWithMatrixVarIdx](#)

There are also other variants of these ...

Definition at line 85 of file [OSgLParseData.h](#).

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

- [OSgLParseData.h](#)

4.136 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

4.136.1 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.

4.136.2 Member Function Documentation

4.136.2.1 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](#).

4.136.2.2 `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](#).

4.136.2.3 `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](#).

4.136.2.4 `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](#).

4.136.2.5 `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](#).

4.136.2.6 `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:

- [OShL.h](#)

4.137 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 [osillineno](#)
if there is a parser error, osillineno 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.

- 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 [stageVariablesOrdered](#)
...we need to track whether the variables, constraints, objectives are given in temporal order...
- int [stageVariableStartIdx](#)
...we need to track the first variable, constraint, objective...
- int [stagevarcount](#)
...and we need to track the number of variables we have seen
- int [nvarcovered](#)
these two integers track how many variables and constraints have been assigned to a stage; this is used for consistency checks.
- 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
- bool [ignoreDataAfterErrors](#)
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

4.137.1 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.

4.137.2 Member Data Documentation

4.137.2.1 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.

4.137.2.2 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.

4.137.2.3 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.

4.137.2.4 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.

4.137.2.5 int OSiLParserData::stageVariableStartIdx

...we need to track the first variable, constraint, objective...

Definition at line 107 of file OSiLParserData.h.

4.137.2.6 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.

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

- [OSiLParserData.h](#)

4.138 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.

4.138.1 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.

4.138.2 Member Function Documentation

4.138.2.1 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:

- [OSiLReader.h](#)

4.139 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

4.139.1 Detailed Description

Take an [OSInstance](#) object and write a string that validates against the OSiL schema.

Definition at line 29 of file OSiLWriter.h.

4.139.2 Member Function Documentation

4.139.2.1 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.

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

- [OSiLWriter.h](#)

4.140 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 [getNumberOfSemiIntegerVariables](#) ()
getNumberOfSemiIntegerVariables
- 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.

- int [getNumberOfValuesForMatrix](#) (int n)
Get the number of (nonzero) values in the matrix.
- std::string [getMatrixName](#) (int n)
Get the name of the matrix.
- bool [matrixHasBase](#) (int n)
Several tools to parse the constructor list of a matrix.
- [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](#) ()
- std::map< int, [MatrixExpressionTree](#) * > [getAllMatrixExpressionTreesMod](#) ()
- int * [getMatrixExpressionTreeIndexes](#) ()
Get all the matrix expression tree indexes, i.e.
- int [getNumberOfMatrixExpressionTreeIndexes](#) ()
Get the number of unique matrix expression tree indexes.
- std::string [getTimeDomainFormat](#) ()
Get the format of the time domain ("stages"/"interval")
- int [getTimeDomainStageNumber](#) ()
Get the number of stages that make up the time domain.
- std::string * [getTimeDomainStageNames](#) ()
Get the names of the stages (NULL or empty string ("") if a stage has not been given a name.
- int * [getTimeDomainStageNumberOfVariables](#) ()

- Get the number of variables contained in each time stage.*

 - int * [getTimeDomainStageNumberOfConstraints](#) ()
- Get the number of constraints contained in each time stage.*

 - int * [getTimeDomainStageNumberOfObjectives](#) ()
- Get the number of objectives contained in each time stage.*

 - int ** [getTimeDomainStageVarList](#) ()
- Get the list of variables in each stage.*

 - int ** [getTimeDomainStageConList](#) ()
- Get the list of constraints in each stage.*

 - int ** [getTimeDomainStageObjList](#) ()
- Get the list of objectives in each stage.*

 - double [getTimeDomainIntervalStart](#) ()
- Get the start for the time domain interval.*

 - double [getTimeDomainIntervalHorizon](#) ()
- Get the horizon for the time domain interval.*

 - bool [setInstanceName](#) (std::string name)
- set the instance name.*

 - bool [setInstanceSource](#) (std::string source)
- set the instance source.*

 - bool [setInstanceDescription](#) (std::string description)
- set the instance description.*

 - bool [setInstanceCreator](#) (std::string fileCreator)
- set the instance creator.*

 - 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 innumberOfChildren, [MatrixNode](#) **m_children)
 - 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 distortionMatrixIdx, double normFactor, int axisDirection, double pNorm, 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 srbitrary 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 srbitrary 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.

4.140.1 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 2262 of file OSInstance.h.

4.140.2 Member Function Documentation

4.140.2.1 `std::string OSInstance::getInstanceName ()`

Get instance name.

Returns

instance name. Null or empty std::string ("") if there is no instance name.

4.140.2.2 `std::string OSInstance::getInstanceSource ()`

Get instance source.

Returns

instance source. Null or empty `std::string ("")` if there is no instance source.

4.140.2.3 `std::string OSInstance::getInstanceDescription ()`

Get instance description.

Returns

instance description. Null or empty `std::string ("")` if there is no instance description.

4.140.2.4 `std::string OSInstance::getInstanceCreator ()`

Get instance fileCreator.

Returns

instance fileCreator. Null or empty `std::string ("")` if there is no instance file creator.

4.140.2.5 `std::string OSInstance::getInstanceLicence ()`

Get instance licence.

Returns

instance licence. Null or empty `std::string ("")` if there is no instance licence.

4.140.2.6 `int OSInstance::getVariableNumber ()`

Get number of variables.

Returns

number of variables.

4.140.2.7 `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.
------------------	--

4.140.2.8 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.
------------------	--

4.140.2.9 int OSInstance::getNumberOfIntegerVariables ()

getNumberOfIntegerVariables

Returns

an integer which is the number of I variables.

4.140.2.10 `int OSInstance::getNumberOfBinaryVariables ()`

`getNumberOfBinaryVariables`

Returns

an integer which is the number of B variables.

4.140.2.11 `int OSInstance::getNumberOfSemiContinuousVariables ()`

`getNumberOfSemiContinuousVariables`

Returns

an integer which is the number of D variables.

4.140.2.12 `int OSInstance::getNumberOfSemiIntegerVariables ()`

`getNumberOfSemiIntegerVariables`

Returns

an integer which is the number of J variables.

4.140.2.13 `int OSInstance::getNumberOfStringVariables ()`

`getNumberOfStringVariables`

Returns

an integer which is the number of S variables.

4.140.2.14 `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.
------------------	--

4.140.2.15 `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.
------------------	--

4.140.2.16 int OSInstance::getObjectiveNumber ()

Get number of objectives.

Returns

number of objectives.

4.140.2.17 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.
------------------	---

4.140.2.18 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.
------------------	---

4.140.2.19 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.
------------------	---

4.140.2.20 `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.
------------------	---

4.140.2.21 `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.
------------------	---

4.140.2.22 `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.
------------------	---

4.140.2.23 double OSInstance::getDenseObjectiveCoefficients ()**

getDenseObjectiveCoefficients.

Returns

an vector of pointers, each pointer points to a dense vector of ObjectiveCoefficients.

4.140.2.24 int OSInstance::getConstraintNumber ()

Get number of constraints.

Returns

number of constraints.

4.140.2.25 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.
------------------	--

4.140.2.26 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.
------------------	--

4.140.2.27 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.
------------------	--

4.140.2.28 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.
------------------	--

4.140.2.29 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.
------------------	--

4.140.2.30 int OSInstance::getLinearConstraintCoefficientNumber ()

Get number of specified (usually nonzero) linear constraint coefficient values.

Returns

number of specified (usually nonzero) linear constraint coefficient values.

4.140.2.31 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.
------------------	---

4.140.2.32 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

4.140.2.33 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

4.140.2.34 int OSInstance::getNumberOfQuadraticTerms ()

Get the number of specified (usually nonzero) qTerms in the quadratic coefficients.

Returns

qTerm number.

4.140.2.35 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

4.140.2.36 `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.

4.140.2.37 `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,

4.140.2.38 `int OSInstance::getNumberOfNonlinearExpressions ()`

Get number of nonlinear expressions.

Returns

the number of nonlinear expressions.

4.140.2.39 `NI** OSInstance::getNonlinearExpressions ()`

Get the pointers to the roots of all expression trees.

Returns

an array of pointers to [NI](#) objects

4.140.2.40 `ScalarExpressionTree* OSInstance::getNonlinearExpressionTree (int rowIdx)`

Get the expression tree for a given row index.

Returns

an expression tree

4.140.2.41 `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

4.140.2.42 `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](#)

4.140.2.43 `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

4.140.2.44 `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

4.140.2.45 `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

4.140.2.46 `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

4.140.2.47 `int OSInstance::getNumberOfNonlinearObjectives ()`

Returns

the number of [Objectives](#) with a nonlinear term

4.140.2.48 `int OSInstance::getNumberOfNonlinearConstraints ()`

Returns

the number of [Constraints](#) with a nonlinear term

4.140.2.49 `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

4.140.2.50 `std::map<int, ScalarExpressionTree* > OSInstance::getAllNonlinearExpressionTreesMod ()`

Returns

a map: the key is the row index and the value is the corresponding expression tree

4.140.2.51 `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.

4.140.2.52 `int OSInstance::getNumberOfNonlinearExpressionTreeIndexes ()`

Get the number of unique nonlinear expression tree indexes.

Returns

the number of unique nonlinear expression tree indexes.

4.140.2.53 `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).

4.140.2.54 `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).

4.140.2.55 `int OSInstance::getMatrixNumber ()`

Get the number of matrices.

Returns

the number of matrices.

4.140.2.56 `ENUM_MATRIX_TYPE OSInstance::getMatrixType (int n)`

Get the matrix type.

Returns

the type of elements contained in the matrix.

Parameters

n	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](#)

4.140.2.57 `ENUM_MATRIX_SYMMETRY OSInstance::getMatrixSymmetry (int n)`

Get the matrix symmetry.

Returns

the type of symmetry found in the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Remarks

for possible symmetry types see [OSParameters.h](#)

4.140.2.58 `int OSInstance::getNumberOfColumnsForMatrix (int n)`

Get the number of blocks in the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the number of blocks. Get the number of columns in the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the number of columns.

4.140.2.59 `int OSInstance::getNumberOfRowsForMatrix (int n)`

Get the number of rows in the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the number of rows.

4.140.2.60 `int OSInstance::getNumberOfValuesForMatrix (int n)`

Get the number of (nonzero) values in the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the number of values.

4.140.2.61 `std::string OSInstance::getMatrixName (int n)`

Get the name of the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the matrix name.

4.140.2.62 `bool OSInstance::matrixHasBase (int n)`

Several tools to parse the constructor list of a matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

4.140.2.63 `OSMatrix* OSInstance::getMatrix (int n)`

Get the list of constructors of the matrix.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the matrix constructors.

4.140.2.64 `GeneralSparseMatrix* OSInstance::getMatrixCoefficientsInColumnMajor (int n)`

Get the (nonzero) elements of the matrix in column major form.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the (nonzero) matrix elements.

4.140.2.65 `GeneralSparseMatrix* OSInstance::getMatrixCoefficientsInRowMajor (int n)`

Get the (nonzero) elements of the matrix in row major form.

Parameters

n	is the index number associated with the matrix.
-----	---

Returns

the (nonzero) matrix elements.

4.140.2.66 **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.

4.140.2.67 int OSInstance::getNumberOfMatrixVariables ()

Get the number of matrix variables.

Returns

the number of matrix variables

4.140.2.68 int OSInstance::getNumberOfMatrixObjectives ()

Get the number of matrix objectives.

Returns

the number of matrix objectives

4.140.2.69 int OSInstance::getNumberOfMatrixConstraints ()

Get the number of matrix constraints.

Returns

the number of matrix constraints

4.140.2.70 int OSInstance::getNumberOfMatrixExpressions ()

Get the number of matrix-valued expressions.

Returns

the number of matrix-valued variables

4.140.2.71 MatrixExpression OSInstance::getMatrixExpressions ()**

Get the pointers to the roots of all matrix expression trees.

Returns

an array of pointers to [MatrixExpression](#) objects

4.140.2.72 MatrixExpressionTree* OSInstance::getMatrixExpressionTree (int rowIdx)

Get the matrix expression tree for a given row index.

Returns

a matrix expression tree

4.140.2.73 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

4.140.2.74 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

4.140.2.75 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

4.140.2.76 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

4.140.2.77 `std::map<int, MatrixExpressionTree* > OSInstance::getAllMatrixExpressionTrees ()`

Returns

a map: the key is the row index and the value is the corresponding expression tree

4.140.2.78 `std::map<int, MatrixExpressionTree* > OSInstance::getAllMatrixExpressionTreesMod ()`

Returns

a map: the key is the row index and the value is the corresponding expression tree

4.140.2.79 `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.

4.140.2.80 `int OSInstance::getNumberOfMatrixExpressionTreeIndexes ()`

Get the number of unique matrix expression tree indexes.

Returns

the number of unique matrix expression tree indexes.

4.140.2.81 `std::string OSInstance::getTimeDomainFormat ()`

Get the format of the time domain ("stages"/"interval")

Returns

the format of the time domain.

4.140.2.82 `int OSInstance::getTimeDomainStageNumber ()`

Get the number of stages that make up the time domain.

Returns

the number of time stages.

4.140.2.83 `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.

4.140.2.84 `int* OSInstance::getTimeDomainStageNumberOfVariables ()`

Get the number of variables contained in each time stage.

Returns

a vector of size `numberOfStages`.

4.140.2.85 `int* OSInstance::getTimeDomainStageNumberOfConstraints ()`

Get the number of constraints contained in each time stage.

Returns

a vector of size `numberOfStages`.

4.140.2.86 `int* OSInstance::getTimeDomainStageNumberOfObjectives ()`

Get the number of objectives contained in each time stage.

Returns

a vector of size `numberOfStages`.

4.140.2.87 `int** OSInstance::getTimeDomainStageVarList ()`

Get the list of variables in each stage.

Returns

one array of integers for each stage.

4.140.2.88 `int** OSInstance::getTimeDomainStageConList ()`

Get the list of constraints in each stage.

Returns

one array of integers for each stage.

4.140.2.89 `int** OSInstance::getTimeDomainStageObjList ()`

Get the list of objectives in each stage.

Returns

one array of integers for each stage.

4.140.2.90 `double OSInstance::getTimeDomainIntervalStart ()`

Get the start for the time domain interval.

Returns

start end of the time interval.

4.140.2.91 `double OSInstance::getTimeDomainIntervalHorizon ()`

Get the horizon for the time domain interval.

Returns

the end of the time interval.

4.140.2.92 `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.

4.140.2.93 `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.

4.140.2.94 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.

4.140.2.95 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.

4.140.2.96 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.

4.140.2.97 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.

4.140.2.98 `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.

4.140.2.99 `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.

4.140.2.100 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.

4.140.2.101 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</i> ↔ <i>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.

4.140.2.102 bool OSInstance::setObjectives (int *number*, std::string * *names*, std::string * *maxOrMins*, double * *constants*, double * *weights*, SparseVector ** *objectiveCoefficients*)

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</i> ↔ <i>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.

4.140.2.103 `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.

4.140.2.104 `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.

4.140.2.105 `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.

4.140.2.106 `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 indexes.length - 1).
<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 starts.length - 1).

Returns

whether the linear constraint coefficients were set successfully.

4.140.2.107 `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 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 indexes.length - 1).
<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 starts.length - 1).

Returns

whether the linear constraint coefficients were copied successfully.

4.140.2.108 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.

4.140.2.109 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.

4.140.2.110 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.

4.140.2.111 `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.

4.140.2.112 `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.

4.140.2.113 `bool OSInstance::addMatrix (int index, std::string name, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry, ENUM_MATRIX_TYPE matrixType, unsigned int numberOfChildren, 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>numberOfChildren</i>	is the number of MatrixNode child elements, i.e., the number of matrix constructors in the m_children array.
<i>m_children</i>	is the array of matrix constructors used in the definition of this matrix.

Returns

whether the matrix was added successfully.

4.140.2.114 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.

4.140.2.115 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.

```
4.140.2.116 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.

4.140.2.117 `bool OSInstance::addCone (int index, int numberOfRows, int numberOfColumns, ENUM_CONE_TYPE coneType, std::string name, int referenceldx, 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>referenceldx</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.

4.140.2.118 `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.

4.140.2.119 `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.

4.140.2.120 `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.

4.140.2.121 `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>numberOf↵ Columns</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>distortionMatrix↵ Idx</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>numberOf↵ OtherIndexes</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.

4.140.2.122 `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>numberOf↵ Columns</i>	holds the number of columns. It is required.
<i>coneType</i>	holds the cone type. For more information consult the enumeration ENUM_CONE_TYP↵ E further up in this file. This argument is required and must be ENUM_CONE_TYPE_↵ nonnegativePolynomials. ENUM_CONE_TYPE_sumOfSquaresPolynomials. ENUM_CONE↵ _TYPE_moment.

<i>name</i>	holds the cone name; use null or empty <code>std::string("")</code> 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>numberOf↔ OtherIndexes</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.

4.140.2.123 `std::string OSInstance::printModel ()`

Print the infix representation of the problem.

Returns

a string with the infix representation

4.140.2.124 `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

4.140.2.125 `bool OSInstance::initializeNonLinearStructures ()`

Initialize the data structures for the nonlinear API.

Returns

true if we have initialized the nonlinear data strucutres.

4.140.2.126 `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.

4.140.2.127 `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\(\)](#).

4.140.2.128 `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\(\)](#).

4.140.2.129 `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\(\)](#).

4.140.2.130 **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\(\)](#).

4.140.2.131 **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](#).

4.140.2.132 **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.

4.140.2.133 **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.

4.140.2.134 **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.

4.140.2.135 **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.

4.140.2.136 `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.

4.140.2.137 `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.

4.140.2.138 `SparseHessianMatrix* OSInstance::calculateHessian (double * x, int idx, bool new_x)`

Calculate the Hessian of a constraint or objective function.

Parameters

x	is a pointer (double array) to the current variable values
new_x	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.

4.140.2.139 `bool OSInstance::getSparseJacobianFromColumnMajor ()`

Returns

true if successful in generating the constraints gradient.

4.140.2.140 `bool OSInstance::getSparseJacobianFromRowMajor ()`

Returns

true if successful in generating the constraints gradient.

4.140.2.141 `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

4.140.2.142 `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

4.140.2.143 `SparseHessianMatrix* OSInstance::getLagrangianHessianSparsityPattern ()`

Returns

a pointer to a [SparseHessianMatrix](#) with the nonzero structure of the Lagrangian Expression Tree

4.140.2.144 `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 [addQTermsToExpressionTree\(\)](#) – see below

4.140.2.145 `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.

4.140.2.146 `SparseJacobianMatrix* OSInstance::getJacobianSparsityPattern ()`

Returns

pointer to a [SparseJacobianMatrix](#).

4.140.2.147 `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>instanceData->variables->numberOfVariables</code>
------------------	--

Returns

if successfully created

4.140.2.148 `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 vdX m_iNumberOfNonlinearVariables

Returns

a double vector equal to the dimension of the range space the result of the forward p sweep

4.140.2.149 `std::vector<double> OSInstance::reverseAD (int p, std::vector< double > vdlambda)`

Perform an AD reverse sweep.

Parameters

p	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 $n \times p$

4.140.2.150 `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

4.140.2.151 `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

4.140.2.152 `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

4.140.2.153 `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 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

4.140.2.154 `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 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

4.140.2.155 `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

4.140.2.156 `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

4.140.2.157 `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 accociated with this stage and `startIdx` gives the first variable in this stage.)

4.140.2.158 `bool OSInstance::setTimeDomainStageVariablesUnordered (int numberOfStages, int * numberOfVariables, int ** varIndex)`

This sets the variables associated with each time domain stage in srbitrary order.

(I.e., for each stage *numberOfVariables* gives the number of variables accociated with this stage and *varIndex[i]* gives the index of each variable in stage[i].)

4.140.2.159 `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 accociated with this stage and *startIdx* gives the first constraint in this stage.)

4.140.2.160 `bool OSInstance::setTimeDomainStageConstraintsUnordered (int numberOfStages, int * numberOfConstraints, int ** conIndex)`

This sets the constraints associated with each time domain stage in srbitrary order.

(I.e., for each stage *numberOfConstraints* gives the number of constraints accociated with this stage and *conIndex[i]* gives the index of each constraint in stage[i].)

4.140.2.161 `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 accociated with this stage and *startIdx* gives the first objective in this stage.)

4.140.2.162 `bool OSInstance::setTimeDomainStageObjectivesUnordered (int numberOfStages, int * numberOfObjectives, int ** varIndex)`

This sets the objectives associated with each time domain stage in arbitrary order.

(I.e., for each stage *numberOfObjectives* gives the number of objectives accociated with this stage and *objIndex[i]* gives the index of each objective in stage[i].)

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

- [OSInstance.h](#)

4.141 OSMatlab Class Reference

The [OSMatlab](#) Class.

```
#include <OSMatlabSolver.h>
```

Collaboration diagram for OSMatlab:

Public Member Functions

- [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

4.141.1 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.

4.141.2 Member Function Documentation

4.141.2.1 `std::string OSMatlab::solve ()`

Solve the problem instance.

Returns

a string with the solution in OSrL format

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

- OSMatlabSolver.h

4.142 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 * createConstructorTreeFromPrefix](#) (std::vector< [MatrixNode](#) * > mtxConstructorVec)
- virtual [ENUM_MATRIX_CONSTRUCTOR_TYPE](#) [getNodeType](#) ()
- virtual std::string [getNodeName](#) ()
- virtual [ENUM_MATRIX_TYPE](#) [getMatrixType](#) ()
- virtual bool [expandElements](#) (bool rowMajor)
A method to process a matrixType into a specific block structure.
- 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 [numberOfChildren](#), [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.

Additional Inherited Members

4.142.1 Detailed Description

a data structure to represent a matrix object (derived from [MatrixType](#))

Definition at line 2185 of file OSMatrix.h.

4.142.2 Member Function Documentation

4.142.2.1 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.

4.142.2.2 virtual `ENUM_MATRIX_CONSTRUCTOR_TYPE OSMatrix::getNodeType ()` [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.142.2.3 virtual `std::string OSMatrix::getNodeName ()` [virtual]

Returns

the name of the operator

Implements [MatrixNode](#).

4.142.2.4 virtual `ENUM_MATRIX_TYPE OSMatrix::getMatrixType ()` [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.142.2.5 virtual `bool OSMatrix::expandElements (bool rowMajor)` [virtual]

A method to process a matrixType into a specific block structure.

Parameters

<i>rowOffset</i>	defines a partition of the matrix rows into the blocks
<i>colOffset</i>	defines a partition of the matrix columns into the blocks
<i>rowMajor</i>	controls whether the blocks are stored by row or by column
<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. 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.

Reimplemented from [MatrixType](#).

4.142.2.6 `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](#).

4.142.2.7 `bool OSMatrix::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.

Parameters

<i>name</i>	holds the matrix name; use null or empty <code>std::string ("")</code> 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 <code>ENUM_MATRIX_SYMMETRY</code> in OSGeneral.h . If no symmetry, use <code>ENUM_MATRIX_SYMMETRY_none</code> .

<i>matrixType</i>	tracks the type of elements contained in this matrix. For more information see the enumeration <code>ENUM_MATRIX_TYPE</code> in OSGeneral.h . If unsure, use <code>ENUM_MATRIX_TYPE_unknown</code> .
<i>numberOfChildren</i>	is the number of MatrixNode child elements, i.e., the number of matrix constructors in the <code>m_mChildren</code> 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.

4.142.2.8 `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](#).

Reimplemented in [OSMatrixWithMatrixConIdx](#), [OSMatrixWithMatrixObjIdx](#), and [OSMatrixWithMatrixVarIdx](#).

4.142.2.9 `OSMatrix * OSMatrix::cloneMatrixNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [MatrixNode](#).

Reimplemented in [OSMatrixWithMatrixConIdx](#), [OSMatrixWithMatrixObjIdx](#), and [OSMatrixWithMatrixVarIdx](#).

4.142.2.10 `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 "numberOfXXX" 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

4.142.2.11 `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

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

- [OSMatrix.h](#)

4.143 OSMatrixWithMatrixConIdx Class Reference

this class extends [OSMatrix](#) for use, e.g., in the matrixCon section of OSoL and OSrL

Inheritance diagram for OSMatrixWithMatrixConIdx:

Collaboration diagram for OSMatrixWithMatrixConIdx:

Public Member Functions

- virtual std::string [getMatrixNodeInXML](#) ()
- virtual [OSMatrixWithMatrixConIdx](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [IsEqual](#) ([OSMatrixWithMatrixConIdx](#) *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](#) ([OSMatrixWithMatrixConIdx](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.143.1 Detailed Description

this class extends [OSMatrix](#) for use, e.g., in the matrixCon section of OSoL and OSrL

Definition at line 2445 of file [OSMatrix.h](#).

4.143.2 Member Function Documentation

4.143.2.1 virtual std::string OSMatrixWithMatrixConIdx::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.

Reimplemented from [OSMatrix](#).

4.143.2.2 `OSMatrix * OSMatrixWithMatrixConIdx::cloneMatrixNode ()` [virtual]

The implementation of the virtual functions.

Returns

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

Reimplemented from [OSMatrix](#).

4.143.2.3 `bool OSMatrixWithMatrixConIdx::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 "numberOf↵XXX" 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

4.143.2.4 `bool OSMatrixWithMatrixConIdx::deepCopyFrom (OSMatrixWithMatrixConIdx * 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:

- [OSMatrix.h](#)

4.144 OSMatrixWithMatrixObjIdx Class Reference

this class extends [OSMatrix](#) for use, e.g., in the matrixObj section of OSoL and OSrL

Inheritance diagram for OSMatrixWithMatrixObjIdx:

Collaboration diagram for OSMatrixWithMatrixObjIdx:

Public Member Functions

- virtual std::string [getMatrixNodeInXML](#) ()
- virtual [OSMatrixWithMatrixObjIdx * cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [IsEqual](#) ([OSMatrixWithMatrixObjIdx *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](#) ([OSMatrixWithMatrixObjIdx](#) *that)

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

Additional Inherited Members

4.144.1 Detailed Description

this class extends [OSMatrix](#) for use, e.g., in the `matrixObj` section of `OSoL` and `OSrL`

Definition at line 2391 of file `OSMatrix.h`.

4.144.2 Member Function Documentation

4.144.2.1 virtual std::string OSMatrixWithMatrixObjIdx::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.

Reimplemented from [OSMatrix](#).

4.144.2.2 OSMatrix * OSMatrixWithMatrixObjIdx::cloneMatrixNode () [virtual]

The implementation of the virtual functions.

Returns

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

Reimplemented from [OSMatrix](#).

4.144.2.3 bool OSMatrixWithMatrixObjIdx::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 "numberOfXXX" 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

4.144.2.4 bool OSMatrixWithMatrixObjIdx::deepCopyFrom (OSMatrixWithMatrixObjIdx * 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:

- [OSMatrix.h](#)

4.145 OSMatrixWithMatrixVarIdx Class Reference

this class extends [OSMatrix](#) for use, e.g., in the matrixVar section of OSoL and OSrL

Inheritance diagram for OSMatrixWithMatrixVarIdx:

Collaboration diagram for OSMatrixWithMatrixVarIdx:

Public Member Functions

- virtual std::string [getMatrixNodeInXML](#) ()
- virtual [OSMatrixWithMatrixVarIdx](#) * [cloneMatrixNode](#) ()
The implementation of the virtual functions.
- bool [isEqual](#) ([OSMatrixWithMatrixVarIdx](#) *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](#) ([OSMatrixWithMatrixVarIdx](#) *that)
A function to make a deep copy of an instance of this class.

Additional Inherited Members

4.145.1 Detailed Description

this class extends [OSMatrix](#) for use, e.g., in the matrixVar section of OSoL and OSrL

Definition at line 2336 of file OSMatrix.h.

4.145.2 Member Function Documentation

4.145.2.1 virtual std::string OSMatrixWithMatrixVarIdx::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.

Reimplemented from [OSMatrix](#).

4.145.2.2 **OSMatrix * OSMatrixWithMatrixVarIdx::cloneMatrixNode ()** [virtual]

The implementation of the virtual functions.

Returns

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

Reimplemented from [OSMatrix](#).

4.145.2.3 **bool OSMatrixWithMatrixVarIdx::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 "numberOf↵XXX" 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

4.145.2.4 **bool OSMatrixWithMatrixVarIdx::deepCopyFrom (OSMatrixWithMatrixVarIdx * *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:

- [OSMatrix.h](#)

4.146 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 or 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).

4.146.1 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`.

4.146.2 Member Function Documentation

4.146.2.1 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.

4.146.3 Member Data Documentation

4.146.3.1 `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`.

4.146.3.2 `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:

- [OSmps2OS.h](#)

4.147 OSmps2osil Class Reference

The [OSmps2osil](#) Class.

```
#include <OSmps2osil.h>
```

Collaboration diagram for `OSmps2osil`:

Public Member Functions

- [OSmps2osil](#) (`std::string mpsfilename`)
the [OSmps2osil](#) class constructor
- [~OSmps2osil](#) ()
the [OSmps2osil](#) class destructor
- `bool` [createOSInstance](#) ()
create an [OSInstance](#) from the MPS instance representation

Public Attributes

- [OSInstance](#) * `osinstance`
`osinstance` is a pointer to the [OSInstance](#) object that gets created from the instance represented in NL format

4.147.1 Detailed Description

The [OSmps2osil](#) Class.

Author

Robert Fourer, Jun Ma, Kipp Martin

Version

1.0, 03/14/2004

Since

OS 1.0

Remarks

the [OSmps2osil](#) class is used for reading an instance in MPS format and creating an [OSInstance](#) object in OSiL format

Definition at line 39 of file OSmps2osil.h.

4.147.2 Member Function Documentation

4.147.2.1 bool OSmps2osil::createOSInstance ()

create an [OSInstance](#) from the MPS instance representation

Returns

whether the instance is created successfully.

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

- [OSmps2osil.h](#)

4.148 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).

4.148.1 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.

4.148.2 Constructor & Destructor Documentation

4.148.2.1 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*

4.148.3 Member Function Documentation

4.148.3.1 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

4.148.3.2 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

4.148.3.3 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" (<i>asl</i> should point to the same location as either <i>rw</i> or <i>cw</i>)

Returns

whether the operation was successful

4.148.3.4 `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.

4.148.3.5 `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')

4.148.3.6 `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

4.148.3.7 `OSnLNode* OSnl2OS::walkTree (expr * e)`

parse an nl tree structure holding a nonlinear expression

Returns

the AMPL nonlinear structure as an [OSnLNode](#).

4.148.4 Member Data Documentation

4.148.4.1 `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.

4.148.4.2 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 ooption is not NULL, the jobID has been duplicated to ooption.

Definition at line 189 of file OSnl2OS.h.

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

- OSnl2OS.h

4.149 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](#) * > nlnodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a matrix-valued [OSExpressionTree](#) 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](#) * > [preOrderOSnLMNodeTraversal](#) (std::vector< [ExprNode](#) * > *prefixVector)
Called by [getPrefixFromExpressionTree\(\)](#).
- [OSnLMNode * createExpressionTreeFromPostfix](#) (std::vector< [ExprNode](#) * > nlnodeVec)
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a matrix-valued [OSExpressionTree](#) 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](#) * > [postOrderOSnLMNodeTraversal](#) (std::vector< [ExprNode](#) * > *postfixVector)
Called by [getPostfixFromExpressionTree\(\)](#).
- virtual [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

4.149.1 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 1760 of file OSnLNode.h.

4.149.2 Member Function Documentation

4.149.2.1 [OSnLMNode*](#) [OSnLMNode::createExpressionTreeFromPrefix](#) ([std::vector](#)< [ExprNode](#) * > *nlnodeVec*)

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a matrix-valued [OS↔ExpressionTree](#) root node.

Parameters

<i>nlnodeVec</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](#).

4.149.2.2 [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](#).

4.149.2.3 [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](#).

4.149.2.4 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](#).

4.149.2.5 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](#).

4.149.2.6 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](#).

4.149.2.7 virtual OSnLMNode* OSnLMNode::copyNodeAndDescendants () [virtual]

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented in [OSnLMNodeMatrixCon](#), [OSnLMNodeMatrixObj](#), [OSnLMNodeMatrixVar](#), [OSnLMNodeMatrix↔Reference](#), [OSnLMNodeMatrixUpperTriangle](#), and [OSnLMNodeMatrixLowerTriangle](#).

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

- [OSnLNode.h](#)

4.150 OSnLMNodeDiagonalMatrixFromVector Class Reference

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**4.150.1 Detailed Description**

Definition at line 2382 of file OSnLNode.h.

4.150.2 Member Function Documentation

4.150.2.1 virtual std::string OSnLMNodeDiagonalMatrixFromVector::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.150.2.2 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:

- [OSnLNode.h](#)

4.151 OSnLMNodeIdentityMatrix Class Reference

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

4.151.1 Detailed Description

Definition at line 2197 of file OSnLNode.h.

4.151.2 Member Function Documentation

4.151.2.1 virtual std::string OSnLMNodeIdentityMatrix::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.151.2.2 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:

- [OSnLNode.h](#)

4.152 OSnLMNodeMatrixCon Class Reference

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 [OSnLMNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- virtual bool [isEqual](#) ([OSnLMNodeMatrixCon](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
The index of the matrixCon.

4.152.1 Detailed Description

Definition at line 2673 of file OSnLNode.h.

4.152.2 Member Function Documentation

4.152.2.1 virtual std::string OSnLMNodeMatrixCon::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.152.2.2 virtual std::string OSnLMNodeMatrixCon::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.152.2.3 `virtual std::string OSnLMNodeMatrixCon::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the [OSnLMNode](#) <matrixCon>.

Reimplemented from [ExprNode](#).

4.152.2.4 `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](#).

4.152.2.5 `virtual OSnLMNode* OSnLMNodeMatrixCon::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLNode.h](#)

4.153 OSnLMNodeMatrixDiagonal Class Reference

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

4.153.1 Detailed Description

Definition at line 2344 of file OSnLNode.h.

4.153.2 Member Function Documentation

4.153.2.1 `virtual std::string OSnLMNodeMatrixDiagonal::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.153.2.2 `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:

- [OSnLNode.h](#)

4.154 OSnLMNodeMatrixDotTimes Class Reference

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

4.154.1 Detailed Description

Definition at line 2159 of file OSnLNode.h.

4.154.2 Member Function Documentation

4.154.2.1 `virtual std::string OSnLMNodeMatrixDotTimes::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.154.2.2 `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:

- [OSnLNode.h](#)

4.155 OSnLMNodeMatrixInverse Class Reference

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**4.155.1 Detailed Description**

Definition at line 2045 of file OSnLNode.h.

4.155.2 Member Function Documentation**4.155.2.1** `virtual std::string OSnLMNodeMatrixInverse::getTokenName () [virtual]`**Returns**

the value of operator name

Implements [ExprNode](#).

4.155.2.2 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:

- [OSnLNode.h](#)

4.156 OSnLMNodeMatrixLowerTriangle Class Reference

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 [OSnLMNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- 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.

4.156.1 Detailed Description

Definition at line 2235 of file OSnLNode.h.

4.156.2 Member Function Documentation

4.156.2.1 virtual std::string OSnLMNodeMatrixLowerTriangle::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.156.2.2 `virtual std::string OSnLMNodeMatrixLowerTriangle::getNonlinearExpressionInXML () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).
the OSiL XML for the [OSnLMNode](#) <matrix>.

Reimplemented from [ExprNode](#).

4.156.2.3 `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](#).

4.156.2.4 `virtual OSnLMNode* OSnLMNodeMatrixLowerTriangle::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLNode.h](#)

4.157 OSnLMNodeMatrixMinus Class Reference

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

4.157.1 Detailed Description

Definition at line 1930 of file OSnLNode.h.

4.157.2 Member Function Documentation

4.157.2.1 `virtual std::string OSnLMNodeMatrixMinus::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.157.2.2 `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:

- [OSnLNode.h](#)

4.158 OSnLMNodeMatrixNegate Class Reference

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

4.158.1 Detailed Description

Definition at line 1968 of file OSnLNode.h.

4.158.2 Member Function Documentation

4.158.2.1 `virtual std::string OSnLMNodeMatrixNegate::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.158.2.2 `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:

- [OSnLNode.h](#)

4.159 OSnLMNodeMatrixObj Class Reference

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 OSnLMNode * copyNodeAndDescendants \(\)`
make a copy of this node and all its descendants
- `virtual bool isEqual \(OSnLMNodeMatrixObj *that\)`
A function to check for the equality of two objects.

Public Attributes

- `int idx`
The index of the matrixObj.

4.159.1 Detailed Description

Definition at line 2602 of file OSnLNode.h.

4.159.2 Member Function Documentation

4.159.2.1 `virtual std::string OSnLMNodeMatrixObj::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.159.2.2 `virtual std::string OSnLMNodeMatrixObj::getTokenNumber () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.159.2.3 `virtual std::string OSnLMNodeMatrixObj::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the [OSnLMNode](#) <matrixObj>.

Reimplemented from [ExprNode](#).

4.159.2.4 `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](#).

4.159.2.5 `virtual OSnLMNode* OSnLMNodeMatrixObj::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLNode.h](#)

4.160 OSnLMNodeMatrixPlus Class Reference

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

4.160.1 Detailed Description

Definition at line 1846 of file OSnLNode.h.

4.160.2 Member Function Documentation

4.160.2.1 virtual std::string OSnLMNodeMatrixPlus::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.160.2.2 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:

- [OSnLNode.h](#)

4.161 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

4.161.1 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 2755 of file OSnLNode.h.

4.161.2 Member Function Documentation

4.161.2.1 virtual std::string OSnLMNodeMatrixProduct::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.161.2.2 OSnLNode * 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:

- [OSnLNode.h](#)

4.162 OSnLMNodeMatrixReference Class Reference

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 [OSnLMNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- virtual bool [isEqual](#) ([OSnLMNodeMatrixReference](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
The index of the matrix.

4.162.1 Detailed Description

Definition at line 2460 of file OSnLNode.h.

4.162.2 Member Function Documentation

4.162.2.1 `virtual std::string OSnLMNodeMatrixReference::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.162.2.2 `virtual std::string OSnLMNodeMatrixReference::getTokenNumber () [virtual]`

Returns

a string token that corresponds to the [OSnLMNode](#).

Reimplemented from [ExprNode](#).

4.162.2.3 `virtual std::string OSnLMNodeMatrixReference::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the [OSnLMNode](#) <matrixReference>.

Reimplemented from [ExprNode](#).

4.162.2.4 `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](#).

4.162.2.5 `virtual OSnLMNode* OSnLMNodeMatrixReference::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLMNode.h](#)

4.163 OSnLMNodeMatrixScalarTimes Class Reference

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

4.163.1 Detailed Description

Definition at line 2121 of file OSnLNode.h.

4.163.2 Member Function Documentation

4.163.2.1 virtual std::string [OSnLMNodeMatrixScalarTimes::getTokenName](#) () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.163.2.2 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:

- [OSnLNode.h](#)

4.164 OSnLMNodeMatrixSubmatrixAt Class Reference

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

4.164.1 Detailed Description

Definition at line 2421 of file OSnLNode.h.

4.164.2 Member Function Documentation

4.164.2.1 virtual std::string OSnLMNodeMatrixSubmatrixAt::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.164.2.2 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:

- [OSnLNode.h](#)

4.165 OSnLMNodeMatrixSum Class Reference

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

4.165.1 Detailed Description

Definition at line 1884 of file OSnLNode.h.

4.165.2 Member Function Documentation

4.165.2.1 `virtual std::string OSnLMNodeMatrixSum::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.165.2.2 `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:

- [OSnLNode.h](#)

4.166 OSnLMNodeMatrixTimes Class Reference

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

4.166.1 Detailed Description

Definition at line 2007 of file OSnLNode.h.

4.166.2 Member Function Documentation

4.166.2.1 `virtual std::string OSnLMNodeMatrixTimes::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.166.2.2 `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:

- [OSnLNode.h](#)

4.167 OSnLMNodeMatrixTranspose Class Reference

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

4.167.1 Detailed Description

Definition at line 2083 of file OSnLNode.h.

4.167.2 Member Function Documentation

4.167.2.1 `virtual std::string OSnLMNodeMatrixTranspose::getTokenName ()` [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.167.2.2 `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:

- [OSnLNode.h](#)

4.168 OSnLMNodeMatrixUpperTriangle Class Reference

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 [OSnLMNode *](#) [copyNodeAndDescendants \(\)](#)
make a copy of this node and all its descendants
- 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.

4.168.1 Detailed Description

Definition at line 2289 of file OSnLNode.h.

4.168.2 Member Function Documentation

4.168.2.1 `virtual std::string OSnLMNodeMatrixUpperTriangle::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.168.2.2 `virtual std::string OSnLMNodeMatrixUpperTriangle::getNonlinearExpressionInXML () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).
the OSiL XML for the [OSnLMNode](#) <matrix>.

Reimplemented from [ExprNode](#).

4.168.2.3 `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](#).

4.168.2.4 `virtual OSnLMNode* OSnLMNodeMatrixUpperTriangle::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLNode.h](#)

4.169 OSnLMNodeMatrixVar Class Reference

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 [OSnLMNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- virtual bool [isEqual](#) ([OSnLMNodeMatrixVar](#) *that)
A function to check for the equality of two objects.

Public Attributes

- int [idx](#)
The index of the matrixVar.

4.169.1 Detailed Description

Definition at line 2531 of file OSnLNode.h.

4.169.2 Member Function Documentation

4.169.2.1 virtual std::string OSnLMNodeMatrixVar::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.169.2.2 virtual std::string OSnLMNodeMatrixVar::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.169.2.3 virtual std::string OSnLMNodeMatrixVar::getNonlinearExpressionInXML () [virtual]

Returns

the OSiL XML for the [OSnLMNode](#) <matrixReference>.

Reimplemented from [ExprNode](#).

4.169.2.4 `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](#).

4.169.2.5 `virtual OSnLMNode* OSnLMNodeMatrixVar::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLMNode](#).

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

- [OSnLNode.h](#)

4.170 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 * > nlNodeVec)`
Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a scalar-valued *OSExpressionTree* 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()`.
- virtual `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.

4.170.1 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*.

4.170.2 Member Function Documentation

4.170.2.1 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](#).

4.170.2.2 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](#).

4.170.2.3 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](#).

4.170.2.4 OSnLNode* OSnLNode::createExpressionTreeFromPrefix (std::vector< ExprNode * > nINodeVec)

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in prefix format and create a scalar-valued [OSnLExpressionTree](#) root node.

Parameters

<i>nlnNodeVec</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](#).

4.170.2.5 `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](#).

4.170.2.6 `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](#).

4.170.2.7 `OSnLNode* OSnLNode::createExpressionTreeFromPostfix (std::vector< ExprNode * > nlnNodeVec)`

Take a vector of ExprNodes (OSnLNodes and OSnLMNodes) in postfix format and create a scalar-valued [OS←ExpressionTree](#) root node.

Parameters

<i>nlnNodeVec</i>	holds a vector of pointers to OSnLNodes in postfix format
-------------------	---

Returns

a pointer to an [OSnLNode](#) which is the root of an [OSExpressionTree](#).

4.170.2.8 `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](#).

4.170.2.9 `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](#).

4.170.2.10 `virtual OSnLNode* OSnLNode::copyNodeAndDescendants () [virtual]`

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented in [OSnLNodeVariable](#), and [OSnLNodeNumber](#).

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

- [OSnLNode.h](#)

4.171 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

4.171.1 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.

4.171.2 Member Function Documentation

4.171.2.1 virtual std::string OSnLNodeAbs::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.171.2.2 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](#).

4.171.2.3 `OSnLNode * OSnLNodeAbs::cloneExprNode ()` [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.171.2.4 `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:

- [OSnLNode.h](#)

4.172 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

4.172.1 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 1560 of file OSnLNode.h.

4.172.2 Member Function Documentation

4.172.2.1 virtual std::string OSnLNodeAlldiff::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.172.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.172.2.3 [OSnLNode](#) * [OSnLNodeAllDiff::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.172.2.4 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:

- [OSnLNode.h](#)

4.173 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

4.173.1 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.

4.173.2 Member Function Documentation

4.173.2.1 virtual std::string OSnLNodeCos::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.173.2.2 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](#).

4.173.2.3 OSnLNode * OSnLNodeCos::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.173.2.4 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:

- [OSnLNode.h](#)

4.174 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

4.174.1 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.

4.174.2 Member Function Documentation

4.174.2.1 virtual std::string OSnLNodeDivide::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.174.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.174.2.3 OSnLNode * OSnLNodeDivide::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.174.2.4 double OSnLNodeDivide::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:

- [OSnLNode.h](#)

4.175 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

4.175.1 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 1346 of file OSnLNode.h.

4.175.2 Member Function Documentation

4.175.2.1 virtual std::string [OSnLNodeE::getTokenNumber](#) () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.175.2.2 virtual std::string [OSnLNodeE::getTokenName](#) () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

4.175.2.3 `virtual std::string OSnLNodeE::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

4.175.2.4 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.175.2.5 `OSnLNode * OSnLNodeE::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.175.2.6 `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:

- [OSnLNode.h](#)

4.176 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

4.176.1 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.

4.176.2 Member Function Documentation

4.176.2.1 virtual std::string OSnLNodeErf::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.176.2.2 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

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.176.2.3 OSnLNode * OSnLNodeErf::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.176.2.4 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:

- [OSnLNode.h](#)

4.177 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

4.177.1 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.

4.177.2 Member Function Documentation

4.177.2.1 virtual std::string OSnLNodeExp::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.177.2.2 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](#).

4.177.2.3 `double OSnLNodeExp::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD)` [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

4.177.2.4 `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:

- [OSnLNode.h](#)

4.178 OSnLNodeIf Class Reference

The [OSnLNodeIf](#) 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

4.178.1 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.

4.178.2 Member Function Documentation

4.178.2.1 virtual std::string OSnLNodeIf::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.178.2.2 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](#).

4.178.2.3 [OSnLNode](#) * [OSnLNodef::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.178.2.4 double [OSnLNodef::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:

- [OSnLNode.h](#)

4.179 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

4.179.1 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 <ln>

Definition at line 815 of file OSnLNode.h.

4.179.2 Member Function Documentation

4.179.2.1 virtual std::string OSnLNodeLn::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.179.2.2 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](#).

4.179.2.3 OSnLNode * OSnLNodeLn::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.179.2.4 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:

- [OSnLNode.h](#)

4.180 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

4.180.1 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 1612 of file OSnLNode.h.

4.180.2 Member Function Documentation

4.180.2.1 virtual std::string [OSnLNodeMatrixDeterminant::getTokenName](#) () [[virtual](#)]

Returns

the value of operator name

Implements [ExprNode](#).

4.180.2.2 double [OSnLNodeMatrixDeterminant::calculateFunction](#) (double * x) [[virtual](#)]

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

4.180.2.3 `double OSnLNodeMatrixDeterminant::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD)`
`[virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

4.180.2.4 `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:

- [OSnLNode.h](#)

4.181 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

4.181.1 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 1712 of file OSnLNode.h.

4.181.2 Member Function Documentation

4.181.2.1 `virtual std::string OSnLNodeMatrixToScalar::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.181.2.2 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.181.2.3 `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](#).

4.181.2.4 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:

- [OSnLNode.h](#)

4.182 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

4.182.1 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 1662 of file OSnLNode.h.

4.182.2 Member Function Documentation

4.182.2.1 `virtual std::string OSnLNodeMatrixTrace::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.182.2.2 `double OSnLNodeMatrixTrace::calculateFunction (double * x) [virtual]`

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

4.182.2.3 `double OSnLNodeMatrixTrace::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

4.182.2.4 `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:

- [OSnLNode.h](#)

4.183 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

4.183.1 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.

4.183.2 Member Function Documentation

4.183.2.1 `virtual std::string OSnLNodeMax::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.183.2.2 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.183.2.3 `OSnLNode * OSnLNodeMax::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.183.2.4 `double OSnLNodeMax::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:

- [OSnLNode.h](#)

4.184 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

4.184.1 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.

4.184.2 Member Function Documentation

4.184.2.1 `virtual std::string OSnLNodeMin::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.184.2.2 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.184.2.3 `OSnLNode * OSnLNodeMin::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.184.2.4 `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:

- [OSnLNode.h](#)

4.185 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

4.185.1 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.

4.185.2 Member Function Documentation

4.185.2.1 `virtual std::string OSnLNodeMinus::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.185.2.2 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.185.2.3 `OSnLNode * OSnLNodeMinus::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.185.2.4 `double OSnLNodeMinus::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:

- [OSnLNode.h](#)

4.186 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

4.186.1 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.

4.186.2 Member Function Documentation

4.186.2.1 `virtual std::string OSnLNodeNegate::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.186.2.2 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.186.2.3 `OSnLNode * OSnLNodeNegate::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.186.2.4 `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:

- [OSnLNode.h](#)

4.187 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 [OSnLNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- 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.

4.187.1 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.

4.187.2 Member Function Documentation

4.187.2.1 `virtual std::string OSnLNodeNumber::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.187.2.2 `virtual std::string OSnLNodeNumber::getTokenNumber () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.187.2.3 `virtual std::string OSnLNodeNumber::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

4.187.2.4 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.187.2.5 `OSnLNode * OSnLNodeNumber::cloneExprNode ()` [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.187.2.6 `virtual OSnLNode* OSnLNodeNumber::copyNodeAndDescendants ()` [virtual]

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLNode](#).

4.187.2.7 `double OSnLNodeNumber::constructADTape (std::map< int, int > * ADIdx, ADvector * XAD)` [virtual]

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

4.187.3 Member Data Documentation

4.187.3.1 `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:

- [OSnLNode.h](#)

4.188 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

4.188.1 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 1412 of file OSnLNode.h.

4.188.2 Member Function Documentation

4.188.2.1 virtual std::string OSnLNodePI::getTokenNumber () [virtual]

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.188.2.2 `virtual std::string OSnLNodePI::getTokenName () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

4.188.2.3 `virtual std::string OSnLNodePI::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the number node.

Reimplemented from [ExprNode](#).

4.188.2.4 `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

<code>x</code>	holds the values of the variables in a double array.
----------------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.188.2.5 `OSnLNode * OSnLNodePI::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.188.2.6 `double OSnLNodePI::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:

- [OSnLNode.h](#)

4.189 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

4.189.1 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.

4.189.2 Member Function Documentation

4.189.2.1 `virtual std::string OSnLNodePlus::getTokenName () [virtual]`

Returns

the value of operator name

Implements [ExprNode](#).

4.189.2.2 `double OSnLNodePlus::calculateFunction (double * x) [virtual]`

The implementation of the virtual functions.

Returns

a double.

Implements [OSnLNode](#).

4.189.2.3 `double OSnLNodePlus::constructADTape (std::map< int, int > * AD/dx, ADvector * XAD) [virtual]`

The implementation of the virtual functions.

Returns

a ADdouble.

Implements [OSnLNode](#).

4.189.2.4 `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:

- [OSnLNode.h](#)

4.190 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

4.190.1 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.

4.190.2 Member Function Documentation

4.190.2.1 virtual std::string OSnLNodePower::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.190.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.190.2.3 OSnLNode * OSnLNodePower::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.190.2.4 double OSnLNodePower::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:

- [OSnLNode.h](#)

4.191 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

4.191.1 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.

4.191.2 Member Function Documentation

4.191.2.1 virtual std::string OSnLNodeProduct::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.191.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.191.2.3 [OSnLNode](#) * [OSnLNodeProduct::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.191.2.4 double [OSnLNodeProduct::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:

- [OSnLNode.h](#)

4.192 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

4.192.1 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.

4.192.2 Member Function Documentation

4.192.2.1 virtual std::string OSnLNodeSin::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.192.2.2 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](#).

4.192.2.3 [OSnLNode](#) * [OSnLNodeSin::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.192.2.4 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:

- [OSnLNode.h](#)

4.193 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

4.193.1 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.

4.193.2 Member Function Documentation

4.193.2.1 virtual std::string OSnLNodeSqrt::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.193.2.2 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](#).

4.193.2.3 OSnLNode * OSnLNodeSqrt::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.193.2.4 double OSnLNodeSqrt::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:

- [OSnLNode.h](#)

4.194 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

4.194.1 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.

4.194.2 Member Function Documentation

4.194.2.1 virtual std::string OSnLNodeSquare::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.194.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.194.2.3 [OSnLNode](#) * [OSnLNodeSquare::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.194.2.4 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:

- [OSnLNode.h](#)

4.195 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

4.195.1 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.

4.195.2 Member Function Documentation

4.195.2.1 virtual std::string OSnLNodeSum::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.195.2.2 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

x	holds the values of the variables in a double array.
-----	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.195.2.3 OSnLNode * OSnLNodeSum::cloneExprNode () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.195.2.4 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:

- [OSnLNode.h](#)

4.196 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

4.196.1 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.

4.196.2 Member Function Documentation

4.196.2.1 virtual std::string OSnLNodeTimes::getTokenName () [virtual]

Returns

the value of operator name

Implements [ExprNode](#).

4.196.2.2 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](#).

4.196.2.3 [OSnLNode](#) * [OSnLNodeTimes::cloneExprNode](#) () [virtual]

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.196.2.4 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:

- [OSnLNode.h](#)

4.197 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 [OSnLNode](#) * [copyNodeAndDescendants](#) ()
make a copy of this node and all its descendants
- 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

4.197.1 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 1478 of file OSnLNode.h.

4.197.2 Member Function Documentation

4.197.2.1 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

<i>a</i>	pointer to a map of the variables in the OSnLNode and its children
----------	--

Reimplemented from [OSnLNode](#).

4.197.2.2 `virtual std::string OSnLNodeVariable::getTokenNumber () [virtual]`

Returns

a string token that corresponds to the [OSnLNode](#).

Reimplemented from [ExprNode](#).

4.197.2.3 `virtual std::string OSnLNodeVariable::getTokenName () [virtual]`

Returns

a std::string token that corresponds to the [OSnLNode](#).

Implements [ExprNode](#).

4.197.2.4 `virtual std::string OSnLNodeVariable::getNonlinearExpressionInXML () [virtual]`

Returns

the OSiL XML for the variable node.

Reimplemented from [ExprNode](#).

4.197.2.5 `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

<i>x</i>	holds the values of the variables in a double array.
----------	--

Returns

the function value given the current variable values.

Implements [OSnLNode](#).

4.197.2.6 `OSnLNode * OSnLNodeVariable::cloneExprNode () [virtual]`

The implementation of the virtual functions.

Returns

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

Implements [ExprNode](#).

4.197.2.7 `virtual OSnLNode* OSnLNodeVariable::copyNodeAndDescendants ()` [virtual]

make a copy of this node and all its descendants

Returns

a pointer to the duplicate node

Reimplemented from [OSnLNode](#).

4.197.2.8 `double OSnLNodeVariable::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:

- [OSnLNode.h](#)

4.198 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
- int [templInt](#)
some temporary items to facilitate code sharing
- [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](#).
- `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

4.198.1 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](#).

4.198.2 Member Data Documentation

4.198.2.1 `ExprNode* OSnLParserData::nlNodePoint`

These entities are used for parsing <nonlinearExpressions>

a pointer to an [OSnLNode](#) object

Definition at line 83 of file [OSnLParserData.h](#).

4.198.2.2 `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](#).

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

- [OSnLParserData.h](#)

4.199 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 [serviceURIPresent](#)
children of the <general> element
- bool [minDiskSpacePresent](#)
children of the <system> element
- bool [serviceTypePresent](#)
children of the <service> element
- bool [maxTimePresent](#)
children of the <job> element
- int [numberOfVariables](#)
children of the <optimization> element
- bool [otherOptionNumberPresent](#)
attributes of <other> options
- bool [solverOptionNamePresent](#)
attributes of <solverOptions> element
- bool [categoryAttributePresent](#)
generic attributes
- std::string * [jobDependencies](#)
all arrays are collected here
- int [templnt](#)
some temporary items to facilitate code sharing
- 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

4.199.1 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](#).

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

- [OSoLParserData.h](#)

4.200 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.

4.200.1 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.

4.200.2 Member Function Documentation

4.200.2.1 [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:

- [OSoLReader.h](#)

4.201 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

4.201.1 Detailed Description

Take an [OSOption](#) object and write a string that validates against the OSoL schema.

Definition at line 29 of file OSoLWriter.h.

4.201.2 Member Function Documentation

4.201.2.1 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.

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

- [OSoLWriter.h](#)

4.202 OSOption Class Reference

The Option Class.

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

Collaboration diagram for OSOption:

Public Member Functions

- [OSOption](#) ()
Default constructor.
- [~OSOption](#) ()
Class destructor.
- bool [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.
- bool [isEqual](#) ([OSOption](#) *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 (OOption *that)`

A function to make a deep copy of an OOption 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 `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 `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 `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 `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 `setAnotherInitBasisStatus` (int type, int idx, int status)
- Set the basis status for another variable, objective or constraint/slack.*

 - 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 `setOtherOptionOrResultEnumeration` (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.*

 - 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 `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 [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 [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.

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.

4.202.1 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.

4.202.2 Member Function Documentation

4.202.2.1 `bool OOption::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

4.202.2.2 `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 "numberOfXXX" attributes and <XXX> children)

4.202.2.3 `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

4.202.2.4 `std::string OSOption::getJobID ()`

Get the job ID.

4.202.2.5 `int OSOption::getNumberOfInitVarValues ()`

Get the number of initial variable values.

Returns

the number of initial variable values.

4.202.2.6 `int OSOption::getNumberOfInitVarValuesString ()`

Get the number of initial variable strings.

Returns

the number of initial variable strings.

4.202.2.7 int OOption::getNumberOfIntegerVariableBranchingWeights ()

Get the number of variables for which integer branching weights are provided.

Returns

the number of variables.

4.202.2.8 int OOption::getNumberOfSOS ()

Get the number of special ordered sets for which branching weights are provided.

Returns

the number of variables.

4.202.2.9 int OOption::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.

4.202.2.10 int OOption::getNumberOfOtherVariableOptions ()

Get the number of other variable options.

Returns

the number of other variable options.

4.202.2.11 int OOption::getNumberOfInitObjValues ()

Get the number of initial objective values.

Returns

the number of initial objective values.

4.202.2.12 int OOption::getNumberOfInitObjBounds ()

Get the number of initial objective bounds.

Returns

the number of initial objective bound values.

4.202.2.13 `int OSOption::getNumberOfOtherObjectiveOptions ()`

Get the number of other objective options.

Returns

the number of other objective options.

4.202.2.14 `int OSOption::getNumberOfInitConValues ()`

Get the number of initial constraint values.

Returns

the number of initial constraint values.

4.202.2.15 `int OSOption::getNumberOfInitDualVarValues ()`

Get the number of initial dual variable values.

Returns

the number of initial dual variable values.

4.202.2.16 `int OSOption::getNumberOfOtherConstraintOptions ()`

Get the number of other constraint options.

Returns

the number of other constraint options.

4.202.2.17 `int OSOption::getNumberOfSolverOptions ()`

Get the number of solver options.

Returns

the number of solver options.

4.202.2.18 `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

4.202.2.19 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

4.202.2.20 OtherOption OSOption::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

4.202.2.21 OtherOption OSOption::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

4.202.2.22 OtherOption OSOption::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*

4.202.2.23 OtherOption OSOption::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

4.202.2.24 std::string* OSOption::getJobDependencies ()

Get the array of job dependencies.

Returns

a vector of pointers to [JobDependencies](#) objects

4.202.2.25 `std::string* OSOption::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

4.202.2.26 `std::string* OSOption::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

4.202.2.27 `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

4.202.2.28 `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

4.202.2.29 `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

4.202.2.30 `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

4.202.2.31 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

4.202.2.32 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

4.202.2.33 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

4.202.2.34 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

4.202.2.35 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

4.202.2.36 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

4.202.2.37 `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

4.202.2.38 `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

4.202.2.39 `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

4.202.2.40 `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

4.202.2.41 `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

4.202.2.42 `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

4.202.2.43 `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

4.202.2.44 `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
--------------------------	-------------------------------

4.202.2.45 `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)

4.202.2.46 `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

4.202.2.47 **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

4.202.2.48 **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

4.202.2.49 **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

4.202.2.50 **SOSWeights**** **OSOption::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

4.202.2.51 **std::vector<OtherVariableOption*>** **OSOption::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.

4.202.2.52 [OtherVariableOption*](#) OSOption::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

4.202.2.53 [OtherVariableOption**](#) OSOption::getAllOtherVariableOptions ()

Get all <other> variable options.

Returns

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

4.202.2.54 [InitObjValue**](#) OSOption::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

4.202.2.55 [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

4.202.2.56 [double*](#) OSOption::getInitObjValuesDense (int *numberOfObjectives*)

Get the initial values associated with the objectives in dense form.

Parameters

<i>numberOf↵ Objectives</i>	holds the dimension of the vector
---------------------------------	-----------------------------------

Returns

a vector of double that hold initial values (or [OSNaN\(\)](#)) for all of the objectives

4.202.2.57 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

4.202.2.58 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

4.202.2.59 double* OSOption::getInitObjLowerBoundsDense (int numberOfObjectives)

Get the initial lower bounds associated with the objectives in dense form.

Parameters

<i>numberOf↵ Objectives</i>	holds the dimension of the vector
---------------------------------	-----------------------------------

Returns

a vector of double that hold initial lower bounds (or [OSNaN\(\)](#)) for all of the objectives

4.202.2.60 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

4.202.2.61 double* OSOption::getInitObjUpperBoundsDense (int numberOfObjectives)

Get the initial upper bounds associated with the objectives in dense form.

Parameters

<i>numberOf↵ Objectives</i>	holds the dimension of the vector
---------------------------------	-----------------------------------

Returns

a vector of double that hold initial upper bounds (or [OSNaN\(\)](#)) for all of the objectives

4.202.2.62 `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>numberOf↵ Objectives</i>	is the dimension of the array
---------------------------------	-------------------------------

4.202.2.63 `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

4.202.2.64 `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

4.202.2.65 OtherObjectiveOption** OSOption::getAllOtherObjectiveOptions ()

Get all <other> objective options.

Returns

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

4.202.2.66 InitConValue** OSOption::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

4.202.2.67 double* OSOption::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

4.202.2.68 double* OSOption::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

4.202.2.69 InitDualVarValue** OSOption::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

4.202.2.70 double* OSOption::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

4.202.2.71 `double*` OSOption::getInitDualVarLowerBoundsDense (int *numberOfConstraints*)

Get the initial dual variables associated with the lower bounds in dense form.

Parameters

<i>numberOf↔ Constraints</i>	holds the dimension of the vector
----------------------------------	-----------------------------------

Returns

a vector of double that hold initial lower bounds for all of the dual variables

4.202.2.72 `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

4.202.2.73 `double* OSOption::getInitDualVarUpperBoundsDense (int numberOfConstraints)`

Get the initial dual variables associated with the upper bounds in dense form.

Parameters

<i>numberOf↔ Constraints</i>	holds the dimension of the vector
----------------------------------	-----------------------------------

Returns

a vector of double that hold initial upper bounds for all of the dual variables

4.202.2.74 `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>numberOf↔ Constraints</i>	is the dimension of the array
----------------------------------	-------------------------------

4.202.2.75 `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

4.202.2.76 [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

4.202.2.77 [OtherConstraintOption**](#) OSOption::getAllOtherConstraintOptions ()

Get all <other> constraint options.

Returns

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

4.202.2.78 [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.

4.202.2.79 [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.

4.202.2.80 SolverOption OSOption::getAllSolverOptions ()**

Get all solver options.

Returns

a pointer to an array [SolverOption](#) objects

4.202.2.81 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

4.202.2.82 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)

4.202.2.83 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

4.202.2.84 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

4.202.2.85 `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

4.202.2.86 `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)

4.202.2.87 `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

4.202.2.88 `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)

4.202.2.89 `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</i> 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>varType</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

4.202.2.90 `bool OSOption::setOtherOptionOrResultEnumeration (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>numberOfEI</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

4.202.2.91 `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>otherOptionNumber</i>	number of the <other> option in the list of <other> options (zero-based)
<i>varNumber</i>	number of the <i>in the array</i> (zero-based)
<i>idx</i>	index of the variable to which this value belongs
<i>value</i>	value of the option (as a string)
<i>lbValue</i>	value associated with the lower bound of the variable (as a string)
<i>ubValue</i>	value associated with the upper bound of the variable (as a string)

4.202.2.92 `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

4.202.2.93 `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>otherOptionNumber</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)

4.202.2.94 `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 <i>array</i>
<i>enumType</i>	type of the data in the <i><enumeration> array</i>
<i>description</i>	further description of this <i><other> element</i>

4.202.2.95 `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 <i><other></i> option in the list of <i><other></i> options (zero-based)
<i>conNumber</i>	number of the <i><obj></i> 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)

4.202.2.96 `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 <i>children contained in this <other> element</i>
<i>name</i>	<i>name of this solver option</i>
<i>value</i>	<i>a value associated with this option</i>
<i>solver</i>	<i>the solver to which this option applies</i>
<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>

4.202.3 Member Data Documentation

4.202.3.1 `GeneralFileHeader*` `OSOption::optionHeader`

`OSOption` has a header and five other children: general, system, service, job, and optimization.

header information

Definition at line 3576 of file `OSOption.h`.

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

- [OSOption.h](#)

4.203 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
- `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 osrlFile`
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
- `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 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 ospInputFile`
name of an input file with xml in OS Process language format, used for example to knock on a server, for example -ospInput ../data/ospFiles/demo.ospl
- `std::string ospInput`
ospInput is the content of the ospInputFile
- `std::string ospOutputFile`
name of an output file where the solver should write the result of a knock or kill service request
- `std::string ospOutput`

osplOutput is the content of the osplOutputFile

- `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 .dat that holds the GAMS control file
- `std::string solverName`
the name of the solver to be invoked, e.g.
- `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 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

4.203.1 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.

4.203.2 Member Data Documentation

4.203.2.1 `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.

4.203.2.2 `std::string osOptionsStruc::solverName`

the name of the solver to be invoked, e.g.

-solver **lpopt**

Definition at line 129 of file OSOptionsStruc.h.

4.203.2.3 `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.

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

- [OSOptionsStruc.h](#)

4.204 OSosrl2ampl Class Reference

The [OSosrl2ampl](#) Class.

```
#include <OSosrl2ampl.h>
```

Public Member Functions

- [OSosl2ampl](#) ()
the [OSosl2ampl](#) class constructor
- [~OSosl2ampl](#) ()
the [OSosl2ampl](#) class destructor
- bool [writeSolFile](#) (std::string osrl, ASL *asl, std::string filename)
Convert the solution to AMPL .sol format.

4.204.1 Detailed Description

The [OSosl2ampl](#) Class.

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

The [OSosl2ampl](#) class is used for writing an AMPL sol file, including any results indexed over variables, constraints, objectives.

Definition at line 44 of file [OSosl2ampl.h](#).

4.204.2 Member Function Documentation

4.204.2.1 bool OSosl2ampl::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:

- [OSosl2ampl.h](#)

4.205 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.*

4.205.1 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.

4.205.2 Member Function Documentation

4.205.2.1 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

4.205.2.2 `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

4.205.2.3 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

4.205.2.4 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

4.205.2.5 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

4.205.2.6 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:

- [OSOutput.h](#)

4.206 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** ()

4.206.1 Detailed Description

a class that holds information about one output channel (file, device, stream, peripheral, etc.)

Definition at line 41 of file OSOutput.h.

4.206.2 Constructor & Destructor Documentation

4.206.2.1 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
-------------	---

4.206.3 Member Function Documentation

4.206.3.1 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

4.206.3.2 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

4.206.3.3 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

4.206.3.4 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

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

- [OSOutput.h](#)

4.207 OSReferencedObject Class Reference

ReferencedObject class.

```
#include <OSReferenced.hpp>
```

Inheritance diagram for OSReferencedObject:

4.207.1 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 [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 [OSSmartPtr](#)s to A and B, but A and B would have Weak [OSSmartPtr](#)s to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak [OSSmartPtr](#)s 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.

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

- OSReferenced.hpp

4.208 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:

4.208.1 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:

- OSReferenced.hpp

4.209 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.
- `int getTimeNumber ()`
Get the number of time measurements.
- `double getTimeValue ()`
Get the time measurement.
- `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.
- bool [getSolutionWeightedObjectives](#) (int solldx)
Get the [i]th optimization solution form of the objective.
- std::string [getSolutionMessage](#) (int solldx)
Get the [i]th optimization solution message, where i equals the given solution index.
- std::vector< [IndexValuePair](#) * > [getOptimalPrimalVariableValues](#) (int solldx)
Get one solution of optimal primal variable values.
- int [getBasisStatusNumberOfEI](#) (int solldx, int object, int status)
Get the number of indices that belong to a particular basis status.
- int [getBasisStatusEI](#) (int solldx, int object, int status, int j)
Get an entry in the array of indices that belong to a particular basis status.
- int [getBasisInformationDense](#) (int solldx, int object, int *resultArray, int dim)
Get the basis information associated with the variables, objectives or constraints for some solution.
- int [getNumberOfOtherVariableResults](#) (int solldx)
Get numberOfOtherVariableResult.
- int [getAnOtherVariableResultNumberOfVar](#) (int solldx, int iOther)
Get getAnOtherVariableResultNumberOfVar.
- std::string [getOtherVariableResultArrayType](#) (int solldx, 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 solldx, int otherIdx, int enumIdx)
Get the value of an enum associated with an <other> result for some solution.
- std::string [getOtherVariableResultEnumerationDescription](#) (int solldx, int otherIdx, int enumIdx)
Get the description of an enum associated with an <other> result for some solution.
- int [getOtherVariableResultEnumerationNumberOfEI](#) (int solldx, int otherIdx, int enumIdx)
Get the size of an enum associated with an <other> result for some solution.
- int [getOtherVariableResultEnumerationEI](#) (int solldx, int otherIdx, int enumIdx, int j)
Get one index of an enum associated with an <other> result for some solution.
- int [getOtherVariableResultArrayDense](#) (int solldx, int otherIdx, std::string *resultArray, int dim)
Get the values of a array or an <enumeration> associated with an <other> result for some solution.
- double [getOptimalObjValue](#) (int objIdx, int solldx)
Get one solution's optimal objective value.
- std::string [getOtherObjectiveResultArrayType](#) (int solldx, 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 solldx, int otherIdx, int enumIdx)
Get the value of an enum associated with an <other> result for some solution.
- std::string [getOtherObjectiveResultEnumerationDescription](#) (int solldx, int otherIdx, int enumIdx)
Get the description of an enum associated with an <other> result for some solution.

- int [getOtherObjectiveResultEnumerationNumberOfEI](#) (int solldx, int otherIdx, int enumIdx)
Get the size of an enum associated with an <other> result for some solution.
- int [getOtherObjectiveResultEnumerationEI](#) (int solldx, int otherIdx, int enumIdx, int j)
Get one index of an enum associated with an <other> result for some solution.
- int [getOtherObjectiveResultArrayDense](#) (int solldx, 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.
- std::vector< [IndexValuePair](#) * > [getOptimalDualVariableValues](#) (int solldx)
Get one solution of optimal dual variable values.
- std::string [getOtherConstraintResultArrayType](#) (int solldx, 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 solldx, int otherIdx, int enumIdx)
Get the value of an enum associated with an <other> result for some solution.
- std::string [getOtherConstraintResultEnumerationDescription](#) (int solldx, int otherIdx, int enumIdx)
Get the description of an enum associated with an <other> result for some solution.
- int [getOtherConstraintResultEnumerationNumberOfEI](#) (int solldx, int otherIdx, int enumIdx)
Get the size of an enum associated with an <other> result for some solution.
- int [getOtherConstraintResultEnumerationEI](#) (int solldx, int otherIdx, int enumIdx, int j)
Get one index of an enum associated with an <other> result for some solution.
- int [getOtherConstraintResultArrayDense](#) (int solldx, 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.
- 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 [setNumberOfSolutionSubstatus](#) (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 [setOtherVariableResultSolver](#) (int solIdx, int otherIdx, std::string solver)
Set the solver of another (non-standard/solver specific) variable-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherVariableResultCategory](#) (int solIdx, int otherIdx, std::string category)
Set the category 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 [setOtherOptionOrResultEnumeration](#) (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 solldx, 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 solldx, 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 solldx, 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 [setOtherObjectiveResultSolver](#) (int solldx, int otherIdx, std::string solver)
Set the solver of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultCategory](#) (int solldx, int otherIdx, std::string category)
Set the category of another (non-standard/solver specific) objective-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherObjectiveResultObjIdx](#) (int solldx, 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 solldx, 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 solldx, 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 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.
- bool [setNumberOfDualValues](#) (int solldx, int numberOfCon)
Set the number of constraints to be given a value.
- bool [setNumberOfDualVariableValues](#) (int solldx, int n)
Set the [i]th optimization solution's number of dual variable values, where i equals the given solution index.
- bool [setDualVariableValuesSparse](#) (int solldx, std::vector< [IndexValuePair](#) * > x)
Set the [i]th optimization solution's dual variable values, where i equals the given solution index.
- bool [setDualVariableValuesDense](#) (int solldx, double *y)
Set the [i]th optimization solution's dual variable values, where i equals the given solution index.
- bool [setConstraintValuesDense](#) (int solldx, double *constraintValues)
Set the [i]th optimization solution's constraint values, where i equals the given solution index.
- bool [setDualValue](#) (int solldx, int number, int idx, std::string name, double val)
Set a dual value.
- bool [setOtherConstraintResultNumberOfCon](#) (int solldx, 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 solldx, 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 solldx, 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 solldx, 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 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.
- bool [setOtherConstraintResultEnumType](#) (int solldx, int otherldx, 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 solldx, int otherldx, 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 solldx, int otherldx, 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 [setOtherConstraintResultSolver](#) (int solldx, int otherldx, std::string solver)
Set the solver of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherConstraintResultCategory](#) (int solldx, int otherldx, std::string category)
Set the category of another (non-standard/solver specific) constraint-related result, for the [i]th solution, where i equals the given solution index.
- bool [setOtherConstraintResultConIdx](#) (int solldx, int otherldx, int conldx, 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 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.
- bool [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.
- bool [setMatrixVariableSolution](#) (int solldx, int numberOfMatrixVar_, int numberOfOtherMatrixVariableResults_)
Set the [i]th optimization solution's [MatrixVariableSolution](#), where i equals the given solution index.
- bool [setMatrixVarValuesAttributes](#) (int solldx, int idx, int matrixVarldx, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry=ENUM_MATRIX_SYMMETRY_none, [ENUM_MATRIX_TYPE](#) type=ENUM_MATRIX_TYPE_unknown, std::string name="")
A method to set general attributes for a matrixVar in the [i]th optimization solution, where i equals the given solution index.
- bool [setMatrixVarValuesBlockStructure](#) (int solldx, int idx, int *colOffset, int colOffsetSize, int *rowOffset, int rowOffsetSize, int numberOfBlocks, int blocksConstructorldx=0)
A method to set the block structure for the values of a matrixVar in the [i]th optimization solution, where i equals the given solution index.
- bool [setMatrixVarValuesBlockElements](#) (int solldx, int idx, int blkno, int blkRowldx, int blkColldx, int nz, int *start, int *index, [MatrixElementValues](#) *value, [ENUM_MATRIX_TYPE](#) valueType, ENUM_MATRIX_SYMMETRY symmetry=ENUM_MATRIX_SYMMETRY_none, bool rowMajor=false)
A method to set the elements within a block of a matrixVar in the [i]th optimization solution, where i equals the given solution index.
- bool [setMatrixVariablesOtherResultGeneralAttributes](#) (int solldx, int idx, std::string name, std::string description, std::string value, std::string type, std::string solver, std::string category, int numberOfMatrixVar=0, std::string matrixType="", int numberOfEnumerations=0, std::string enumType="")
A method to set general attributes for another (non-standard/solver specific) result associated with the matrix variables in the [i]th optimization solution, where i equals the given solution index.

- bool [setMatrixVariablesOtherResultMatrixAttributes](#) (int solIdx, int otherIdx, int matrixVarIdx, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry=ENUM_MATRIX_SYMMETRY_none, ENUM_MATRIX_TYPE type=ENUM_MATRIX_TYPE_unknown, std::string name="")
A method to set attributes for a matrixVar in the [j]th other result associated with matrix variables in the [i]th optimization solution, where i equals the given solution index and j equals a given other result index.
- bool [setMatrixVariablesOtherResultBlockStructure](#) (int solIdx, int otherIdx, int matrixVarIdx, int *colOffset, int colOffsetSize, int *rowOffset, int rowOffsetSize, int numberOfBlocks, int blocksConstructorIdx=0)
A method to set the block structure for the values of a matrixVar associated with the [j]th "other" result of the [i]th optimization solution, where i equals the given solution index, and j equals the index of the other result.
- bool [setMatrixVariablesOtherResultBlockElements](#) (int solIdx, int otherIdx, int matrixVarIdx, int blkno, int blkRowIdx, int blkColIdx, int nz, int *start, int *index, [MatrixElementValues](#) *value, ENUM_MATRIX_TYPE valueType, ENUM_MATRIX_SYMMETRY symmetry=ENUM_MATRIX_SYMMETRY_none, bool rowMajor=false)
A method to set the elements within a block of a matrixVar associated with the [j]th "other" result in the [i]th optimization solution, where i equals the given solution index and j equals the index of the other result.
- 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.

4.209.1 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 2548 of file OSResult.h.

4.209.2 Member Function Documentation

4.209.2.1 `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

4.209.2.2 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.209.2.3 `GeneralStatus* OSResult::getGeneralStatus ()`

Get the general status.

Returns

the general status.

4.209.2.4 `std::string OSResult::getGeneralStatusType ()`

Get the general status type, which can be: success, error, warning.

Returns

the general status type, null if none.

4.209.2.5 `std::string OSResult::getGeneralStatusDescription ()`

Get the general status description.

Returns

the general status description, null or empty std::string if none.

4.209.2.6 `int OSResult::getNumberOfGeneralSubstatues ()`

Get the number of substatures.

Returns

the number of substatures, -1 if general status does not exist.

4.209.2.7 `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.

4.209.2.8 `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.

4.209.2.9 `std::string OSResult::getGeneralMessage ()`

Get the general message.

Returns

the general message.

4.209.2.10 `std::string OSResult::getServiceName ()`

Get service name.

Returns

the service name.

4.209.2.11 `std::string OSResult::getServiceURI ()`

Get service uri.

Returns

the service uri.

4.209.2.12 `std::string OSResult::getInstanceName ()`

Get instance name.

Returns

the instance name.

4.209.2.13 `std::string OSResult::getJobID ()`

Get the job id.

Returns

the job id.

4.209.2.14 `std::string OSResult::getSolverInvoked ()`

Get the solver invoked.

Returns

the solver invoked.

4.209.2.15 `std::string OSResult::getTimeStamp ()`

Get the time stamp.

Returns

the time stamp.

4.209.2.16 `int OSResult::getNumberOfOtherGeneralResults ()`

Get the number of other results in the <general> element.

Returns

the number of other <general> results.

4.209.2.17 `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.

4.209.2.18 `int OSResult::getTimeNumber ()`

Get the number of time measurements.

Returns

the number of time measurements

4.209.2.19 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

4.209.2.20 int OSResult::getVariableNumber ()

Get variable number.

Returns

variable number, -1 if no information.

4.209.2.21 int OSResult::getObjectiveNumber ()

Get objective number.

Returns

objective number, -1 if no information.

4.209.2.22 int OSResult::getConstraintNumber ()

Get constraint number.

Returns

constraint number, -1 if no information.

4.209.2.23 int OSResult::getSolutionNumber ()

get the number of solutions.

Returns

the number of solutions, 0 if none.

4.209.2.24 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>solldx</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

4.209.2.25 `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.

4.209.2.26 `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.

4.209.2.27 `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.

4.209.2.28 `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.

4.209.2.29 `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.

4.209.2.30 `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 ENUM_BASIS_STATUS — see OSGeneral.h)
<i>status</i>	gives the basis status type

4.209.2.31 `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

4.209.2.32 `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

4.209.2.33 int OSResult::getNumberOfOtherVariableResults (int *solldx*)

Get numberOfOtherVariableResult.

Returns

numberOfOtherVariableResult, -1 if no information.

4.209.2.34 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.

4.209.2.35 std::string OSResult::getOtherVariableResultArrayType (int *solldx*, int *otherIdx*)

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>otherIdx</i>	is the index of the current <other> result

Returns

the array type

4.209.2.36 std::string OSResult::getOtherVariableResultEnumerationValue (int *solldx*, int *otherIdx*, int *enumIdx*)

Get the value of an enum associated with an <other> result for some solution.

Parameters

<i>solIdx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the value

4.209.2.37 `std::string OSResult::getOtherVariableResultEnumerationDescription (int solIdx, int otherIdx, int enumIdx)`

Get the description of an enum associated with an <other> result for some solution.

Parameters

<i>solIdx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the description

4.209.2.38 `int OSResult::getOtherVariableResultEnumerationNumberOfEI (int solIdx, int otherIdx, int enumIdx)`

Get the size of an enum associated with an <other> result for some solution.

Parameters

<i>solIdx</i>	is the solution index
<i>otherIdx</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

4.209.2.39 `int OSResult::getOtherVariableResultEnumerationEI (int solIdx, int otherIdx, int enumIdx, int j)`

Get one index of an enum associated with an <other> result for some solution.

Parameters

<i>solIdx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result
<i>enumIdx</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

4.209.2.40 `int OSResult::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.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</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

4.209.2.41 double OSResult::getOptimalObjValue (int *objIdx*, int *solldx*)

Get one solution's optimal objective value.

Parameters

<i>objIdx</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.

4.209.2.42 std::string OSResult::getOtherObjectiveResultArrayType (int *solldx*, int *otherIdx*)

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>otherIdx</i>	is the index of the current <other> result

Returns

the array type

4.209.2.43 std::string OSResult::getOtherObjectiveResultEnumerationValue (int *solldx*, int *otherIdx*, int *enumIdx*)

Get the value of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the value

4.209.2.44 `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>otherIdx</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the description

4.209.2.45 `int OSResult::getOtherObjectiveResultEnumerationNumberOfEl (int solldx, int otherIdx, int enumIdx)`

Get the size of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</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

4.209.2.46 `int OSResult::getOtherObjectiveResultEnumerationEl (int solldx, int otherIdx, int enumIdx, 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>otherIdx</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

4.209.2.47 `int OSResult::getOtherObjectiveResultArrayDense (int solldx, 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.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</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

4.209.2.48 `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.

4.209.2.49 `std::string OSResult::getOtherConstraintResultArrayType (int solldx, int otherIdx)`

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>otherIdx</i>	is the index of the current <other> result

Returns

the array type

4.209.2.50 `std::string OSResult::getOtherConstraintResultEnumerationValue (int solldx, int otherIdx, int enumIdx)`

Get the value of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result
<i>enumIdx</i>	is the index of the current enumeration level

Returns

the value

4.209.2.51 `std::string OSResult::getOtherConstraintResultEnumerationDescription (int solldx, int otherIdx, int enumIdx)`

Get the description of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	is the index of the current <other> result is the index of the current enumeration level

Returns

the description

4.209.2.52 `int OSResult::getOtherConstraintResultEnumerationNumberOfEI (int solldx, int otherIdx, int enumIdx)`

Get the size of an enum associated with an <other> result for some solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</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

4.209.2.53 `int OSResult::getOtherConstraintResultEnumerationEl (int solldx, int otherIdx, int enumIdx, 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>otherIdx</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

4.209.2.54 `int OSResult::getOtherConstraintResultArrayDense (int solldx, 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.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</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

4.209.2.55 `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.

4.209.2.56 `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.

4.209.2.57 `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.

4.209.2.58 `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.

4.209.2.59 `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.

4.209.2.60 `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.

4.209.2.61 `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.

4.209.2.62 `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.

4.209.2.63 `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.

4.209.2.64 `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.

4.209.2.65 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.

4.209.2.66 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.

4.209.2.67 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.

4.209.2.68 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.

4.209.2.69 `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.

4.209.2.70 `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.

4.209.2.71 `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.

4.209.2.72 `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.

4.209.2.73 `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.

4.209.2.74 `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.

4.209.2.75 `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.

4.209.2.76 `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.

4.209.2.77 `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.

4.209.2.78 `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.

4.209.2.79 `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.

4.209.2.80 `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.

4.209.2.81 `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.

4.209.2.82 `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.

4.209.2.83 `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.

4.209.2.84 `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.

4.209.2.85 `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.

4.209.2.86 `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.

4.209.2.87 `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.

4.209.2.88 `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.

4.209.2.89 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.

4.209.2.90 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.

4.209.2.91 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.

4.209.2.92 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.

4.209.2.93 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.

4.209.2.94 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.

4.209.2.95 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.

4.209.2.96 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.

4.209.2.97 `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.

4.209.2.98 `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.

4.209.2.99 `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.

4.209.2.100 `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.

4.209.2.101 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.

4.209.2.102 bool OSResult::setTime (double *time*)

Set time.

Parameters

<i>time</i>	holds the time.
-------------	-----------------

Returns

whether the time is set successfully.

4.209.2.103 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.

4.209.2.104 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.

4.209.2.105 `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.

4.209.2.106 `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.

4.209.2.107 `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.

4.209.2.108 `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.

4.209.2.109 `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.

4.209.2.110 `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.

4.209.2.111 `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.

4.209.2.112 `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.

4.209.2.113 `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.

4.209.2.114 `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.

4.209.2.115 `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.

4.209.2.116 `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.

4.209.2.117 `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.

4.209.2.118 `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.

4.209.2.119 `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.

4.209.2.120 `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.

4.209.2.121 `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.

4.209.2.122 `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.

4.209.2.123 `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.

4.209.2.124 `bool OSResult::setConstraintNumber (int constraintNumber)`

Set the constraint number.

Parameters

<i>constraintNumber</i>	holds the number of constraints
-------------------------	---------------------------------

Returns

whether the constraint number is set successfully or not.

4.209.2.125 `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\(int\)](#)

4.209.2.126 `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 substatures. 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\(int\)](#)

4.209.2.127 `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.

4.209.2.128 `bool OSResult::setNumberOfSolutionSubstatuses (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.

4.209.2.129 `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.

4.209.2.130 `bool OSResult::setSolutionSubstatusType (int solldx, int substatusldx, std::string type)`

Set the solution substatus type.

Parameters

<i>solldx</i>	holds the solution index whose status to set.
<i>substatusldx</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.

4.209.2.131 `bool OSResult::setSolutionSubstatusDescription (int solldx, int substatusldx, 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.

4.209.2.132 `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 [setSolution↵
Number\(int\)](#) method has to be called first.

Parameters

<i>solldx</i>	holds the solution index to set the objective index.
<i>objectiveldx</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\(int\)](#)

4.209.2.133 `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 [setSolution↵
Number\(int\)](#) 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\(int\)](#)

4.209.2.134 `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.

4.209.2.135 `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 [setSolution↔
Number\(int\)](#) method has to be called first.

Parameters

<i>solldx</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\(int\)](#)

4.209.2.136 `bool OSResult::setNumberOfPrimalVariableValues (int solldx, 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>solldx</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\(int\)](#)

4.209.2.137 `bool OSResult::setPrimalVariableValuesSparse (int solldx, 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\(int\)](#)

4.209.2.138 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\(int\)](#)

4.209.2.139 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\(int\)](#)

4.209.2.140 `bool OSResult::setVarValue (int solldx, 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>solldx</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\(int\)](#)

4.209.2.141 `bool OSResult::setNumberOfVarValuesString (int solldx, 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>solldx</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\(int\)](#)

4.209.2.142 `bool OSResult::setVarValueString (int solldx, 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>solldx</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\(int\)](#)

4.209.2.143 `bool OSResult::setBasisStatus (int solldx, 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>i</i>

Returns

whether basis status was set successfully or not.

See also

[setSolutionNumber\(int\)](#)

4.209.2.144 `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>numberOfOtherVariableResult</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 OtherVarResult 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\(int\)](#)

4.209.2.145 `bool OSResult::setAnOtherVariableResultSparse (int solldx, int otherldx, 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\(int\)](#)

4.209.2.146 `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 <other> element's value attribute</i>
<i>varType</i>	<i>holds the type of the <other> element's array</i>
<i>enumType</i>	<i>holds the type of the <other> element's <enumeration> 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\(int\)](#)

4.209.2.147 `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\(int\)](#)

4.209.2.148 `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>	<i>holds the solution index</i>
<i>otherldx</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>s</i>	<i>holds a pointer to the array of values of the var element</i>
<i>type</i>	<i>holds the type of the <other> element's value attribute</i>
<i>varType</i>	<i>holds the type of the <other> element's array</i>
<i>enumType</i>	<i>holds the type of the <other> element's <enumeration> 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\(int\)](#)

4.209.2.149 `bool OSResult::setOtherVariableResultNumberOfVar (int solldx, int otherldx, 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>solldx</i>	holds the solution index
<i>otherldx</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\(int\)](#)

4.209.2.150 `bool OSResult::setOtherVariableResultNumberOfEnumerations (int solldx, int otherldx, 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>solldx</i>	holds the solution index
<i>otherldx</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\(int\)](#)

4.209.2.151 `bool OSResult::setOtherVariableResultName (int solldx, int otherldx, 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>solIdx</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\(int\)](#)

4.209.2.152 bool OSResult::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.

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>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\(int\)](#)

4.209.2.153 bool OSResult::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.

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>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\(int\)](#)

4.209.2.154 `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\(int\)](#)

4.209.2.155 `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\(int\)](#)

4.209.2.156 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\(int\)](#)

4.209.2.157 bool OSResult::setOtherVariableResultSolver (int *solldx*, int *otherldx*, std::string *solver*)

Set the solver 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>solver</i>	holds the type of the other element

Returns

whether the other variable result's solver was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

4.209.2.158 `bool OSResult::setOtherVariableResultCategory (int solldx, int otherldx, std::string category)`

Set the category 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>category</i>	holds the type of the other element

Returns

whether the other variable result's category was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherVariableResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherVarResult
[setSolutionNumber\(int\)](#)

4.209.2.159 `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\(int\)](#)

4.209.2.160 `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\(int\)](#)

4.209.2.161 `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\(int\)](#)

4.209.2.162 `bool OSResult::setOtherOptionOrResultEnumeration (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>otherIdx</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 OtherOptionOrResultEnumeration 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\(int\)](#)

4.209.2.163 bool OSResult::setNumberOfOtherObjectiveResults (int *solldx*, 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>solldx</i>	is the solution index
<i>numberOfOtherObjectiveResult</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 OtherObjResult 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\(int\)](#)

4.209.2.164 bool OSResult::setNumberOfObjValues (int *solldx*, 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>solldx</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\(int\)](#)

4.209.2.165 `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\(int\)](#)

4.209.2.166 `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\(int\)](#)

4.209.2.167 `bool OSResult::setObjectiveValuesDense (int solIdx, 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><code>solIdx</code></i>	holds the solution index to set the objective values.
<i><code>objectiveValues</code></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\(int\)](#)

4.209.2.168 `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><code>solIdx</code></i>	holds the solution index to set the objective values.
<i><code>number</code></i>	holds the location within the sparse array <code>obj</code> that is to be used
<i><code>idx</code></i>	holds the index of the objective that is to be set
<i><code>name</code></i>	holds the objective name (or an empty string).
<i><code>val</code></i>	holds the objective value to set.

Returns

whether primal variable value was set successfully or not.

See also

[setSolutionNumber\(int\)](#)

4.209.2.169 `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>solldx</i>	holds the solution index
<i>otherldx</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\(int\)](#)

4.209.2.170 bool OSResult::setOtherObjectiveResultNumberOfEnumerations (int *solldx*, int *otherldx*, 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>solldx</i>	holds the solution index
<i>otherldx</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\(int\)](#)

4.209.2.171 bool OSResult::setOtherObjectiveResultName (int *solldx*, int *otherldx*, 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>solldx</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\(int\)](#)

4.209.2.172 bool OSResult::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.

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 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\(int\)](#)

4.209.2.173 bool OSResult::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.

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 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\(int\)](#)

4.209.2.174 `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\(int\)](#)

4.209.2.175 `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\(int\)](#)

4.209.2.176 `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\(int\)](#)

4.209.2.177 `bool OSResult::setOtherObjectiveResultSolver (int solldx, int otherldx, std::string solver)`

Set the solver 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>solver</i>	holds the solver of the other element

Returns

whether the other [Objective](#) result's solver was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

4.209.2.178 `bool OSResult::setOtherObjectiveResultCategory (int solldx, int otherldx, std::string category)`

Set the category 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>otherIdx</i>	holds the index of the OtherObjectiveResult object
<i>category</i>	holds the category of the other element

Returns

whether the other objective result's category was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherObjectiveResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherObjResult
[setSolutionNumber\(int\)](#)

4.209.2.179 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\(int\)](#)

4.209.2.180 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\(int\)](#)

4.209.2.181 `bool OSResult::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.

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>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\(int\)](#)

4.209.2.182 `bool OSResult::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.

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>solIdx</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\(int\)](#)

4.209.2.183 bool OSResult::setNumberOfDualValues (int *solldx*, 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>solldx</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\(int\)](#)

4.209.2.184 bool OSResult::setNumberOfDualVariableValues (int *solldx*, 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>solldx</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\(int\)](#)

4.209.2.185 bool OSResult::setDualVariableValuesSparse (int *solldx*, 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\(int\)](#)

4.209.2.186 `bool OSResult::setDualVariableValuesDense (int solIdx, 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>solIdx</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\(int\)](#)

4.209.2.187 `bool OSResult::setConstraintValuesDense (int solIdx, 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>solIdx</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\(int\)](#)

4.209.2.188 `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\(int\)](#)

4.209.2.189 `bool OSResult::setOtherConstraintResultNumberOfCon (int solldx, 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>solldx</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\(int\)](#)

4.209.2.190 `bool OSResult::setOtherConstraintResultNumberOfEnumerations (int solldx, 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>solldx</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\(int\)](#)

4.209.2.191 bool OSResult::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.

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>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\(int\)](#)

4.209.2.192 bool OSResult::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.

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>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\(int\)](#)

4.209.2.193 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>otherldx</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\(int\)](#)

4.209.2.194 bool OSResult::setOtherConstraintResultEnumType (int *solldx*, int *otherldx*, 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>otherldx</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\(int\)](#)

4.209.2.195 `bool OSResult::setOtherConstraintResultValue (int solldx, int otherldx, 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>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\(int\)](#)

4.209.2.196 bool OSResult::setOtherConstraintResultDescription (int *solldx*, int *otherldx*, 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>solldx</i>	holds the solution index
<i>otherldx</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\(int\)](#)

4.209.2.197 bool OSResult::setOtherConstraintResultSolver (int *solldx*, int *otherldx*, std::string *solver*)

Set the solver 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>solver</i>	holds the solver of the other element

Returns

whether the other constraint result's solver was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

4.209.2.198 `bool OSResult::setOtherConstraintResultCategory (int solIdx, int otherIdx, std::string category)`

Set the category 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>category</i>	holds the category of the other element

Returns

whether the other constraint result's category was set successfully or not.

See also

org.optimizationservices.oscommon.datastructure.osresult.OtherConstraintResult
 org.optimizationservices.oscommon.datastructure.osresult.OtherConResult
[setSolutionNumber\(int\)](#)

4.209.2.199 `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\(int\)](#)

4.209.2.200 `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\(int\)](#)

4.209.2.201 `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\(int\)](#)

4.209.2.202 `bool OSResult::setMatrixVariableSolution (int solIdx, int numberOfMatrixVar_, int numberOfOtherMatrixVariableResults_)`

Set the [i]th optimization solution's [MatrixVariableSolution](#), 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 [MatrixVariableSolution](#) objects. Further methods then store the values into the data structure.

Parameters

<i>solIdx</i>	is the solution index
<i>numberOfMatrixVar_</i>	holds the number of matrixVar elements for which values are to be provided
<i>numberOfOtherMatrixVariableResults_</i>	holds the number of <other> elements for which values are to be provided

Returns

whether the matrix variable results are set successfully or not.

4.209.2.203 `bool OSResult::setMatrixVarValuesAttributes (int solIdx, int idx, int matrixVarIdx, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry = ENUM_MATRIX_SYMMETRY_none, ENUM_MATRIX_TYPE type = ENUM_MATRIX_TYPE_unknown, std::string name = " ")`

A method to set general attributes for a matrixVar in the [i]th optimization solution, where i equals the given solution index.

Before this method is called, the [setMatrixVariableSolution\(\)](#) method has to be called first. This method then sets the matrix dimensions and other basic attributes. Further methods then store the values into the data structure.

Parameters

<i>solIdx</i>	is the solution index
<i>idx</i>	holds the index of the matrixVar (in the MatrixVariableValues array)
<i>matrixVarIdx</i>	holds the index of the matrixVar (as defined in the OS instance)
<i>numberOfRows</i>	holds the number of rows in the matrixVar
<i>numberOfColumns</i>	holds the number of columns in the matrixVar
<i>symmetry</i>	(optional) holds the type of symmetry (if not present, the default value is "none")
<i>type</i>	(optional) holds the type of values in the nonzeros of the matrix (if not present, the default value is "unknown")
<i>name</i>	(optional) holds the name of this matrixVar (the default is the empty string) for which values are to be provided

Returns

whether the matrix variable attributes are set successfully or not.

4.209.2.204 `bool OSResult::setMatrixVarValuesBlockStructure (int solIdx, int idx, int * colOffset, int colOffsetSize, int * rowOffset, int rowOffsetSize, int numberOfBlocks, int blocksConstructorIdx = 0)`

A method to set the block structure for the values of a matrixVar in the [i]th optimization solution, where i equals the given solution index.

Before this method is called, the [setMatrixVariableSolution\(\)](#) method has to be called first. This method then allocates the block structure for the new [MatrixVar](#) objects in the solution. Further methods then store the values into the data structure.

Parameters

<i>solIdx</i>	is the solution index
<i>idx</i>	holds the index of the matrixVar for which values are to be provided (as derived from the <values> element in matrixProgramming
<i>colOffset</i>	is an array of column offsets that define the partition of the columns within the block structure
<i>colOffsetSize</i>	gives the size of the colOffset array
<i>rowOffset</i>	is an array of row offsets that define the partition of the rows within the block structure
<i>rowOffsetSize</i>	gives the size of the rowOffset array
<i>numberOfBlocks</i>	gives the number of blocks
<i>blocks</i> ↔ <i>ConstructorIdx</i>	gives the index of the MatrixBlocks node in the array of constructors of the matrixVar. The default is 0.

Returns

whether the matrix variable block structure was set successfully or not.

4.209.2.205 `bool OSResult::setMatrixVarValuesBlockElements (int solIdx, int idx, int blkno, int blkRowIdx, int blkColIdx, int nz, int * start, int * index, MatrixElementValues * value, ENUM_MATRIX_TYPE valueType, ENUM_MATRIX_SYMMETRY symmetry = ENUM_MATRIX_SYMMETRY_none, bool rowMajor = false)`

A method to set the elements within a block of a matrixVar in the [i]th optimization solution, where i equals the given solution index.

Before this method is called, the [setMatrixVarBlockStructure\(\)](#) method has to be called first. This method then allocates the elements for the new [MatrixVar](#) blocks in the solution.

Parameters

<i>solIdx</i>	is the solution index
<i>idx</i>	holds the index of the matrixVar for which values are to be provided (as derived from the <values> element in matrixProgramming
<i>blkno</i>	is the number of the block for which elements are provided the partition of the columns within the block structure
<i>blkRowIdx</i>	gives the index of the block row in which the block is located
<i>blkColIdx</i>	gives the index of the block column in which the block is located
<i>nz</i>	gives the number of (nonzero) values
<i>start</i>	gives the start elements (column or row starts, depending on whether rowMajor is false or true)
<i>index</i>	gives the array of row or column (depending on rowMajor) indices
<i>value</i>	gives the data structure for (nonzero) values that need to be stored
<i>valueType</i>	gives the type of values (see OSParameters.h)
<i>symmetry</i>	gives the form of symmetry. (The default is NONE.)
<i>rowMajor</i>	indicates whether the elements are stored column by column (if rowMajor is false) or row by row (if rowMajor is true). The default is rowMajor = false.

Remarks

each block object can handle only one type of elements, although different blocks may contain different types of values

Returns

whether the matrix variable block structure was set successfully or not.

4.209.2.206 `bool OSResult::setMatrixVariablesOtherResultGeneralAttributes (int solldx, int idx, std::string name, std::string description, std::string value, std::string type, std::string solver, std::string category, int numberOfMatrixVar = 0, std::string matrixType = "", int numberOfEnumerations = 0, std::string enumType = "")`

A method to set general attributes for another (non-standard/solver specific) result associated with the matrix variables in the [i]th optimization solution, where i equals the given solution index.

Before this method is called, the [setMatrixVariableSolution\(\)](#) method has to be called first. This method then sets the matrix dimensions and other basic attributes. Further methods then store the values into the data structure.

Parameters

<i>solldx</i>	is the solution index
<i>idx</i>	holds the index of the other result (in the array of <other> solutions)
<i>name</i>	holds the name of the <other> result
<i>description</i>	can be used to hold a further description of the result
<i>value</i>	holds a scalar value associated with the <other> result
<i>type</i>	describes the type of value represented by the scalar result
<i>solver</i>	gives the solver with which this result is associated
<i>category</i>	can be used to specify a further category within the solver
<i>numberOf↵ MatrixVar</i>	gives the number of matrixVar elements associated with this result. This argument is optional and defaults to 0.
<i>matrixType</i>	can be used to associate a type with the values of the matrixVar elements. This argument is optional and defaults to the empty string ("").
<i>numberOf↵ Enumerations</i>	gives the number of levels associated with an enumeration of matrixVar elements pertaining to this result. The argument is optional and defaults to 0.
<i>enumType</i>	can be used to associate a type with the values of the enumeration. This argument is optional and defaults to the empty string ("").

Returns

whether the attributes are set successfully or not.

4.209.2.207 `bool OSResult::setMatrixVariablesOtherResultMatrixAttributes (int solldx, int otherIdx, int matrixVarIdx, int numberOfRows, int numberOfColumns, ENUM_MATRIX_SYMMETRY symmetry = ENUM_MATRIX_SYMMETRY_none, ENUM_MATRIX_TYPE type = ENUM_MATRIX_TYPE_unknown, std::string name = "")`

A method to set attributes for a matrixVar in the [j]th other result associated with matrix variables in the the [i]th optimization solution, where i equals the given solution index and j equals a given other result index.

Before this method is called, the [setMatrixVariableSolution\(\)](#) and [setMatrixVariablesOtherResultGeneralAttributes\(\)](#) methods have to be called. This method then sets the matrix dimensions and other basic attributes. Further methods then store the values into the data structure.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	holds the index of the other matrix variables result
<i>matrixVarIdx</i>	holds the index of the matrixVar (as defined in the OS instance)
<i>numberOfRows</i>	holds the number of rows in the matrixVar
<i>numberOfColumns</i>	holds the number of columns in the matrixVar
<i>symmetry</i>	(optional) holds the type of symmetry (if not present, the default value is "none")
<i>type</i>	(optional) holds the type of values in the nonzeros of the matrix (if not present, the default value is "unknown")
<i>name</i>	(optional) holds the name of this matrixVar (the default is the empty string) for which values are to be provided

Returns

whether the matrix variable attributes are set successfully or not.

4.209.2.208 `bool OSResult::setMatrixVariablesOtherResultBlockStructure (int solldx, int otherIdx, int matrixVarIdx, int * colOffset, int colOffsetSize, int * rowOffset, int rowOffsetSize, int numberOfBlocks, int blocksConstructorIdx = 0)`

A method to set the block structure for the values of a matrixVar associated with the [j]th "other" result of the [i]th optimization solution, where i equals the given solution index, and j equals the index of the other result.

Before this method is called, the [setMatrixVariableSolution\(\)](#) and [setMatrixVariablesOtherSolutionAttributes\(\)](#) methods have to be called. This method then allocates the block structure for the new [MatrixVar](#) objects in the solution. Further methods then store the values into the data structure.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	holds the index of the other solution
<i>matrixVarIdx</i>	holds the index of the matrixVar for which values are to be provided
<i>colOffset</i>	is an array of column offsets that define the partition of the columns within the block structure
<i>colOffsetSize</i>	gives the size of the colOffset array
<i>rowOffset</i>	is an array of row offsets that define the partition of the rows within the block structure
<i>rowOffsetSize</i>	gives the size of the rowOffset array
<i>numberOfBlocks</i>	gives the number of blocks
<i>blocksConstructorIdx</i>	gives the index of the MatrixBlocks node in the array of constructors of the matrixVar. The default is 0.

Returns

whether the matrix variable block structure was set successfully or not.

4.209.2.209 `bool OSResult::setMatrixVariablesOtherResultBlockElements (int solldx, int otherIdx, int matrixVarIdx, int blkno, int blkRowIdx, int blkColIdx, int nz, int * start, int * index, MatrixElementValues * value, ENUM_MATRIX_TYPE valueType, ENUM_MATRIX_SYMMETRY symmetry = ENUM_MATRIX_SYMMETRY_none, bool rowMajor = false)`

A method to set the elements within a block of a matrixVar associated with the [j]th "other" result in the [i]th optimization solution, where i equals the given solution index and j equals the index of the other result.

Before this method is called, the [setMatrixVariablesOtherResultBlockStructure\(\)](#) method has to be called first. This method then allocates the elements for the new [MatrixVar](#) blocks in the solution.

Parameters

<i>solldx</i>	is the solution index
<i>otherIdx</i>	holds the index of the other solution
<i>matrixVarIdx</i>	holds the index of the matrixVar for which values are to be provided
<i>blkno</i>	is the number of the block for which elements are provided the partition of the columns within the block structure
<i>blkRowIdx</i>	gives the index of the block row in which the block is located
<i>blkColIdx</i>	gives the index of the block column in which the block is located
<i>nz</i>	gives the number of (nonzero) values
<i>start</i>	gives the start elements (column or row starts, depending on whether rowMajor is false or true)
<i>index</i>	gives the array of row or column (depending on rowMajor) indices
<i>value</i>	gives the data structure for (nonzero) values that need to be stored
<i>valueType</i>	gives the type of values (see OSParameters.h)
<i>symmetry</i>	gives the form of symmetry. (The default is NONE.)
<i>rowMajor</i>	indicates whether the elements are stored column by column (if rowMajor is false) or row by row (if rowMajor is true). The default is rowMajor = false.

Remarks

each block object can handle only one type of elements, although different blocks may contain different types of values

Returns

whether the matrix variable block structure was set successfully or not.

4.209.2.210 `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>numberOf↔ OtherSolution↔ Results</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\(int\)](#)

4.209.2.211 `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.

4.209.2.212 `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.

4.209.2.213 `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.

4.209.2.214 `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>otherIdx</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.

4.209.2.215 `bool OSResult::setOtherSolutionResultNumberOfItems (int solldx, int otherIdx, 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>otherIdx</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.

4.209.2.216 `bool OSResult::setOtherSolutionResultItem (int solldx, int otherIdx, int itemIdx, 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.

4.209.2.217 `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.

4.209.2.218 `bool OSResult::setNumberOfSolverOutputs (int numberOfSolverOutputs)`

Set the number of other solver outputs.

Parameters

<i>numberOfOtherSolverOutputs</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.

4.209.2.219 `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.

4.209.2.220 `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.

4.209.2.221 `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.

4.209.2.222 `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.

4.209.2.223 `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.

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

- [OSResult.h](#)

4.210 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.

4.210.1 Detailed Description

Reads an optimization result and stores result and solution in a Gams Modeling Object.

Definition at line 20 of file OSosrl2gams.hpp.

4.210.2 Constructor & Destructor Documentation

4.210.2.1 OSrL2Gams::OSrL2Gams (struct gmoRec * gmo_)

Constructor.

Parameters

<i>gmo_</i>	GMO handler.
-------------	--------------

4.210.3 Member Function Documentation

4.210.3.1 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.
-----------------	--------------------------------

4.210.3.2 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:

- OSosrl2gams.hpp

4.211 OSrLParserData Class Reference

The [OSrLParserData](#) Class.

```
#include <OSrLParserData.h>
```

Collaboration diagram for OSrLParserData:

Public Member Functions

- [OSrLParserData](#) ()
the [OSrLParserData](#) class constructor

Public Attributes

- std::string [statusType](#)
the status type of the result
- std::string [statusDescription](#)
the status Description of the solution
- double [timeValue](#)
the next few variables store a time measurement and associated attribute values
- int [numberOfTimes](#)
There could be more than one time measurement; numberOfTimes stores the number of them.
- std::string [tmpOtherValue](#)
Provide temporary storage for attribute values associated with an [OtherVarResult](#).
- std::string [itemContent](#)
Provide temporary storage for a single <record> contained in an [OtherSolutionResult](#).
- void * [scanner](#)
scanner is used to store data in a reentrant lexer we use this to pass an [OSrLParserData](#) object to the parser
- unsigned int [numberOfSolutions](#)
number of result solutions
- int [numberOfVariables](#)
total number of variables in the model instance
- int [numberOfConstraints](#)
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 [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*
- int [numberOfOtherMatrixProgrammingResults](#)
 - the number of types of matrix programming results other than those associated with matrix variables, matrix objectives or matrix constraints*
- int [numberOfOtherMatrixVariableResults](#)
 - the number of types of matrix variable results other than the value of the matrix variable*
- 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 variables to track which of the attributes have been found in the present list.*
- std::string [categoryAttribute](#)

many attributes, particularly those that return strings, are used multiple times, and the parser uses generic constructs for them.

- bool [generalStatusPresent](#)
set general...Present to true if the corresponding element (child of the <general> element) has been parsed
- bool [systemInformationPresent](#)
set system...Present to true if the corresponding element (child of the <system> element) has been parsed
- bool [serviceCurrentStatePresent](#)
set service...Present to true if the corresponding element (child of the <service> element) has been parsed
- bool [jobStatusPresent](#)
set job...Present to true if the corresponding element (child of the <job> element) has been parsed
- struct [IndexValuePair](#) * [primalValPair](#)
for each solution we will build a vector of index-value pairs of primal values
- struct [IndexValuePair](#) * [objValPair](#)
for each solution we will build a vector of index-value pairs of objective function values
- struct [IndexValuePair](#) * [dualValPair](#)
for each solution we will build a vector of index-value pairs of dual values
- 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

4.211.1 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.

4.211.2 Member Data Documentation

4.211.2.1 int OSrLParserData::kounter

a temporary counter to count variables, number of attributes, etc.

Definition at line 166 of file OSrLParserData.h.

4.211.2.2 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 232 of file OSrLParserData.h.

4.211.2.3 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 259 of file OSrLParserData.h.

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

- [OSrLParserData.h](#)

4.212 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.

4.212.1 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.

4.212.2 Member Function Documentation

4.212.2.1 [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:

- [OSrLReader.h](#)

4.213 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, colIdx, rowIdx, or values) from linear constraints coefficients encoded in base64 binary

4.213.1 Detailed Description

Take an [OSResult](#) object and write a string that validates against OSrL.

Definition at line 30 of file OSrLWriter.h.

4.213.2 Member Function Documentation

4.213.2.1 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.

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

- [OSrLWriter.h](#)

4.214 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 [OSSmartPtr](#)s.
- 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 [OSSmartPtr](#)s.
- 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.

4.214.1 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 [OSSmartPtr](#)s to A and B, but A and B would have Weak [OSSmartPtr](#)s to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak [OSSmartPtr](#)s 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 [OSSmartPtr](#)s 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](#).

4.214.2 Constructor & Destructor Documentation

4.214.2.1 `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](#).

4.214.3 Member Function Documentation

4.214.3.1 `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.

4.214.3.2 `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.

4.214.4 Friends And Related Function Documentation

4.214.4.1 `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.

Definition at line 507 of file OSSmartPtr.hpp.

4.214.4.2 `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](#).

Definition at line 514 of file OSSmartPtr.hpp.

4.214.4.3 `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.

Definition at line 528 of file OSSmartPtr.hpp.

4.214.4.4 `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.

Definition at line 535 of file OSSmartPtr.hpp.

4.214.4.5 `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.

Definition at line 452 of file OSSmartPtr.hpp.

4.214.4.6 `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)`

Definition at line 465 of file `OSSmartPtr.hpp`.

4.214.4.7 `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)`

Definition at line 471 of file `OSSmartPtr.hpp`.

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

- `OSSmartPtr.hpp`

4.215 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](#)

4.215.1 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.

4.215.2 Constructor & Destructor Documentation

4.215.2.1 OSSolverAgent::OSSolverAgent (std::string *solverURI*)

Default constructor.

Parameters

<i>solverURI</i>	is the location of remote solver or scheduler
------------------	---

4.215.3 Member Function Documentation

4.215.3.1 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](#).

4.215.3.2 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](#).

4.215.3.3 `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](#).

4.215.3.4 `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](#).

4.215.3.5 `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](#).

4.215.3.6 `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](#).

4.215.3.7 `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:

- [OSSolverAgent.h](#)

4.216 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

4.216.1 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.

4.216.2 Member Function Documentation

4.216.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.216.2.2 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

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

- [OSOption.h](#)

4.217 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.

4.217.1 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 1709 of file OSResult.h.

4.217.2 Member Function Documentation**4.217.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.218 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
- std::string [enumType](#)
type of the values in the enumeration array

4.218.1 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.

4.218.2 Member Function Documentation

4.218.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

4.218.2.2 `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

4.218.2.3 `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

4.218.2.4 `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

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

- [OSOption.h](#)

4.219 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
- std::string [solver](#)
the solver of the result value
- std::string [category](#)
the category of the result value
- std::string [conType](#)
type of the values in the con array
- std::string [enumType](#)
type of the values in the enumeration array

4.219.1 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 1767 of file OSResult.h.

4.219.2 Member Function Documentation**4.219.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.220 OtherMatrixVariableResult Class Reference

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

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

Collaboration diagram for OtherMatrixVariableResult:

Public Member Functions

- [OtherMatrixVariableResult](#) ()
The standard [MatrixVariableValues](#) class constructor.
- [OtherMatrixVariableResult](#) (std::string name_)
Altenname [MatrixVariableValues](#) class constructor.
- [~OtherMatrixVariableResult](#) ()
The [MatrixVariableValues](#) class destructor.
- bool [isEqual](#) ([OtherMatrixVariableResult](#) *that)
A function to check for the equality of two objects.

Public Attributes

- std::string [name](#)
Gives a name to this result.
- std::string [description](#)
other data elements are optional

- int [numberOfMatrixVar](#)
number of matrix variables affected by or associated with this result
- std::string [matrixType](#)
the type of matrixVar
- [OSMatrixWithMatrixVarIdx](#) ** [matrixVar](#)
the list of matrices and their values
- int [numberOfEnumerations](#)
number of levels in an enumeration associated with this result
- std::string [enumType](#)
the type of the enumeration
- [OtherOptionOrResultEnumeration](#) ** [enumeration](#)
the enumeration.

4.220.1 Detailed Description

The in-memory representation of the `<matrixVariables>` `<other>` element.

Definition at line 1939 of file OSResult.h.

4.220.2 Member Data Documentation

4.220.2.1 std::string OtherMatrixVariableResult::name

Gives a name to this result.

This is s mandatory data element

Definition at line 1943 of file OSResult.h.

4.220.2.2 OtherOptionOrResultEnumeration** OtherMatrixVariableResult::enumeration

the enumeration.

Each enumeration level has a list of matrixVar indices associated with that level

Definition at line 1970 of file OSResult.h.

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

- [OSResult.h](#)

4.221 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
- std::string [enumType](#)
type of the values in the enumeration array

4.221.1 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.

4.221.2 Member Function Documentation

4.221.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.221.2.2 `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

4.221.2.3 `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

4.221.2.4 `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

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

- [OSOption.h](#)

4.222 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
- `std::string` [solver](#)
the solver of the result value
- `std::string` [category](#)
the category of the result value
- `std::string` [objType](#)
type of the values in the obj array
- `std::string` [enumType](#)
type of the values in the enumeration array

4.222.1 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 1443 of file OSResult.h.

4.222.2 Member Function Documentation

4.222.2.1 `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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.223 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

4.223.1 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.

4.223.2 Member Function Documentation

4.223.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.223.2.2 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

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

- [OSOption.h](#)

4.224 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

4.224.1 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 1387 of file OSResult.h.

4.224.2 Member Function Documentation

4.224.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.225 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

4.225.1 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.

4.225.2 Member Function Documentation

4.225.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.225.2.2 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

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

- [OSOption.h](#)

4.226 OtherOptionOrResultEnumeration Class Reference

brief an integer vector data structure used in [OSOption](#) and [OSResult](#)

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

Inheritance diagram for OtherOptionOrResultEnumeration:

Collaboration diagram for OtherOptionOrResultEnumeration:

Public Member Functions

- [OtherOptionOrResultEnumeration](#) (int n)
alternate constructor
- bool [IsEqual](#) ([OtherOptionOrResultEnumeration](#) *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](#) ([OtherOptionOrResultEnumeration](#) *that)
A function to make a deep copy of an instance of this class.
- bool [setOtherOptionOrResultEnumeration](#) (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.

Additional Inherited Members

4.226.1 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.

4.226.2 Member Function Documentation

4.226.2.1 `bool OtherOptionOrResultEnumeration::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 "numberOf↵XXX" 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

4.226.2.2 `bool OtherOptionOrResultEnumeration::deepCopyFrom (OtherOptionOrResultEnumeration * 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

4.226.2.3 `bool OtherOptionOrResultEnumeration::setOtherOptionOrResultEnumeration (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

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

- [OSGeneral.h](#)

4.227 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

4.227.1 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.

4.227.2 Member Function Documentation

4.227.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.227.2.2 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

4.227.2.3 bool OtherOptions::setOther (int *numberOfOptions*, OtherOption ** *other*)

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

Parameters

<i>numberOf↵ Options</i>	number of <other> elements to be set
<i>other</i>	the array of <other> elements that are to be set

4.227.2.4 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)

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

- [OSOption.h](#)

4.228 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

4.228.1 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 162 of file OSResult.h.

4.228.2 Member Function Documentation

4.228.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.229 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

4.229.1 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 216 of file OSResult.h.

4.229.2 Member Function Documentation**4.229.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.230 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)

4.230.1 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 2136 of file OSResult.h.

4.230.2 Member Function Documentation

4.230.2.1 `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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.231 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

4.231.1 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 2205 of file OSResult.h.

4.231.2 Member Function Documentation

4.231.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.232 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

4.232.1 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 2419 of file OSResult.h.

4.232.2 Member Function Documentation**4.232.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.233 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
- std::string [enumType](#)
type of the values in the enumeration array

4.233.1 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.

4.233.2 Member Function Documentation

4.233.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.233.2.2 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

4.233.2.3 `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

4.233.2.4 `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

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

- [OSOption.h](#)

4.234 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
- std::string [solver](#)
the solver of the result value
- std::string [category](#)
the category of the result value
- std::string [varType](#)
type of the values in the var array
- std::string [enumType](#)
type of the values in the enumeration array

4.234.1 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 1124 of file OSResult.h.

4.234.2 Member Function Documentation

4.234.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.235 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.

4.235.1 Detailed Description

A structure to information about an [OtherVariableResult](#) element.

Definition at line 29 of file OSrLParserData.h.

4.235.2 Member Data Documentation

4.235.2.1 `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.

4.235.2.2 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:

- [OSrLParserData.h](#)

4.236 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

4.236.1 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.

4.236.2 Member Function Documentation

4.236.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.236.2.2 `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

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

- [OSOption.h](#)

4.237 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

4.237.1 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 1065 of file OSResult.h.

4.237.2 Member Function Documentation

4.237.2.1 `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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.238 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

4.238.1 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.

4.238.2 Member Function Documentation**4.238.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.238.2.2 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

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

- [OSOption.h](#)

4.239 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

4.239.1 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.

4.239.2 Member Function Documentation

4.239.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.239.2.2 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

4.239.2.3 bool PathPairs::setPathPair (int *numberOfPathPairs*, PathPair ** *pathPair*)

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

Parameters

<i>numberOfPath↵Pairs</i>	number of <pathPair> elements to be set
<i>path</i>	the array of <pathPair> elements that are to be set

4.239.2.4 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>numberOfPath↵Pairs</i>	number of <pathPair> elements to be set

4.239.2.5 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

<code>makecopy</code>	tracks whether a copy is to be made
-----------------------	-------------------------------------

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

- [OSOption.h](#)

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

4.240.1 Detailed Description

The in-memory representation of a polar cone.

Definition at line 1465 of file OSInstance.h.

4.240.2 Member Function Documentation

4.240.2.1 `virtual std::string PolarCone::getConeName () [virtual]`

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.240.2.2 `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 "numberOfXXX" 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

4.240.2.3 `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

4.240.3 Member Data Documentation

4.240.3.1 `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 1486 of file `OSInstance.h`.

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

- [OSInstance.h](#)

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

4.241.1 Detailed Description

The in-memory representation of a polyhedral cone.

Definition at line 786 of file OSInstance.h.

4.241.2 Member Function Documentation

4.241.2.1 virtual std::string PolyhedralCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.241.2.2 `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](#).

4.241.2.3 `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 "numberOf↵XXX" 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

4.241.2.4 `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

4.241.3 Member Data Documentation4.241.3.1 `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`.

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

- [OSInstance.h](#)

4.242 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

4.242.1 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.

4.242.2 Member Function Documentation

4.242.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.242.2.2 `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

4.242.2.3 `bool Processes::setProcess (int numberOfProcesses, std::string * process)`

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

Parameters

<i>numberOf↵Processes</i>	number of <process> elements to be set
<i>path</i>	the array of <process> elements that are to be set

4.242.2.4 `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
----------------	-----------------------------------

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

- [OSOption.h](#)

4.243 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 [numberOfOtherIndexes](#)

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

- 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

4.243.1 Detailed Description

The in-memory representation of a product cone.

Definition at line 1248 of file `OSInstance.h`.

4.243.2 Member Function Documentation

4.243.2.1 virtual std::string ProductCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.243.2.2 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](#).

4.243.2.3 `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 "numberOf↵XXX" 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

4.243.2.4 `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

4.243.3 Member Data Documentation

4.243.3.1 `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 1269 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.244 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

4.244.1 Detailed Description

The in-memory representation of the [quadraticCoefficients](#) element.

Definition at line 380 of file [OSInstance.h](#).

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

- [OSInstance.h](#)

4.245 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 [numberOfOtherIndexes](#)
Multidimensional tensors can also form cones (the Kronecker product, for instance, can be thought of as a four-dimensional tensor).
- 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 [axisDirection](#)
*The index of the first component can be changed Since there are possibly many dimensions, the index is coded as $i_0 * n_1 * n_2 * \dots$*

4.245.1 Detailed Description

The in-memory representation of a quadratic cone.

Definition at line 860 of file `OSInstance.h`.

4.245.2 Member Function Documentation

4.245.2.1 virtual std::string QuadraticCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.245.2.2 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](#).

4.245.2.3 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 "numberOf↵XXX" 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

4.245.2.4 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

4.245.3 Member Data Documentation

4.245.3.1 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.

4.245.3.2 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.

4.245.3.3 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:

- [OSInstance.h](#)

4.246 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

4.246.1 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`.

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

- [OSInstance.h](#)

4.247 QuadraticTerms Class Reference

a data structure for holding quadratic terms

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

Public Member Functions

- [QuadraticTerms](#) ()
Default constructor.

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.

4.247.1 Detailed Description

a data structure for holding quadratic terms

Definition at line 431 of file OSGeneral.h.

4.247.2 Member Data Documentation

4.247.2.1 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.

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

- [OSGeneral.h](#)

4.248 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 [numberOfOtherIndexes](#)

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

- 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 [firstAxisDirection](#)

The indices of the first two component can be changed Since there are possibly many dimensions, each index is coded as $i_0*n_1*n_2*...$

4.248.1 Detailed Description

The in-memory representation of a rotated quadratic cone.

Definition at line 951 of file `OSInstance.h`.

4.248.2 Member Function Documentation

4.248.2.1 virtual std::string RotatedQuadraticCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.248.2.2 `virtual std::string RotatedQuadraticCone::getConcInXML () [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](#).

4.248.2.3 `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 "numberOf↵XXX" 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

4.248.2.4 `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

4.248.3 Member Data Documentation

4.248.3.1 `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.

4.248.3.2 `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.

4.248.3.3 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.

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

- [OSInstance.h](#)

4.249 ScalarExpressionTree Class Reference

Used to hold part of the instance in memory.

```
#include <OSExpressionTree.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.

4.249.1 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.

4.249.2 Member Function Documentation

4.249.2.1 `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.

4.249.2.2 `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.

4.249.2.3 `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.

4.249.2.4 `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.

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

- [OSExpressionTree.h](#)

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

4.250.1 Detailed Description

The in-memory representation of a cone of semidefinite matrices.

Definition at line 1046 of file OSInstance.h.

4.250.2 Member Function Documentation

4.250.2.1 virtual std::string SemidefiniteCone::getConeName () [virtual]

Returns

the type of cone as a string

Reimplemented from [Cone](#).

4.250.2.2 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](#).

4.250.2.3 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 "numberOf↵XXX" 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

4.250.2.4 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

4.250.3 Member Data Documentation

4.250.3.1 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.

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

- [OSInstance.h](#)

4.251 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

4.251.1 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.

4.251.2 Member Function Documentation

4.251.2.1 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 "numberOf← XXX" attributes and <XXX> children)

4.251.2.2 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

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

- [OSOption.h](#)

4.252 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

4.252.1 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 415 of file OSResult.h.

4.252.2 Member Function Documentation

4.252.2.1 `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 "numberOfXXX" attributes and <XXX> children)
-------------------	--

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

- [OSResult.h](#)

4.253 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

4.253.1 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.

4.253.2 Member Function Documentation

4.253.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.253.2.2 `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

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

- [OSOption.h](#)

4.254 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

4.254.1 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.

4.254.2 Member Function Documentation**4.254.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.254.2.2 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

4.254.2.3 bool SolverOptions::setSolverOptions (int *numberOfOptions*, SolverOption ** *solverOption*)

A function to set an array of solver options.

Parameters

<i>numberOf↵ Options</i>	number of solver options to be set
<i>solverOption</i>	the array of solver options that are to be set

4.254.2.4 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)

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

- [OSOption.h](#)

4.255 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)

4.255.1 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 2354 of file OSResult.h.

4.255.2 Member Function Documentation**4.255.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.256 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

4.256.1 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.

4.256.2 Member Function Documentation

4.256.2.1 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 "numberOfXXX" attributes and <XXX> children)
-------------------	--

4.256.2.2 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

4.256.2.3 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

4.256.2.4 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

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

- [OSOption.h](#)

4.257 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

4.257.1 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.

4.257.2 Member Function Documentation

4.257.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.257.2.2 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

4.257.2.3 bool SOSWeights::setVar (int numberOfVar, BranchingWeight ** 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>

4.257.2.4 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

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

- [OSOption.h](#)

4.258 SparseHessianMatrix Class Reference

The in-memory representation of a [SparseHessianMatrix](#).

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.

4.258.1 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.

4.258.2 Constructor & Destructor Documentation

4.258.2.1 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.

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

- [OSGeneral.h](#)

4.259 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.

4.259.1 Detailed Description

a sparse vector data structure for integer vectors

Definition at line 171 of file OSGeneral.h.

4.259.2 Constructor & Destructor Documentation

4.259.2.1 SparseIntVector::SparseIntVector (int *number*)

Constructor.

Parameters

<i>number</i>	holds the size of the vector.
---------------	-------------------------------

4.259.3 Member Data Documentation

4.259.3.1 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.

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

- [OSGeneral.h](#)

4.260 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

4.260.1 Detailed Description

a sparse Jacobian matrix data structure

Definition at line 300 of file OSGeneral.h.

4.260.2 Constructor & Destructor Documentation

4.260.2.1 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.

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

- [OSGeneral.h](#)

4.261 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.

4.261.1 Detailed Description

a sparse matrix data structure

Definition at line 223 of file OSGeneral.h.

4.261.2 Constructor & Destructor Documentation

4.261.2.1 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.

4.261.3 Member Function Documentation

4.261.3.1 bool SparseMatrix::display (int *secondaryDim*)

This method displays data structure in the matrix format.

Returns

4.261.4 Member Data Documentation

4.261.4.1 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.

4.261.4.2 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.

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

- [OSGeneral.h](#)

4.262 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.

4.262.1 Detailed Description

a sparse vector data structure

Definition at line 122 of file OSGeneral.h.

4.262.2 Constructor & Destructor Documentation

4.262.2.1 SparseVector::SparseVector (int *number*)

Constructor.

Parameters

<i>number</i>	holds the size of the vector.
---------------	-------------------------------

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

- [OSGeneral.h](#)

4.263 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

4.263.1 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.

4.263.2 Member Function Documentation

4.263.2.1 `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 "numberOfXXX" attributes and <XXX> children)

4.263.2.2 `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

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

- [OSGeneral.h](#)

4.264 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

4.264.1 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.

4.264.2 Member Function Documentation

4.264.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

4.264.2.2 `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

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

- [OSOption.h](#)

4.265 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

4.265.1 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 349 of file OSResult.h.

4.265.2 Member Function Documentation**4.265.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.266 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

4.266.1 Detailed Description

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

Definition at line 2139 of file OSInstance.h.

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

- [OSInstance.h](#)

4.267 TimeDomainInterval Class Reference

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.

4.267.1 Detailed Description

Definition at line 2114 of file OSInstance.h.

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

- [OSInstance.h](#)

4.268 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

4.268.1 Detailed Description

The in-memory representation of the `<stage>` element.

Definition at line 2062 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.269 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

4.269.1 Detailed Description

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

Definition at line 1976 of file `OSInstance.h`.

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

- [OSInstance.h](#)

4.270 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

4.270.1 Detailed Description

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

Definition at line 1994 of file OSInstance.h.

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

- [OSInstance.h](#)

4.271 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

4.271.1 Detailed Description

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

Definition at line 2019 of file OSInstance.h.

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

- [OSInstance.h](#)

4.272 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

4.272.1 Detailed Description

The in-memory representation of the <**objectives**> child of the <stage> element.

Definition at line 2037 of file OSInstance.h.

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

- [OSInstance.h](#)

4.273 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

4.273.1 Detailed Description

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

Definition at line 2091 of file OSInstance.h.

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

- [OSInstance.h](#)

4.274 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

4.274.1 Detailed Description

The in-memory representation of the **element**.

Definition at line 1933 of file OSInstance.h.

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

- [OSInstance.h](#)

4.275 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

4.275.1 Detailed Description

The in-memory representation of the `<variables>` child of the `<stage>` element.

Definition at line 1951 of file OSInstance.h.

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

- [OSInstance.h](#)

4.276 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.

4.276.1 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 546 of file OSResult.h.

4.276.2 Member Function Documentation

4.276.2.1 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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.277 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

4.277.1 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.

4.277.2 Member Function Documentation

4.277.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

4.277.2.2 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

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

- [OSGeneral.h](#)

4.278 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.

4.278.1 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 604 of file OSResult.h.

4.278.2 Member Function Documentation

4.278.2.1 `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 "numberOf← XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.279 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

4.279.1 Detailed Description

The in-memory representation of the **variable** element.

Definition at line 44 of file OSInstance.h.

4.279.2 Member Data Documentation

4.279.2.1 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.

4.279.2.2 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.

4.279.2.3 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.

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

- [OSInstance.h](#)

4.280 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

4.280.1 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.

4.280.2 Member Function Documentation

4.280.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

4.280.2.2 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

4.280.2.3 bool VariableOption::setOther (int *numberOfOptions*, OtherVariableOption ** *other*)

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

Parameters

<i>numberOf↵ Options</i>	number of <other> elements to be set
<i>other</i>	the array of <other> elements that are to be set

4.280.2.4 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
--------------	--

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

- [OSOption.h](#)

4.281 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.

4.281.1 Detailed Description

The in-memory representation of the **variables** element.

Definition at line 83 of file OSInstance.h.

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

- [OSInstance.h](#)

4.282 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

4.282.1 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 1217 of file OSResult.h.

4.282.2 Member Function Documentation

4.282.2.1 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 "numberOf↵XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.283 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

4.283.1 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 901 of file OSResult.h.

4.283.2 Member Function Documentation

4.283.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.284 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

4.284.1 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 1011 of file OSResult.h.

4.284.2 Member Function Documentation

4.284.2.1 `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 "numberOf↵ XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.285 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

- 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](#) * [value](#)
The variable references (indexes of core variables) of the elements.

4.285.1 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 835 of file OSMatrix.h.

4.285.2 Member Function Documentation

4.285.2.1 virtual ENUM_MATRIX_CONSTRUCTOR_TYPE [VarReferenceMatrixElements::getNodeType](#) () [virtual]

Returns

the value of nType

Reimplemented from [MatrixNode](#).

4.285.2.2 virtual ENUM_MATRIX_TYPE [VarReferenceMatrixElements::getMatrixType](#) () [virtual]

Returns

the type of the matrix elements

Implements [MatrixNode](#).

4.285.2.3 virtual std::string [VarReferenceMatrixElements::getNodeName](#) () [virtual]

Returns

the name of the matrix constructor

Implements [MatrixNode](#).

4.285.2.4 `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](#).

4.285.2.5 `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](#).

4.285.2.6 `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](#).

4.285.2.7 `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 "numberOf↵XXX" 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

4.285.2.8 `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

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

- [OSMatrix.h](#)

4.286 VarReferenceMatrixValues Class Reference

A concrete class that is used to store a specific type of matrix values, references to variable indexes defined in the core section.

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

Inheritance diagram for VarReferenceMatrixValues:

Collaboration diagram for VarReferenceMatrixValues:

Public Member Functions

- 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.
- virtual 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.

4.286.1 Detailed Description

A concrete class that is used to store a specific type of matrix values, references to variable indexes defined in the core section.

Definition at line 563 of file OSMatrix.h.

4.286.2 Member Function Documentation

4.286.2.1 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 "numberOf↵XXX" 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

4.286.2.2 virtual bool VarReferenceMatrixValues::deepCopyFrom (VarReferenceMatrixValues * *that*) [virtual]

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:

- [OSMatrix.h](#)

4.287 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

4.287.1 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 847 of file OSResult.h.

4.287.2 Member Function Documentation

4.287.2.1 `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 "numberOfXXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.288 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

4.288.1 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 956 of file OSResult.h.

4.288.2 Member Function Documentation

4.288.2.1 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 "numberOf↵ XXX" attributes and <XXX> children)

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

- [OSResult.h](#)

4.289 WSUtil Class Reference

Used by [OSSolverAgent](#) client for help in invoking a remote solver.

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 servicePortNumber)
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

4.289.1 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.

4.289.2 Constructor & Destructor Documentation

4.289.2.1 WSUtil::WSUtil ()

Default constructor.

Parameters

<i>solverURI</i>	is the location of remote solver or scheduler
------------------	---

4.289.3 Member Function Documentation

4.289.3.1 `static std::string WSUtil::sendSOAPMessage (std::string theSOAP, std::string servicIP, 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>solverPortNumber</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.

4.289.3.2 `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.

4.289.3.3 `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.

4.289.3.4 `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.

4.289.3.5 `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.

4.289.3.6 `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:

- [OSWSUtil.h](#)

4.290 YYLTYPE Struct Reference

4.290.1 Detailed Description

Definition at line 874 of file OSParseosil.tab.hpp.

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

- OSParseosil.tab.hpp
- OSParseosol.tab.hpp
- OSParseosrl.tab.hpp

4.291 YYSTYPE Union Reference

4.291.1 Detailed Description

Definition at line 854 of file OSParseosil.tab.hpp.

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

- OSParseosil.tab.hpp
- OSParseosol.tab.hpp
- OSParseosrl.tab.hpp

Chapter 5

File Documentation

5.1 OSCommandLine.h File Reference

```
#include "OSInstance.h"  
#include "OSOption.h"  
#include <string>
```

Include dependency graph for OSCommandLine.h:

5.2 OSCommandLineReader.h File Reference

```
#include "OSCommandLine.h"  
#include "OSErrorClass.h"  
#include <string>
```

Include dependency graph for OSCommandLineReader.h:

Classes

- class [OSCommandLineReader](#)
The [OSCommandLineReader](#) Class.

5.2.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin

Remarks

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.

5.3 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.

5.3.1 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.

5.4 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 "parameters.h"
#include <cstdlib>
#include <stdlib.h>
#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.

5.4.1 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.

5.5 OSExpressionTree.h File Reference

```
#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.

5.5.1 Detailed Description

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.

5.6 OSGeneral.h File Reference

```
#include "OSConfig.h"
#include "OSParameters.h"
#include "OSnLNode.h"
#include "OSExpressionTree.h"
#include <string>
#include <vector>
```

Include dependency graph for OSGeneral.h: This graph shows which files directly or indirectly include this file:

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 [OtherOptionOrResultEnumeration](#)

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](#)

5.6.1 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.

5.7 OSgLParseData.h File Reference

```
#include "OSGeneral.h"
#include "OSMatrix.h"
#include <stdio.h>
#include <string>
```

Include dependency graph for OSgLParseData.h: This graph shows which files directly or indirectly include this file:

Classes

- class [OSgLParseData](#)

The [OSgLParseData](#) Class.

5.7.1 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.

5.8 OSGLWriter.h File Reference

```
#include "OSGeneral.h"
#include "OSILWriter.h"
#include "OSInstance.h"
#include "OSParameters.h"
#include "OSBase64.h"
#include "OSMathUtil.h"
#include <string>
#include <sstream>
Include dependency graph for OSGLWriter.h:
```

Functions

- std::string [writeIntVectorData](#) ([IntVector](#) *v, bool addWhiteSpace, bool writeBase64)
Take an [IntVector](#) object and write a string that validates against the OSGL schema.
- std::string [writeGeneralFileHeader](#) ([GeneralFileHeader](#) *v, bool addWhiteSpace)
Take a [GeneralFileHeader](#) object and write a string that validates against the OSGL schema.
- std::string [writeOtherOptionOrResultEnumeration](#) ([OtherOptionOrResultEnumeration](#) *e, bool addWhiteSpace, bool writeBase64)
Take an [OtherOptionOrResultEnumeration](#) object and write a string that validates against the OSGL schema.
- 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.

5.8.1 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.

5.8.2 Function Documentation

5.8.2.1 `std::string writeIntVectorData (IntVector * v, bool addWhiteSpace, bool writeBase64)`

Take an [IntVector](#) object and write a string that validates against the OSgL schema.

Parameters

<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)

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)

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)

5.8.2.2 `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

5.8.2.3 `std::string writeOtherOptionOrResultEnumeration (OtherOptionOrResultEnumeration * e, bool addWhiteSpace, bool writeBase64)`

Take an [OtherOptionOrResultEnumeration](#) object and write a string that validates against the OSgL schema.

Parameters

<i>e</i>	is the OtherOptionOrResultEnumeration 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)

5.8.2.4 `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)

5.8.2.5 `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 <el> (including mult and incr attributes)

5.9 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.

5.9.1 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.

5.10 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.

5.10.1 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.

5.11 OSiLReader.h File Reference

```
#include "OSInstance.h"
```

```
#include "OSiLParserData.h"
```

```
#include "OSgLParserData.h"
```

```
#include "OSnLParserData.h"
```

```
#include "OSErrorClass.h"
```

```
#include <string>
```

Include dependency graph for OSiLReader.h: This graph shows which files directly or indirectly include this file:

Classes

- class [OSiLReader](#)
Used to read an OSiL string.

5.11.1 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.

5.12 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.

5.12.1 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.

5.13 OSInstance.h File Reference

This file defines the [OSInstance](#) class along with its supporting classes.

```
#include "OSConfig.h"
#include "OSParameters.h"
#include "OSGeneral.h"
#include "OSMatrix.h"
#include "OSnLNode.h"
#include "OSExpressionTree.h"
#include <string>
#include <map>
```

Include dependency graph for OSInstance.h: This graph shows which files directly or indirectly include this file:

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.

5.13.1 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 <nI> element is called ExpressionTree
3. Root is not called osil; it is called osinstance

5.14 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 [Matrix↔Blocks](#)) 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 $c \cdot x_{\{k\}}$, 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
- 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](#)
A concrete class that is used to store a specific type of matrix values, references to variable indexes defined in the core section.
- 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 [MixedRowReferenceMatrixElements](#)
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 [OSMatrixWithMatrixVarIdx](#)
this class extends [OSMatrix](#) for use, e.g., in the matrixVar section of OSoL and OSrL
- class [OSMatrixWithMatrixObjIdx](#)
this class extends [OSMatrix](#) for use, e.g., in the matrixObj section of OSoL and OSrL
- class [OSMatrixWithMatrixConIdx](#)
this class extends [OSMatrix](#) for use, e.g., in the matrixCon section of OSoL and OSrL
- class [MatrixBlock](#)
a data structure to represent a [MatrixBlock](#) object (derived from [MatrixType](#))

Functions

- [LinearMatrixElement](#) * [convertToLinearMatrixElement](#) (double val)
Some methods to convert one type of matrix element into another.

5.14.1 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.

5.15 OSmps2OS.h File Reference

```
#include <CoinMpsIO.hpp>
#include <CoinPackedMatrix.hpp>
#include <string>
#include "OSInstance.h"
#include "OSOption.h"
#include "OSoLReader.h"
Include dependency graph for OSmps2OS.h:
```

Classes

- class [OSmps2OS](#)

The [OSmps2OS](#) Class.

5.15.1 Detailed Description

Author

Robert Fourer, Horand Gassmann, Jun Ma, Kipp Martin

Remarks

Copyright (C) 2005-2013, 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.

5.16 OSmps2osil.h File Reference

```
#include <CoinMpsIO.hpp>
#include <CoinPackedMatrix.hpp>
#include <string>
#include "OSInstance.h"
Include dependency graph for OSmps2osil.h:
```

Classes

- class [OSmps2osil](#)

The [OSmps2osil](#) Class.

5.16.1 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.

5.17 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.

5.17.1 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) 2005-2015, 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](#).

5.18 OSnLParserData.h File Reference

```
#include "OSnLNode.h"
```

```
#include <vector>
```

Include dependency graph for OSnLParserData.h: This graph shows which files directly or indirectly include this file:

Classes

- class [OSnLParserData](#)
The [OSnLParserData](#) Class.

5.18.1 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.

5.19 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.

5.19.1 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.

5.20 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.

5.20.1 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.

5.21 OSoLWriter.h File Reference

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

Include dependency graph for OSoLWriter.h:

Classes

- class [OSoLWriter](#)

Take an [OOption](#) object and write a string that validates against the OSoL schema.

5.21.1 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.

5.22 OOption.h File Reference

```
#include <string>
#include <vector>
#include "OSGeneral.h"
#include "OSMathUtil.h"
```

Include dependency graph for OOption.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.

5.22.1 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.

5.23 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.

5.23.1 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.

5.24 OSosrl2ampl.h File Reference

```
#include "OSResult.h"
#include "OSMathUtil.h"
#include <string>
#include <vector>
Include dependency graph for OSosrl2ampl.h:
```

Classes

- class [OSosrl2ampl](#)

The [OSosrl2ampl](#) Class.

5.24.1 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.

5.25 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.

5.25.1 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.

5.26 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_NEAR_EQUAL](#) 1e-2
we use OS_NEAR_EQUAL in unitTest to see if we are close to the optimal obj value

Enumerations

- enum [ENUM_OUTPUT_LEVEL](#)
Enumeration for the different verbosity levels that can be used in producing output.
- enum [ENUM_OUTPUT_AREA](#)
Enumeration for the different areas that can produce output.
- enum [ENUM_BASIS_STATUS](#)
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_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.
- enum [ENUM_CONREFERENCE_VALUETYPE](#)

An enum to track the type of value contained in a reference to a constraint.

- enum `ENUM_COMBINE_ARRAYS`

An enum to streamline `set()` methods of vectors.

Functions

- bool `OSIsnan` (double x)
checks whether a given double is NaN
- double `OSNaN` ()
returns the value for NaN used in OS
- `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.

5.26.1 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.

5.26.2 Enumeration Type Documentation

5.26.2.1 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.

Definition at line 107 of file `OSParameters.h`.

5.26.2.2 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

Definition at line 128 of file `OSParameters.h`.

5.26.2.3 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 `OSosl2ampl.cpp` for an application.)

Definition at line 456 of file `OSParameters.h`.

5.27 OSResult.h File Reference

```
#include <string>
#include <vector>
#include "OSGeneral.h"
#include "OSMatrix.h"
```

Include dependency graph for OSResult.h: This graph shows which files directly or indirectly include this file:

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 [MatrixVariableValues](#)
The in-memory representation of the `<matrixVariables>` element.
- class [OtherMatrixVariableResult](#)
The in-memory representation of the `<matrixVariables>` `<other>` element.
- class [MatrixVariableSolution](#)
The in-memory representation of the `<MatrixVariableSolution>` element.
- class [MatrixObjectiveSolution](#)
The in-memory representation of the `<MatrixVariableSolution>` element.
- class [MatrixConstraintSolution](#)
The in-memory representation of the `<MatrixConstraintSolution>` element.
- class [MatrixProgrammingSolution](#)
The in-memory representation of the `<MatrixProgrammingSolution>` element.
- 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.

5.27.1 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.

5.28 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.

5.28.1 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.

5.29 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.

5.29.1 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.

5.30 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.

5.30.1 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.

5.31 OSRunSolver.h File Reference

```
#include "OSDefaultSolver.h"
```

Include dependency graph for OSRunSolver.h:

Functions

- `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.

5.31.1 Detailed Description

Author

Horand Gassmann, Jun Ma, Kipp Martin,

Remarks

Copyright (C) 2005-2013, 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.

5.31.2 Function Documentation

5.31.2.1 `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

5.31.2.2 `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

5.31.2.3 `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

5.31.2.4 `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

5.31.2.5 `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

5.32 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.

5.32.1 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.

5.33 OSStringUtil.h File Reference

```
#include <sstream>
```

Include dependency graph for OSStringUtil.h:

Functions

- std::string [writeStringData](#) (std::string str)

writeStringData

5.33.1 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.

5.33.2 Function Documentation

5.33.2.1 `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

5.34 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.

5.34.1 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.

Index

- ~OSSmartPtr
 - OSSmartPtr, [589](#)
- AddChannel
 - OSOutput, [481](#)
- addCon
 - InitConstraintValues, [123](#)
 - InitDualVariableValues, [128](#)
 - OtherConstraintOption, [600](#)
- addCone
 - OSInstance, [318–323](#)
- addConstraint
 - OSInstance, [314](#)
- addIdx
 - BasisStatus, [46](#)
- addJobID
 - JobDependencies, [169](#)
- addMatrix
 - OSInstance, [317](#)
- addObj
 - InitObjectiveBounds, [136](#)
 - InitObjectiveValues, [139, 140](#)
 - OtherObjectiveOption, [605](#)
- addObjective
 - OSInstance, [313](#)
- addOther
 - ConstraintOption, [75](#)
 - ObjectiveOption, [245](#)
 - OtherOptions, [616](#)
 - VariableOption, [693](#)
- addPath
 - DirectoriesAndFiles, [89](#)
- addPathPair
 - PathPairs, [636](#)
- addProcess
 - Processes, [643](#)
- addQTermsToExpressionTree
 - OSInstance, [329](#)
- addQTermsToExpressionTree
 - OSInstance, [329](#)
- addSOS
 - SOSVariableBranchingWeights, [667](#)
- addSlackVars
 - LindoSolver, [177](#)
- addSolverOption
 - SolverOptions, [663](#)
- addTimingInformation
 - OSResult, [526](#)
- addVar
 - InitVariableValues, [144](#)
 - InitVariableValuesString, [148](#)
 - InitialBasisStatus, [132](#)
 - IntegerVariableBranchingWeights, [156, 158](#)
 - OtherVariableOption, [625](#)
 - SOSWeights, [669](#)
- addVariable
 - OSInstance, [312](#)
- alignsOnBlockBoundary
 - BaseMatrix, [44](#)
 - ConReferenceMatrixElements, [65](#)
 - ConstantMatrixElements, [68](#)
 - GeneralMatrixElements, [107](#)
 - LinearMatrixElements, [181](#)
 - MatrixBlock, [190](#)
 - MatrixBlocks, [193](#)
 - MatrixNode, [206](#)
 - MatrixTransformation, [216](#)
 - MatrixType, [219](#)
 - MixedRowReferenceMatrixElements, [235](#)
 - OSMatrix, [339](#)
 - ObjReferenceMatrixElements, [250](#)
 - VarReferenceMatrixElements, [700](#)
- axisDirection
 - QuadraticCone, [648](#)
- bADMustReTape
 - OSExpressionTree, [271](#)
- Base64, [41](#)
 - decodeb64, [42](#)
 - encodeb64, [42](#)
- BaseMatrix, [42](#)
 - alignsOnBlockBoundary, [44](#)
 - cloneMatrixNode, [44](#)
 - getMatrixNodeInXML, [44](#)
 - getMatrixType, [44](#)
 - getNodeName, [43](#)
 - getNodeType, [43](#)
- BasisStatus, [45](#)
 - addIdx, [46](#)
 - deepCopyFrom, [46](#)
 - getBasisDense, [48](#)
 - getEI, [46](#)

- getIntVector, [48](#)
 - getNumberOfEI, [46](#)
 - setIntVector, [46](#)
 - setRandom, [45](#)
- blockColumns
 - ExpandedMatrixBlocks, [98](#)
- blockRows
 - ExpandedMatrixBlocks, [97](#)
- blocks
 - ExpandedMatrixBlocks, [98](#)
- BonminProblem, [49](#)
 - finalize_solution, [51](#)
 - get_bounds_info, [51](#)
 - get_constraints_linearity, [50](#)
 - get_nlp_info, [50](#)
 - get_variables_linearity, [50](#)
 - get_variables_types, [50](#)
- BonminSolver, [51](#)
 - setSolverOptions, [52](#)
- BranchingWeight, [52](#)
 - deepCopyFrom, [54](#)
 - setRandom, [53](#)
- browser
 - OSCommandLine, [268](#)
 - osOptionsStruc, [477](#)
- buildSolverInstance
 - CoinSolver, [55](#)
 - CsdpSolver, [86](#)
 - IpoptSolver, [166](#)
- CPUNumber, [81](#)
 - deepCopyFrom, [83](#)
 - setRandom, [82](#)
- CPU Speed, [83](#)
 - deepCopyFrom, [84](#)
 - setRandom, [84](#)
- calculateAllConstraintFunctionGradients
 - OSInstance, [326](#)
- calculateAllConstraintFunctionValues
 - OSInstance, [325](#)
- calculateAllObjectiveFunctionGradients
 - OSInstance, [327](#)
- calculateAllObjectiveFunctionValues
 - OSInstance, [325](#), [326](#)
- calculateConstraintFunctionGradient
 - OSInstance, [326](#), [327](#)
- calculateFunction
 - OSnLNode, [379](#)
 - OSnLNodeAbs, [382](#)
 - OSnLNodeAllDiff, [384](#)
 - OSnLNodeCos, [386](#)
 - OSnLNodeDivide, [388](#)
 - OSnLNodeE, [391](#)
 - OSnLNodeErf, [392](#)
 - OSnLNodeExp, [394](#)
 - OSnLNodeIf, [396](#)
 - OSnLNodeLn, [398](#)
 - OSnLNodeMatrixDeterminant, [400](#)
 - OSnLNodeMatrixToScalar, [402](#)
 - OSnLNodeMatrixTrace, [404](#)
 - OSnLNodeMax, [406](#)
 - OSnLNodeMin, [408](#)
 - OSnLNodeMinus, [410](#)
 - OSnLNodeNegate, [412](#)
 - OSnLNodeNumber, [414](#)
 - OSnLNodePI, [417](#)
 - OSnLNodePlus, [419](#)
 - OSnLNodePower, [420](#)
 - OSnLNodeProduct, [422](#)
 - OSnLNodeSin, [424](#)
 - OSnLNodeSqrt, [426](#)
 - OSnLNodeSquare, [428](#)
 - OSnLNodeSum, [430](#)
 - OSnLNodeTimes, [432](#)
 - OSnLNodeVariable, [435](#)
 - ScalarExpressionTree, [654](#)
- calculateFunctionValue
 - OSInstance, [324](#)
- calculateHessian
 - OSInstance, [328](#)
- calculateLagrangianHessian
 - OSInstance, [328](#)
- calculateObjectiveFunctionGradient
 - OSInstance, [327](#), [328](#)
- categoryAttribute
 - OSrLParserData, [582](#)
- cloneExprNode
 - ExprNode, [101](#)
 - OSnLMNodeDiagonalMatrixFromVector, [355](#)
 - OSnLMNodeIdentityMatrix, [356](#)
 - OSnLMNodeMatrixCon, [358](#)
 - OSnLMNodeMatrixDiagonal, [359](#)
 - OSnLMNodeMatrixDotTimes, [360](#)
 - OSnLMNodeMatrixInverse, [360](#)
 - OSnLMNodeMatrixLowerTriangle, [362](#)
 - OSnLMNodeMatrixMinus, [363](#)
 - OSnLMNodeMatrixNegate, [364](#)
 - OSnLMNodeMatrixObj, [365](#)
 - OSnLMNodeMatrixPlus, [366](#)
 - OSnLMNodeMatrixProduct, [367](#)
 - OSnLMNodeMatrixReference, [369](#)
 - OSnLMNodeMatrixScalarTimes, [370](#)
 - OSnLMNodeMatrixSubmatrixAt, [371](#)
 - OSnLMNodeMatrixSum, [372](#)
 - OSnLMNodeMatrixTimes, [373](#)
 - OSnLMNodeMatrixTranspose, [374](#)
 - OSnLMNodeMatrixUpperTriangle, [375](#)
 - OSnLMNodeMatrixVar, [376](#)

- OSnLNodeAbs, 383
- OSnLNodeAllDiff, 385
- OSnLNodeCos, 387
- OSnLNodeDivide, 389
- OSnLNodeE, 391
- OSnLNodeErf, 393
- OSnLNodeExp, 395
- OSnLNodeIf, 397
- OSnLNodeLn, 399
- OSnLNodeMatrixDeterminant, 401
- OSnLNodeMatrixToScalar, 402
- OSnLNodeMatrixTrace, 404
- OSnLNodeMax, 406
- OSnLNodeMin, 408
- OSnLNodeMinus, 410
- OSnLNodeNegate, 412
- OSnLNodeNumber, 414
- OSnLNodePI, 417
- OSnLNodePlus, 419
- OSnLNodePower, 421
- OSnLNodeProduct, 423
- OSnLNodeSin, 425
- OSnLNodeSqrt, 427
- OSnLNodeSquare, 429
- OSnLNodeSum, 431
- OSnLNodeTimes, 433
- OSnLNodeVariable, 435
- cloneMatrixNode
 - BaseMatrix, 44
 - ConReferenceMatrixElements, 65
 - ConstantMatrixElements, 70
 - GeneralMatrixElements, 108
 - LinearMatrixElements, 182
 - MatrixBlock, 190
 - MatrixBlocks, 195
 - MatrixNode, 206
 - MatrixTransformation, 216
 - MixedRowReferenceMatrixElements, 236
 - OSMatrix, 340
 - OSMatrixWithMatrixConIdx, 341
 - OSMatrixWithMatrixObjIdx, 343
 - OSMatrixWithMatrixVarIdx, 344
 - ObjReferenceMatrixElements, 251
 - VarReferenceMatrixElements, 700
- CoinSolver, 54
 - buildSolverInstance, 55
 - dataEchoCheck, 56
 - getCoinSolverType, 56
 - setCoinPackedMatrix, 55
 - setSolverOptions, 55
 - solve, 55
- CompletelyPositiveMatricesCone, 56
 - deepCopyFrom, 58
 - getConeInXML, 57
 - getConeName, 57
 - setRandom, 57
- conReference
 - ConReferenceMatrixElement, 63
- ConReferenceMatrixElement, 62
 - conReference, 63
 - deepCopyFrom, 63
 - setRandom, 62
 - valueType, 63
- ConReferenceMatrixElements, 63
 - alignsOnBlockBoundary, 65
 - cloneMatrixNode, 65
 - deepCopyFrom, 66
 - getMatrixNodeInXML, 65
 - getMatrixType, 64
 - getNodeName, 64
 - getNodeType, 64
 - setRandom, 65
- ConReferenceMatrixValues, 66
 - deepCopyFrom, 67
 - setRandom, 67
- Cone, 58
 - deepCopyFrom, 60
 - getConeInXML, 59
 - getConeName, 59
 - numberOfOtherIndexes, 60
 - setRandom, 59
- Cones, 60
 - deepCopyFrom, 61
 - setRandom, 61
- ConstantMatrixElements, 67
 - alignsOnBlockBoundary, 68
 - cloneMatrixNode, 70
 - deepCopyFrom, 70
 - getMatrixNodeInXML, 68
 - getMatrixType, 68
 - getNodeName, 68
 - getNodeType, 68
 - setRandom, 70
- ConstantMatrixValues, 70
 - deepCopyFrom, 72
 - isEqual, 71
 - setRandom, 71
- Constraint, 72
- ConstraintOption, 73
 - addOther, 75
 - deepCopyFrom, 74
 - setOther, 74
 - setRandom, 74
- ConstraintSolution, 76
 - setRandom, 77
- Constraints, 75
- constructADTape
 - OSnLNode, 379

- OSnLNodeAbs, [383](#)
- OSnLNodeAllDiff, [385](#)
- OSnLNodeCos, [387](#)
- OSnLNodeDivide, [389](#)
- OSnLNodeE, [391](#)
- OSnLNodeErf, [393](#)
- OSnLNodeExp, [395](#)
- OSnLNodeIf, [397](#)
- OSnLNodeLn, [399](#)
- OSnLNodeMatrixDeterminant, [400](#)
- OSnLNodeMatrixToScalar, [402](#)
- OSnLNodeMatrixTrace, [404](#)
- OSnLNodeMax, [406](#)
- OSnLNodeMin, [408](#)
- OSnLNodeMinus, [410](#)
- OSnLNodeNegate, [412](#)
- OSnLNodeNumber, [415](#)
- OSnLNodePI, [417](#)
- OSnLNodePlus, [419](#)
- OSnLNodePower, [421](#)
- OSnLNodeProduct, [423](#)
- OSnLNodeSin, [425](#)
- OSnLNodeSqrt, [427](#)
- OSnLNodeSquare, [429](#)
- OSnLNodeSum, [431](#)
- OSnLNodeTimes, [433](#)
- OSnLNodeVariable, [436](#)
- ContactOption, [77](#)
 - deepCopyFrom, [78](#)
 - setRandom, [78](#)
- convertLinearConstraintCoefficientMatrixToTheOther←
 - Major
 - MathUtil, [186](#)
- convertSolverNameToUpperCase
 - OSCommandLine, [267](#)
- convertToOtherMajor
 - MatrixType, [220](#)
- CopositiveMatricesCone, [78](#)
 - deepCopyFrom, [80](#)
 - getConeInXML, [79](#)
 - getConeName, [79](#)
 - setRandom, [79](#)
- copyLinearConstraintCoefficients
 - OSInstance, [315](#)
- copyNodeAndDescendants
 - OSnLMNode, [354](#)
 - OSnLMNodeMatrixCon, [358](#)
 - OSnLMNodeMatrixLowerTriangle, [362](#)
 - OSnLMNodeMatrixObj, [365](#)
 - OSnLMNodeMatrixReference, [369](#)
 - OSnLMNodeMatrixUpperTriangle, [375](#)
 - OSnLMNodeMatrixVar, [377](#)
 - OSnLNode, [381](#)
 - OSnLNodeNumber, [415](#)
 - OSnLNodeVariable, [435](#)
- CouenneSolver, [80](#)
 - setSolverOptions, [81](#)
- createConstructorTreeFromPrefix
 - OSMatrix, [337](#)
- createExpressionTreeFromPostfix
 - OSnLMNode, [354](#)
 - OSnLNode, [380](#)
- createExpressionTreeFromPrefix
 - OSnLMNode, [353](#)
 - OSnLNode, [379](#)
- createFormDataUpload
 - WSUtil, [708](#)
- createOSADFun
 - OSInstance, [330](#)
- createOSInstance
 - OSgams2osil, [271](#)
 - OSmps2osil, [348](#)
- createOSObjects
 - OSmps2OS, [346](#)
 - OSnl2OS, [350](#)
- createSOAPMessage
 - WSUtil, [707](#)
- CsdpSolver, [85](#)
 - buildSolverInstance, [86](#)
 - setSolverOptions, [86](#)
- dataEchoCheck
 - CoinSolver, [56](#)
- deSOAPify
 - WSUtil, [707](#)
- decodeb64
 - Base64, [42](#)
- deepCopyFrom
 - BasisStatus, [46](#)
 - BranchingWeight, [54](#)
 - CPUNumber, [83](#)
 - CPUSpeed, [84](#)
 - CompletelyPositiveMatricesCone, [58](#)
 - ConReferenceMatrixElement, [63](#)
 - ConReferenceMatrixElements, [66](#)
 - ConReferenceMatrixValues, [67](#)
 - Cone, [60](#)
 - Cones, [61](#)
 - ConstantMatrixElements, [70](#)
 - ConstantMatrixValues, [72](#)
 - ConstraintOption, [74](#)
 - ContactOption, [78](#)
 - CopositiveMatricesCone, [80](#)
 - DirectoriesAndFiles, [89](#)
 - DualCone, [91](#)
 - GeneralFileHeader, [105](#)
 - GeneralMatrixElements, [108](#)
 - GeneralMatrixValues, [109](#)

- GeneralOption, [111](#)
- InitBasStatus, [121](#)
- InitConValue, [125](#)
- InitConstraintValues, [122](#)
- InitDualVarValue, [129](#)
- InitDualVariableValues, [127](#)
- InitObjBound, [133](#)
- InitObjValue, [141](#)
- InitObjectiveBounds, [135](#)
- InitObjectiveValues, [139](#)
- InitVarValue, [149](#)
- InitVarValueString, [151](#)
- InitVariableValues, [143](#)
- InitVariableValuesString, [147](#)
- InitialBasisStatus, [131](#)
- InstanceLocationOption, [154](#)
- IntVector, [162](#)
- IntegerVariableBranchingWeights, [156](#)
- IntersectionCone, [159](#)
- JobDependencies, [167](#)
- JobOption, [171](#)
- LinearMatrixElement, [180](#)
- LinearMatrixElementTerm, [183](#)
- LinearMatrixElements, [182](#)
- LinearMatrixValues, [185](#)
- Matrices, [188](#)
- MatrixBlock, [192](#)
- MatrixBlocks, [195](#)
- MatrixElementValues, [200](#)
- MatrixNode, [208](#)
- MatrixProgramming, [213](#)
- MatrixProgrammingSolution, [214](#)
- MatrixTransformation, [217](#)
- MatrixType, [225](#)
- MixedRowReferenceMatrixElements, [236](#)
- NonnegativeCone, [240](#)
- NonpositiveCone, [242](#)
- OSMatrix, [340](#)
- OSMatrixWithMatrixConIdx, [342](#)
- OSMatrixWithMatrixObjIdx, [343](#)
- OSMatrixWithMatrixVarIdx, [345](#)
- OSOption, [452](#)
- ObjReferenceMatrixElements, [251](#)
- ObjReferenceMatrixValues, [252](#)
- ObjectiveOption, [245](#)
- OptimizationOption, [256](#)
- OrthantCone, [264](#)
- OtherConOption, [596](#)
- OtherConstraintOption, [599](#)
- OtherObjOption, [608](#)
- OtherObjectiveOption, [605](#)
- OtherOption, [611](#)
- OtherOptionOrResultEnumeration, [614](#)
- OtherOptions, [616](#)
- OtherVarOption, [629](#)
- OtherVariableOption, [624](#)
- PathPair, [633](#)
- PathPairs, [636](#)
- PolarCone, [638](#)
- PolyhedralCone, [640](#)
- Processes, [643](#)
- ProductCone, [645](#)
- QuadraticCone, [648](#)
- RotatedQuadraticCone, [652](#)
- SOSVariableBranchingWeights, [666](#)
- SOSWeights, [669](#)
- SemidefiniteCone, [656](#)
- ServiceOption, [658](#)
- SolverOption, [661](#)
- SolverOptions, [663](#)
- StorageCapacity, [676](#)
- SystemOption, [678](#)
- TimeSpan, [689](#)
- VarReferenceMatrixElements, [700](#)
- VarReferenceMatrixValues, [702](#)
- VariableOption, [693](#)
- DefaultSolver, [86](#)
 - sSolverName, [87](#)
- DeleteChannel
 - OSOutput, [481](#)
- DirectoriesAndFiles, [88](#)
 - addPath, [89](#)
 - deepCopyFrom, [89](#)
 - setPath, [89](#)
 - setRandom, [89](#)
- disassembleMatrix
 - MatrixType, [224](#)
- display
 - ExpandedMatrixBlocks, [97](#)
 - GeneralSparseMatrix, [115](#)
 - SparseMatrix, [674](#)
- DoubleVector, [89](#)
- DualCone, [90](#)
 - deepCopyFrom, [91](#)
 - getConeName, [91](#)
 - numberOfOtherIndexes, [91](#)
 - setRandom, [91](#)
- DualVarValue, [93](#)
 - setRandom, [94](#)
- DualVariableValues, [92](#)
 - setRandom, [93](#)
- ENUM_BASIS_STATUS
 - OSParameters.h, [736](#)
- ENUM_OUTPUT_AREA
 - OSParameters.h, [736](#)
- ENUM_OUTPUT_LEVEL
 - OSParameters.h, [736](#)

- encodeb64
 - Base64, [42](#)
- enumeration
 - OtherMatrixVariableResult, [603](#)
- ErrorClass, [94](#)
 - ErrorClass, [95](#)
- expandElements
 - MatrixBlock, [190](#)
 - MatrixType, [220](#)
 - OSMatrix, [338](#)
- ExpandedMatrixBlocks, [95](#)
 - blockColumns, [98](#)
 - blockRows, [97](#)
 - blocks, [98](#)
 - display, [97](#)
 - ExpandedMatrixBlocks, [96](#)
 - getBlock, [97](#)
 - isBlockDiagonal, [97](#)
 - isRowMajor, [97](#)
- ExprNode, [98](#)
 - cloneExprNode, [101](#)
 - getNonlinearExpressionInXML, [100](#)
 - getPostfixFromExpressionTree, [101](#)
 - getPrefixFromExpressionTree, [100](#)
 - getTokenName, [99](#)
 - getTokenNumber, [99](#)
 - inodeType, [102](#)
 - postOrderOSnLNodeTraversal, [101](#)
 - preOrderOSnLNodeTraversal, [100](#)
- extendIntVector
 - IntVector, [163](#)
- extractBlock
 - MatrixType, [222](#)
- fileUpload
 - OSSolverAgent, [594](#)
- FileUtil, [102](#)
 - getFileAsChar, [103](#)
 - getFileAsString, [103](#)
 - writeFileFromChar, [104](#)
 - writeFileFromString, [103](#)
- finalize_solution
 - BonminProblem, [51](#)
- FindChannel
 - OSOutput, [481](#)
- firstAxisDirection
 - RotatedQuadraticCone, [652](#)
- format_os_dtoa
 - MathUtil, [187](#)
- forwardAD
 - OSInstance, [330](#)
- GeneralFileHeader, [104](#)
 - deepCopyFrom, [105](#)
 - getHeaderItem, [105](#)
 - setHeader, [106](#)
 - setRandom, [105](#)
- GeneralMatrixElements, [106](#)
 - alignsOnBlockBoundary, [107](#)
 - cloneMatrixNode, [108](#)
 - deepCopyFrom, [108](#)
 - getMatrixNodeInXML, [107](#)
 - getMatrixType, [107](#)
 - getNodeName, [107](#)
 - getNodeType, [107](#)
 - setRandom, [108](#)
- GeneralMatrixValues, [108](#)
 - deepCopyFrom, [109](#)
 - setRandom, [109](#)
- GeneralOption, [110](#)
 - deepCopyFrom, [111](#)
 - setRandom, [111](#)
- GeneralResult, [112](#)
 - setRandom, [113](#)
- GeneralSparseMatrix, [113](#)
 - display, [115](#)
 - GeneralSparseMatrix, [114](#)
 - index, [115](#)
 - isDiagonal, [115](#)
 - isRowMajor, [115](#)
 - symmetry, [115](#)
 - vType, [115](#)
 - value, [115](#)
- GeneralStatus, [116](#)
 - setRandom, [117](#)
- GeneralSubstatus, [117](#)
 - setRandom, [118](#)
- generateLindoModel
 - LindoSolver, [177](#)
- get_bounds_info
 - BonminProblem, [51](#)
- get_constraints_linearity
 - BonminProblem, [50](#)
- get_nlp_info
 - BonminProblem, [50](#)
- get_variables_linearity
 - BonminProblem, [50](#)
- get_variables_types
 - BonminProblem, [50](#)
- getADSparsityHessian
 - OSInstance, [331](#)
- getASL
 - OSnI2OS, [350](#)
- getAllMatrixExpressionTrees
 - OSInstance, [308](#)
- getAllMatrixExpressionTreesMod
 - OSInstance, [308](#)
- getAllNonlinearExpressionTrees
 - OSInstance, [301](#)

- getAllNonlinearExpressionTreesMod
 - OSInstance, [301](#)
- getAllNonlinearVariablesIndexMap
 - OSInstance, [329](#)
- getAllOtherConstraintOptions
 - OSOption, [467](#)
- getAllOtherObjectiveOptions
 - OSOption, [463](#)
- getAllOtherOptions
 - OSOption, [455](#)
- getAllOtherVariableOptions
 - OSOption, [461](#)
- getAllSolverOptions
 - OSOption, [468](#)
- getAnOtherVariableResultNumberOfVar
 - OSResult, [507](#)
- getBasisDense
 - BasisStatus, [48](#)
- getBasisInformationDense
 - OSResult, [506](#)
- getBasisStatusEl
 - OSResult, [506](#)
- getBasisStatusNumberOfEl
 - OSResult, [506](#)
- getBlock
 - ExpandedMatrixBlocks, [97](#)
- getBlocks
 - MatrixType, [222](#)
- getCoinSolverType
 - CoinSolver, [56](#)
- getColumnPartition
 - MatrixType, [220](#)
- getColumnPartitionSize
 - MatrixType, [220](#)
- getConeInXML
 - CompletelyPositiveMatricesCone, [57](#)
 - Cone, [59](#)
 - CopositiveMatricesCone, [79](#)
 - IntersectionCone, [159](#)
 - NonnegativeCone, [239](#)
 - NonpositiveCone, [241](#)
 - OrthantCone, [263](#)
 - PolyhedralCone, [639](#)
 - ProductCone, [644](#)
 - QuadraticCone, [647](#)
 - RotatedQuadraticCone, [651](#)
 - SemidefiniteCone, [656](#)
- getConeName
 - CompletelyPositiveMatricesCone, [57](#)
 - Cone, [59](#)
 - CopositiveMatricesCone, [79](#)
 - DualCone, [91](#)
 - IntersectionCone, [159](#)
 - NonnegativeCone, [239](#)
 - NonpositiveCone, [241](#)
 - OrthantCone, [263](#)
 - PolarCone, [638](#)
 - PolyhedralCone, [639](#)
 - ProductCone, [644](#)
 - QuadraticCone, [647](#)
 - RotatedQuadraticCone, [651](#)
 - SemidefiniteCone, [656](#)
- getConstraintConstants
 - OSInstance, [297](#)
- getConstraintLowerBounds
 - OSInstance, [296](#)
- getConstraintNames
 - OSInstance, [296](#)
- getConstraintNumber
 - OSInstance, [296](#)
 - OSResult, [504](#)
- getConstraintTypes
 - OSInstance, [297](#)
- getConstraintUpperBounds
 - OSInstance, [296](#)
- getDenseObjectiveCoefficients
 - OSInstance, [296](#)
- getDirectoriesToDelete
 - OSOption, [457](#)
- getDirectoriesToMake
 - OSOption, [456](#)
- getEl
 - BasisStatus, [46](#)
 - IntVector, [163](#)
- getFileAsChar
 - FileUtil, [103](#)
- getFileAsString
 - FileUtil, [103](#)
- getFilesToDelete
 - OSOption, [457](#)
- getFilesToMake
 - OSOption, [456](#)
- getFirstOrderResults
 - OSInstance, [331](#)
- getGeneralMessage
 - OSResult, [502](#)
- getGeneralStatus
 - OSResult, [500](#)
- getGeneralStatusDescription
 - OSResult, [500](#)
- getGeneralStatusType
 - OSResult, [500](#)
- getGeneralSubstatusDescription
 - OSResult, [502](#)
- getGeneralSubstatusName
 - OSResult, [500](#)
- getHeaderItem
 - GeneralFileHeader, [105](#)

- getInitBasisStatusDense
 - OSOption, [459](#)
- getInitBasisStatusSparse
 - OSOption, [458](#)
- getInitConValuesDense
 - OSOption, [464](#)
- getInitConValuesSparse
 - OSOption, [464](#)
- getInitDualVarLowerBoundsDense
 - OSOption, [464](#)
- getInitDualVarUpperBoundsDense
 - OSOption, [466](#)
- getInitDualVarValuesSparse
 - OSOption, [464](#)
- getInitObjBoundsSparse
 - OSOption, [462](#)
- getInitObjLowerBoundsDense
 - OSOption, [462](#)
- getInitObjUpperBoundsDense
 - OSOption, [462](#)
- getInitObjValuesDense
 - OSOption, [461](#)
- getInitObjValuesSparse
 - OSOption, [461](#)
- getInitVarValuesDense
 - OSOption, [457](#), [458](#)
- getInitVarValuesSparse
 - OSOption, [457](#)
- getInitVarValuesStringDense
 - OSOption, [458](#)
- getInitVarValuesStringSparse
 - OSOption, [458](#)
- getInitialBasisElements
 - OSOption, [459](#)
- getInputDirectoriesToMove
 - OSOption, [456](#)
- getInputFilesToMove
 - OSOption, [456](#)
- getInstanceCreator
 - OSInstance, [291](#)
- getInstanceDescription
 - OSInstance, [291](#)
- getInstanceLicence
 - OSInstance, [291](#)
- getInstanceName
 - OSInstance, [290](#)
 - OSResult, [502](#)
- getInstanceSource
 - OSInstance, [290](#)
- getIntVector
 - BasisStatus, [48](#)
- getIntegerVariableBranchingWeightsDense
 - OSOption, [460](#)
- getIntegerVariableBranchingWeightsSparse
 - OSOption, [460](#)
- getIterateResults
 - OSInstance, [331](#)
- getJacobianSparsityPattern
 - OSInstance, [330](#)
- getJobDependencies
 - OSOption, [455](#)
- getJobID
 - OSOption, [452](#)
 - OSResult, [502](#)
 - OSSolverAgent, [593](#)
 - OSHL, [275](#)
- getLagrangianExpTree
 - OSInstance, [329](#)
- getLagrangianHessianSparsityPattern
 - OSInstance, [329](#)
- getLinearConstraintCoefficientMajor
 - OSInstance, [297](#)
- getLinearConstraintCoefficientNumber
 - OSInstance, [297](#)
- getLinearConstraintCoefficientsInColumnMajor
 - OSInstance, [298](#)
- getLinearConstraintCoefficientsInRowMajor
 - OSInstance, [298](#)
- getMatrix
 - OSInstance, [304](#)
- getMatrixBlockInColumnMajorForm
 - OSInstance, [304](#)
- getMatrixCoefficientsInColumnMajor
 - OSInstance, [304](#)
- getMatrixCoefficientsInRowMajor
 - OSInstance, [304](#)
- getMatrixExpressionTree
 - OSInstance, [307](#)
- getMatrixExpressionTreeInInfix
 - OSInstance, [307](#)
- getMatrixExpressionTreeInPostfix
 - OSInstance, [307](#)
- getMatrixExpressionTreeInPrefix
 - OSInstance, [307](#)
- getMatrixExpressionTreeIndexes
 - OSInstance, [308](#)
- getMatrixExpressionTreeModInPostfix
 - OSInstance, [307](#)
- getMatrixExpressions
 - OSInstance, [306](#)
- getMatrixName
 - OSInstance, [303](#)
- getMatrixNodeInXML
 - BaseMatrix, [44](#)
 - ConReferenceMatrixElements, [65](#)
 - ConstantMatrixElements, [68](#)
 - GeneralMatrixElements, [107](#)
 - LinearMatrixElements, [181](#)

- MatrixBlock, [189](#)
- MatrixBlocks, [193](#)
- MatrixNode, [205](#)
- MatrixTransformation, [216](#)
- MixedRowReferenceMatrixElements, [235](#)
- OSMatrix, [340](#)
- OSMatrixWithMatrixConIdx, [341](#)
- OSMatrixWithMatrixObjIdx, [343](#)
- OSMatrixWithMatrixVarIdx, [344](#)
- ObjReferenceMatrixElements, [250](#)
- VarReferenceMatrixElements, [699](#)
- getMatrixNumber
 - OSInstance, [302](#)
- getMatrixSymmetry
 - OSInstance, [302](#)
- getMatrixType
 - BaseMatrix, [44](#)
 - ConReferenceMatrixElements, [64](#)
 - ConstantMatrixElements, [68](#)
 - GeneralMatrixElements, [107](#)
 - LinearMatrixElements, [181](#)
 - MatrixBlock, [189](#)
 - MatrixBlocks, [193](#)
 - MatrixNode, [205](#)
 - MatrixTransformation, [215](#)
 - MixedRowReferenceMatrixElements, [235](#)
 - OSInstance, [302](#)
 - OSMatrix, [338](#)
 - ObjReferenceMatrixElements, [250](#)
 - VarReferenceMatrixElements, [699](#)
- getNodeName
 - BaseMatrix, [43](#)
 - ConReferenceMatrixElements, [64](#)
 - ConstantMatrixElements, [68](#)
 - GeneralMatrixElements, [107](#)
 - LinearMatrixElements, [181](#)
 - MatrixBlock, [189](#)
 - MatrixBlocks, [193](#)
 - MatrixNode, [205](#)
 - MatrixTransformation, [215](#)
 - MixedRowReferenceMatrixElements, [235](#)
 - OSMatrix, [338](#)
 - ObjReferenceMatrixElements, [250](#)
 - VarReferenceMatrixElements, [699](#)
- getNodeType
 - BaseMatrix, [43](#)
 - ConReferenceMatrixElements, [64](#)
 - ConstantMatrixElements, [68](#)
 - GeneralMatrixElements, [107](#)
 - LinearMatrixElements, [181](#)
 - MatrixBlock, [189](#)
 - MatrixBlocks, [193](#)
 - MatrixNode, [205](#)
 - MatrixTransformation, [215](#)
 - MixedRowReferenceMatrixElements, [235](#)
 - OSMatrix, [338](#)
 - ObjReferenceMatrixElements, [250](#)
 - VarReferenceMatrixElements, [699](#)
- getNonlinearExpressionInXML
 - ExprNode, [100](#)
 - OSnLMNodeMatrixCon, [357](#)
 - OSnLMNodeMatrixLowerTriangle, [362](#)
 - OSnLMNodeMatrixObj, [365](#)
 - OSnLMNodeMatrixReference, [369](#)
 - OSnLMNodeMatrixUpperTriangle, [375](#)
 - OSnLMNodeMatrixVar, [376](#)
 - OSnLNodeE, [390](#)
 - OSnLNodeNumber, [414](#)
 - OSnLNodePI, [417](#)
 - OSnLNodeVariable, [435](#)
- getNonlinearExpressionTree
 - OSInstance, [299](#)
- getNonlinearExpressionTreeInInfix
 - OSInstance, [300](#)
- getNonlinearExpressionTreeInPostfix
 - OSInstance, [300](#)
- getNonlinearExpressionTreeInPrefix
 - OSInstance, [300](#)
- getNonlinearExpressionTreeIndexes
 - OSInstance, [301](#)
- getNonlinearExpressionTreeMod
 - OSInstance, [299](#)
- getNonlinearExpressionTreeModInPostfix
 - OSInstance, [300](#)
- getNonlinearExpressionTreeModInPrefix
 - OSInstance, [300](#)
- getNonlinearExpressionTreeModIndexes
 - OSInstance, [301](#)
- getNonlinearExpressions
 - OSInstance, [299](#)
- getNumberOfBinaryVariables
 - OSInstance, [292](#)
- getNumberOfColumnsForMatrix
 - OSInstance, [303](#)
- getNumberOfEI
 - BasisStatus, [46](#)
- getNumberOfGeneralSubstatuses
 - OSResult, [500](#)
- getNumberOfInitConValues
 - OSOption, [454](#)
- getNumberOfInitDualVarValues
 - OSOption, [454](#)
- getNumberOfInitObjBounds
 - OSOption, [453](#)
- getNumberOfInitObjValues
 - OSOption, [453](#)
- getNumberOfInitVarValues
 - OSOption, [452](#)

- getNumberOfInitVarValuesString
 - OSOption, [452](#)
- getNumberOfInitialBasisElements
 - OSOption, [459](#)
- getNumberOfIntegerVariableBranchingWeights
 - OSOption, [452](#)
- getNumberOfIntegerVariables
 - OSInstance, [292](#)
- getNumberOfMatrixConstraints
 - OSInstance, [306](#)
- getNumberOfMatrixExpressionTreeIndexes
 - OSInstance, [308](#)
- getNumberOfMatrixExpressions
 - OSInstance, [306](#)
- getNumberOfMatrixObjectives
 - OSInstance, [306](#)
- getNumberOfMatrixVariables
 - OSInstance, [306](#)
- getNumberOfNonlinearConstraints
 - OSInstance, [301](#)
- getNumberOfNonlinearExpressionTreeIndexes
 - OSInstance, [301](#)
- getNumberOfNonlinearExpressionTreeModIndexes
 - OSInstance, [302](#)
- getNumberOfNonlinearExpressions
 - OSInstance, [299](#)
- getNumberOfNonlinearObjectives
 - OSInstance, [301](#)
- getNumberOfOtherConstraintOptions
 - OSOption, [454](#)
- getNumberOfOtherGeneralResults
 - OSResult, [503](#)
- getNumberOfOtherObjectiveOptions
 - OSOption, [453](#)
- getNumberOfOtherVariableOptions
 - OSOption, [453](#)
- getNumberOfOtherVariableResults
 - OSResult, [507](#)
- getNumberOfQuadraticRowIndexes
 - OSInstance, [299](#)
- getNumberOfQuadraticTerms
 - OSInstance, [298](#)
- getNumberOfRowsForMatrix
 - OSInstance, [303](#)
- getNumberOfSOS
 - OSOption, [453](#)
- getNumberOfSOSVarBranchingWeights
 - OSOption, [453](#)
- getNumberOfSemiContinuousVariables
 - OSInstance, [293](#)
- getNumberOfSemiIntegerVariables
 - OSInstance, [293](#)
- getNumberOfSolverOptions
 - OSOption, [454](#)
- getNumberOfStringVariables
 - OSInstance, [293](#)
- getNumberOfValuesForMatrix
 - OSInstance, [303](#)
- getOSInstance
 - OSgams2osil, [272](#)
- getOSxL
 - WSUtil, [708](#)
- getObjectiveCoefficientNumbers
 - OSInstance, [294](#)
- getObjectiveCoefficients
 - OSInstance, [295](#)
- getObjectiveConstants
 - OSInstance, [295](#)
- getObjectiveInitialBasisStatusDense
 - OSOption, [463](#)
- getObjectiveMaxOrMins
 - OSInstance, [294](#)
- getObjectiveNames
 - OSInstance, [294](#)
- getObjectiveNumber
 - OSInstance, [294](#)
 - OSResult, [504](#)
- getObjectiveWeights
 - OSInstance, [295](#)
- getOptimalDualVariableValues
 - OSResult, [511](#)
- getOptimalObjValue
 - OSResult, [509](#)
- getOptimalPrimalVariableValues
 - OSResult, [506](#)
- getOtherConstraintOption
 - OSOption, [467](#)
- getOtherConstraintOptions
 - OSOption, [466](#)
- getOtherConstraintResultArrayDense
 - OSResult, [514](#)
- getOtherConstraintResultArrayType
 - OSResult, [513](#)
- getOtherConstraintResultEnumerationDescription
 - OSResult, [513](#)
- getOtherConstraintResultEnumerationEI
 - OSResult, [514](#)
- getOtherConstraintResultEnumerationNumberOfEI
 - OSResult, [513](#)
- getOtherConstraintResultEnumerationValue
 - OSResult, [513](#)
- getOtherGeneralOptions
 - OSOption, [454](#)
- getOtherGeneralResultName
 - OSResult, [503](#)
- getOtherJobOptions
 - OSOption, [455](#)
- getOtherObjectiveOption

- OSOption, [463](#)
- getOtherObjectiveOptions
 - OSOption, [463](#)
- getOtherObjectiveResultArrayDense
 - OSResult, [511](#)
- getOtherObjectiveResultArrayType
 - OSResult, [509](#)
- getOtherObjectiveResultEnumerationDescription
 - OSResult, [509](#)
- getOtherObjectiveResultEnumerationEI
 - OSResult, [511](#)
- getOtherObjectiveResultEnumerationNumberOfEI
 - OSResult, [511](#)
- getOtherObjectiveResultEnumerationValue
 - OSResult, [509](#)
- getOtherOptions
 - OSOption, [455](#)
- getOtherServiceOptions
 - OSOption, [455](#)
- getOtherSystemOptions
 - OSOption, [454](#)
- getOtherVariableOption
 - OSOption, [461](#)
- getOtherVariableOptions
 - OSOption, [460](#)
- getOtherVariableResultArrayDense
 - OSResult, [508](#)
- getOtherVariableResultArrayType
 - OSResult, [507](#)
- getOtherVariableResultEnumerationDescription
 - OSResult, [508](#)
- getOtherVariableResultEnumerationEI
 - OSResult, [508](#)
- getOtherVariableResultEnumerationNumberOfEI
 - OSResult, [508](#)
- getOtherVariableResultEnumerationValue
 - OSResult, [507](#)
- getOutputDirectoriesToMove
 - OSOption, [456](#)
- getOutputFilesToMove
 - OSOption, [457](#)
- getPostfixFromExpressionTree
 - ExprNode, [101](#)
 - MatrixExpressionTree, [203](#)
 - OSnLMNode, [354](#)
 - OSnLNode, [380](#)
 - ScalarExpressionTree, [654](#)
- getPostfixFromNodeTree
 - MatrixNode, [206](#)
- getPrefixFromExpressionTree
 - ExprNode, [100](#)
 - MatrixExpressionTree, [203](#)
 - OSnLMNode, [353](#)
 - OSnLNode, [380](#)
- ScalarExpressionTree, [654](#)
- getPrefixFromNodeTree
 - MatrixNode, [206](#)
- getProcessesToKill
 - OSOption, [457](#)
- getQuadraticRowIndex
 - OSInstance, [299](#)
- getQuadraticTerms
 - OSInstance, [298](#)
- GetRawPtr
 - OSSmartPtr, [590](#)
- getRequiredDirectories
 - OSOption, [455](#)
- getRequiredFiles
 - OSOption, [456](#)
- getRowPartition
 - MatrixType, [219](#)
- getRowPartitionSize
 - MatrixType, [219](#)
- getSOSVariableBranchingWeightsSparse
 - OSOption, [460](#)
- getSecondOrderResults
 - OSInstance, [333](#)
- getServiceName
 - OSResult, [502](#)
- getServiceURI
 - OSResult, [502](#)
- getSlackVariableInitialBasisStatusDense
 - OSOption, [466](#)
- getSolutionMessage
 - OSResult, [505](#)
- getSolutionNumber
 - OSResult, [504](#)
- getSolutionStatus
 - OSResult, [504](#)
- getSolutionStatusDescription
 - OSResult, [505](#)
- getSolutionStatusType
 - OSResult, [505](#)
- getSolutionWeightedObjectives
 - OSResult, [505](#)
- getSolverInvoked
 - OSResult, [503](#)
- getSolverOptions
 - OSOption, [467](#)
- getSparseJacobianFromColumnMajor
 - OSInstance, [329](#)
- getSparseJacobianFromRowMajor
 - OSInstance, [329](#)
- getTimeDomainFormat
 - OSInstance, [308](#)
- getTimeDomainIntervalHorizon
 - OSInstance, [310](#)
- getTimeDomainIntervalStart

- OSInstance, 310
- getTimeDomainStageConList
 - OSInstance, 309
- getTimeDomainStageNames
 - OSInstance, 309
- getTimeDomainStageNumber
 - OSInstance, 308
- getTimeDomainStageNumberOfConstraints
 - OSInstance, 309
- getTimeDomainStageNumberOfObjectives
 - OSInstance, 309
- getTimeDomainStageNumberOfVariables
 - OSInstance, 309
- getTimeDomainStageObjList
 - OSInstance, 310
- getTimeDomainStageVarList
 - OSInstance, 309
- getTimeNumber
 - OSResult, 503
- getTimeStamp
 - OSResult, 503
- getTimeValue
 - OSResult, 503
- getTokenName
 - ExprNode, 99
 - OSnLMNodeDiagonalMatrixFromVector, 355
 - OSnLMNodeIdentityMatrix, 356
 - OSnLMNodeMatrixCon, 357
 - OSnLMNodeMatrixDiagonal, 359
 - OSnLMNodeMatrixDotTimes, 359
 - OSnLMNodeMatrixInverse, 360
 - OSnLMNodeMatrixLowerTriangle, 361
 - OSnLMNodeMatrixMinus, 363
 - OSnLMNodeMatrixNegate, 364
 - OSnLMNodeMatrixObj, 365
 - OSnLMNodeMatrixPlus, 366
 - OSnLMNodeMatrixProduct, 367
 - OSnLMNodeMatrixReference, 369
 - OSnLMNodeMatrixScalarTimes, 370
 - OSnLMNodeMatrixSubmatrixAt, 371
 - OSnLMNodeMatrixSum, 372
 - OSnLMNodeMatrixTimes, 373
 - OSnLMNodeMatrixTranspose, 373
 - OSnLMNodeMatrixUpperTriangle, 375
 - OSnLMNodeMatrixVar, 376
 - OSnLNodeAbs, 382
 - OSnLNodeAllDiff, 384
 - OSnLNodeCos, 386
 - OSnLNodeDivide, 388
 - OSnLNodeE, 390
 - OSnLNodeErf, 392
 - OSnLNodeExp, 394
 - OSnLNodeIf, 396
 - OSnLNodeLn, 398
 - OSnLNodeMatrixDeterminant, 400
 - OSnLNodeMatrixToScalar, 402
 - OSnLNodeMatrixTrace, 404
 - OSnLNodeMax, 406
 - OSnLNodeMin, 408
 - OSnLNodeMinus, 410
 - OSnLNodeNegate, 412
 - OSnLNodeNumber, 414
 - OSnLNodePI, 416
 - OSnLNodePlus, 419
 - OSnLNodePower, 420
 - OSnLNodeProduct, 422
 - OSnLNodeSin, 424
 - OSnLNodeSqrt, 426
 - OSnLNodeSquare, 428
 - OSnLNodeSum, 430
 - OSnLNodeTimes, 432
 - OSnLNodeVariable, 435
- getTokenNumber
 - ExprNode, 99
 - OSnLMNodeMatrixCon, 357
 - OSnLMNodeMatrixObj, 365
 - OSnLMNodeMatrixReference, 369
 - OSnLMNodeMatrixVar, 376
 - OSnLNodeE, 390
 - OSnLNodeNumber, 414
 - OSnLNodePI, 416
 - OSnLNodeVariable, 435
- getVariableIndexMap
 - OSnLNode, 378
 - OSnLNodeVariable, 434
- getVariableIndicesMap
 - ScalarExpressionTree, 654
- getVariableInitialBasisStatusDense
 - OSOption, 459
- getVariableLowerBounds
 - OSInstance, 293
- getVariableNames
 - OSInstance, 291
- getVariableNumber
 - OSInstance, 291
 - OSResult, 504
- getVariableTypes
 - OSInstance, 292
- getVariableUpperBounds
 - OSInstance, 293
- getZeroOrderResults
 - OSInstance, 331
- iNumberOfStartElements
 - LinearConstraintCoefficients, 179
- id
 - OSnLNodeNumber, 415
- index

- GeneralSparseMatrix, 115
- IndexStringPair, 118
- IndexValuePair, 119
- indexes
 - SparseIntVector, 671
 - SparseMatrix, 674
- InitBasStatus, 119
 - deepCopyFrom, 121
 - setRandom, 120
- InitConValue, 124
 - deepCopyFrom, 125
 - setRandom, 125
- InitConstraintValues, 121
 - addCon, 123
 - deepCopyFrom, 122
 - setCon, 123
 - setRandom, 122
- InitDualVarValue, 128
 - deepCopyFrom, 129
 - setRandom, 129
- InitDualVariableValues, 125
 - addCon, 128
 - deepCopyFrom, 127
 - setCon, 127
 - setRandom, 127
- initForAlgDiff
 - OSInstance, 333
- InitObjBound, 132
 - deepCopyFrom, 133
 - setRandom, 133
- initObjGradients
 - OSInstance, 333
- InitObjValue, 140
 - deepCopyFrom, 141
 - setRandom, 141
- InitObjectiveBounds, 134
 - addObj, 136
 - deepCopyFrom, 135
 - setObj, 135, 136
 - setRandom, 135
- InitObjectiveValues, 136
 - addObj, 139, 140
 - deepCopyFrom, 139
 - setObj, 139
 - setRandom, 138
- InitVarValue, 148
 - deepCopyFrom, 149
 - setRandom, 149
- InitVarValueString, 150
 - deepCopyFrom, 151
 - setRandom, 151
- InitVariableValues, 142
 - addVar, 144
 - deepCopyFrom, 143
 - setRandom, 143
 - setVar, 143, 144
- InitVariableValuesString, 144
 - addVar, 148
 - deepCopyFrom, 147
 - setRandom, 146
 - setVar, 147
- InitialBasisStatus, 130
 - addVar, 132
 - deepCopyFrom, 131
 - setRandom, 131
 - setVar, 131
- initializeNonLinearStructures
 - OSInstance, 324
- inodeType
 - ExprNode, 102
- InstanceData, 151
- InstanceLocationOption, 153
 - deepCopyFrom, 154
 - setRandom, 154
- IntVector, 161
 - deepCopyFrom, 162
 - extendIntVector, 163
 - getEI, 163
 - setIntVector, 162
 - setRandom, 162
- IntegerVariableBranchingWeights, 154
 - addVar, 156, 158
 - deepCopyFrom, 156
 - setRandom, 155
 - setVar, 156
- IntersectionCone, 158
 - deepCopyFrom, 159
 - getConeInXML, 159
 - getConeName, 159
 - numberOfOtherIndexes, 161
 - setRandom, 159
- Interval, 161
- lpoptProblem, 163
- lpoptSolver, 165
 - buildSolverInstance, 166
 - setSolverOptions, 166
- isBlockDiagonal
 - ExpandedMatrixBlocks, 97
- isColumnMajor
 - SparseMatrix, 674
- isDiagonal
 - GeneralSparseMatrix, 115
- IsEqual
 - ConstantMatrixValues, 71
 - MatrixElements, 199
- IsNull
 - OSSmartPtr, 590
- isRowMajor

- ExpandedMatrixBlocks, 97
- GeneralSparseMatrix, 115
- IsValid
 - OSSmartPtr, 590
- JobDependencies, 166
 - addJobID, 169
 - deepCopyFrom, 167
 - setJobID, 167
 - setRandom, 167
- jobID
 - OSmps2OS, 347
 - OSnl2OS, 351
- JobOption, 169
 - deepCopyFrom, 171
 - setRandom, 170
- JobResult, 171
 - setRandom, 172
 - usedMemory, 173
- kill
 - OSSolverAgent, 593
 - OShL, 275
- KnitroProblem, 173
- KnitroSolver, 173
 - setSolverOptions, 174
- knock
 - OSSolverAgent, 594
 - OShL, 277
- kounter
 - OSrLParserData, 582
- lb
 - Variable, 691
- lindoAPIErrorCheck
 - LindoSolver, 177
- LindoSolver, 175
 - addSlackVars, 177
 - generateLindoModel, 177
 - lindoAPIErrorCheck, 177
 - optimize, 176
 - processConstraints, 176
 - processNonlinearExpressions, 177
 - processQuadraticTerms, 177
 - processVariables, 176
 - setSolverOptions, 176
- LinearConstraintCoefficients, 178
 - iNumberOfStartElements, 179
- LinearMatrixElement, 179
 - deepCopyFrom, 180
 - setRandom, 179
- LinearMatrixElementTerm, 183
 - deepCopyFrom, 183
 - setRandom, 183
- LinearMatrixElements, 180
 - alignsOnBlockBoundary, 181
 - cloneMatrixNode, 182
 - deepCopyFrom, 182
 - getMatrixNodeInXML, 181
 - getMatrixType, 181
 - getNodeName, 181
 - getNodeType, 181
 - setRandom, 182
 - value, 182
- LinearMatrixValues, 184
 - deepCopyFrom, 185
 - setRandom, 184
- m_bIndexMapGenerated
 - OSExpressionTree, 271
- MathUtil, 185
 - convertLinearConstraintCoefficientMatrixToThe↵
 - OtherMajor, 186
 - format_os_dtoa, 187
- Matrices, 187
 - deepCopyFrom, 188
 - setRandom, 188
- MatrixBlock, 188
 - alignsOnBlockBoundary, 190
 - cloneMatrixNode, 190
 - deepCopyFrom, 192
 - expandElements, 190
 - getMatrixNodeInXML, 189
 - getMatrixType, 189
 - getNodeName, 189
 - getNodeType, 189
 - setRandom, 190
- MatrixBlocks, 192
 - alignsOnBlockBoundary, 193
 - cloneMatrixNode, 195
 - deepCopyFrom, 195
 - getMatrixNodeInXML, 193
 - getMatrixType, 193
 - getNodeName, 193
 - getNodeType, 193
 - setRandom, 195
- MatrixCon, 196
- MatrixConstraintSolution, 197
- MatrixConstraints, 197
- MatrixConstructor, 198
- MatrixElementValues, 200
 - deepCopyFrom, 200
 - numberOfEI, 201
- MatrixElements, 199
 - isEqual, 199
- MatrixExpression, 201
 - shape, 202
- MatrixExpressionTree, 203
 - getPostfixFromExpressionTree, 203

- getPrefixFromExpressionTree, 203
- MatrixExpressions, 202
- matrixHasBase
 - OSInstance, 304
- MatrixNode, 204
 - alignsOnBlockBoundary, 206
 - cloneMatrixNode, 206
 - deepCopyFrom, 208
 - getMatrixNodeInXML, 205
 - getMatrixType, 205
 - getNodeName, 205
 - getNodeType, 205
 - getPostfixFromNodeTree, 206
 - getPrefixFromNodeTree, 206
 - postOrderMatrixNodeTraversal, 206
 - setRandom, 208
- MatrixObj, 208
- MatrixObjectiveSolution, 210
- MatrixObjectives, 209
- MatrixProgramming, 211
 - deepCopyFrom, 213
 - setRandom, 212
- MatrixProgrammingSolution, 213
 - deepCopyFrom, 214
 - setRandom, 214
- MatrixTransformation, 214
 - alignsOnBlockBoundary, 216
 - cloneMatrixNode, 216
 - deepCopyFrom, 217
 - getMatrixNodeInXML, 216
 - getMatrixType, 215
 - getNodeName, 215
 - getNodeType, 215
 - setRandom, 216
 - shape, 217
- MatrixType, 217
 - alignsOnBlockBoundary, 219
 - convertToOtherMajor, 220
 - deepCopyFrom, 225
 - disassembleMatrix, 224
 - expandElements, 220
 - extractBlock, 222
 - getBlocks, 222
 - getColumnPartition, 220
 - getColumnPartitionSize, 220
 - getRowPartition, 219
 - getRowPartitionSize, 219
 - printExpandedMatrix, 219
 - processBlockPartition, 221
 - processBlocks, 221
 - setRandom, 224
 - symmetry, 225
 - type, 225
- MatrixVar, 225
 - varType, 226
- MatrixVariableSolution, 227
- MatrixVariableValues, 228
- MatrixVariables, 227
- matrixWithMatrixVarIdx
 - OSGLParserData, 273
- MaxTime, 229
- MinCPUNumber, 230
- MinCPUSpeed, 231
- MinDiskSpace, 232
- MinMemorySize, 233
- MixedRowReferenceMatrixElements, 234
 - alignsOnBlockBoundary, 235
 - cloneMatrixNode, 236
 - deepCopyFrom, 236
 - getMatrixNodeInXML, 235
 - getMatrixType, 235
 - getNodeName, 235
 - getNodeType, 235
 - setRandom, 236
 - value, 236
- name
 - OtherMatrixVariableResult, 603
- NI, 237
 - shape, 238
- nlNodePoint
 - OSnLPParserData, 438
- NonlinearExpressions, 238
- NonnegativeCone, 239
 - deepCopyFrom, 240
 - getConeInXML, 239
 - getConeName, 239
 - setRandom, 240
- NonpositiveCone, 240
 - deepCopyFrom, 242
 - getConeInXML, 241
 - getConeName, 241
 - setRandom, 241
- normScaleFactor
 - QuadraticCone, 648
 - RotatedQuadraticCone, 652
- numberAttributePresent
 - OSrLPParserData, 582
- numberOfEI
 - MatrixElementValues, 201
- numberOfOtherIndexes
 - Cone, 60
 - DualCone, 91
 - IntersectionCone, 161
 - PolarCone, 638
 - PolyhedralCone, 640
 - ProductCone, 645
 - QuadraticCone, 648

- RotatedQuadraticCone, [652](#)
- SemidefiniteCone, [656](#)
- nvarcovered
 - OSILParserData, [280](#)
- OS_AMPL_SUFFIX, [264](#)
- OSCommandLine, [264](#)
 - browser, [268](#)
 - convertSolverNameToUpperCase, [267](#)
 - osilOutputFile, [267](#)
 - osinstance, [267](#)
 - osolOutputFile, [267](#)
 - osoption, [267](#)
 - osplOutputFile, [268](#)
 - quit, [268](#)
 - serviceMethod, [267](#)
- OSCommandLine.h, [711](#)
- OSCommandLineReader, [268](#)
 - OSCommandLineReader, [269](#)
 - parseString, [269](#)
 - readCommandLine, [269](#)
- OSCommandLineReader.h, [711](#)
- OSCouenneSolver.h, [712](#)
- OSCsdpSolver.h, [712](#)
- OSExpressionTree, [270](#)
 - bADMustReTape, [271](#)
 - m_bIndexMapGenerated, [271](#)
- OSExpressionTree.h, [713](#)
- OSGeneral, [272](#)
- OSGeneral.h, [714](#)
- OSInstance, [282](#)
 - addCone, [318–323](#)
 - addConstraint, [314](#)
 - addMatrix, [317](#)
 - addObjective, [313](#)
 - addQTermsToExpressionTree, [329](#)
 - addQTermsToExressionTree, [329](#)
 - addVariable, [312](#)
 - calculateAllConstraintFunctionGradients, [326](#)
 - calculateAllConstraintFunctionValues, [325](#)
 - calculateAllObjectiveFunctionGradients, [327](#)
 - calculateAllObjectiveFunctionValues, [325, 326](#)
 - calculateConstraintFunctionGradient, [326, 327](#)
 - calculateFunctionValue, [324](#)
 - calculateHessian, [328](#)
 - calculateLagrangianHessian, [328](#)
 - calculateObjectiveFunctionGradient, [327, 328](#)
 - copyLinearConstraintCoefficients, [315](#)
 - createOSADFun, [330](#)
 - forwardAD, [330](#)
 - getADSparsityHessian, [331](#)
 - getAllMatrixExpressionTrees, [308](#)
 - getAllMatrixExpressionTreesMod, [308](#)
 - getAllNonlinearExpressionTrees, [301](#)
 - getAllNonlinearExpressionTreesMod, [301](#)
 - getAllNonlinearVariablesIndexMap, [329](#)
 - getConstraintConstants, [297](#)
 - getConstraintLowerBounds, [296](#)
 - getConstraintNames, [296](#)
 - getConstraintNumber, [296](#)
 - getConstraintTypes, [297](#)
 - getConstraintUpperBounds, [296](#)
 - getDenseObjectiveCoefficients, [296](#)
 - getFirstOrderResults, [331](#)
 - getInstanceCreator, [291](#)
 - getInstanceDescription, [291](#)
 - getInstanceLicence, [291](#)
 - getInstanceName, [290](#)
 - getInstanceSource, [290](#)
 - getIterateResults, [331](#)
 - getJacobianSparsityPattern, [330](#)
 - getLagrangianExpTree, [329](#)
 - getLagrangianHessianSparsityPattern, [329](#)
 - getLinearConstraintCoefficientMajor, [297](#)
 - getLinearConstraintCoefficientNumber, [297](#)
 - getLinearConstraintCoefficientsInColumnMajor, [298](#)
 - getLinearConstraintCoefficientsInRowMajor, [298](#)
 - getMatrix, [304](#)
 - getMatrixBlockInColumnMajorForm, [304](#)
 - getMatrixCoefficientsInColumnMajor, [304](#)
 - getMatrixCoefficientsInRowMajor, [304](#)
 - getMatrixExpressionTree, [307](#)
 - getMatrixExpressionTreeInInfix, [307](#)
 - getMatrixExpressionTreeInPostfix, [307](#)
 - getMatrixExpressionTreeInPrefix, [307](#)
 - getMatrixExpressionTreeIndexes, [308](#)
 - getMatrixExpressionTreeModInPostfix, [307](#)
 - getMatrixExpressions, [306](#)
 - getMatrixName, [303](#)
 - getMatrixNumber, [302](#)
 - getMatrixSymmetry, [302](#)
 - getMatrixType, [302](#)
 - getNonlinearExpressionTree, [299](#)
 - getNonlinearExpressionTreeInInfix, [300](#)
 - getNonlinearExpressionTreeInPostfix, [300](#)
 - getNonlinearExpressionTreeInPrefix, [300](#)
 - getNonlinearExpressionTreeIndexes, [301](#)
 - getNonlinearExpressionTreeMod, [299](#)
 - getNonlinearExpressionTreeModInPostfix, [300](#)
 - getNonlinearExpressionTreeModInPrefix, [300](#)
 - getNonlinearExpressionTreeModIndexes, [301](#)
 - getNonlinearExpressions, [299](#)
 - getNumberOfBinaryVariables, [292](#)
 - getNumberOfColumnsForMatrix, [303](#)
 - getNumberOfIntegerVariables, [292](#)
 - getNumberOfMatrixConstraints, [306](#)
 - getNumberOfMatrixExpressionTreeIndexes, [308](#)
 - getNumberOfMatrixExpressions, [306](#)

- getNumberOfMatrixObjectives, 306
- getNumberOfMatrixVariables, 306
- getNumberOfNonlinearConstraints, 301
- getNumberOfNonlinearExpressionTreeIndexes, 301
- getNumberOfNonlinearExpressionTreeModIndexes, 302
- getNumberOfNonlinearExpressions, 299
- getNumberOfNonlinearObjectives, 301
- getNumberOfQuadraticRowIndexes, 299
- getNumberOfQuadraticTerms, 298
- getNumberOfRowsForMatrix, 303
- getNumberOfSemiContinuousVariables, 293
- getNumberOfSemiIntegerVariables, 293
- getNumberOfStringVariables, 293
- getNumberOfValuesForMatrix, 303
- getObjectiveCoefficientNumbers, 294
- getObjectiveCoefficients, 295
- getObjectiveConstants, 295
- getObjectiveMaxOrMins, 294
- getObjectiveNames, 294
- getObjectiveNumber, 294
- getObjectiveWeights, 295
- getQuadraticRowIndexes, 299
- getQuadraticTerms, 298
- getSecondOrderResults, 333
- getSparseJacobianFromColumnMajor, 329
- getSparseJacobianFromRowMajor, 329
- getTimeDomainFormat, 308
- getTimeDomainIntervalHorizon, 310
- getTimeDomainIntervalStart, 310
- getTimeDomainStageConList, 309
- getTimeDomainStageNames, 309
- getTimeDomainStageNumber, 308
- getTimeDomainStageNumberOfConstraints, 309
- getTimeDomainStageNumberOfObjectives, 309
- getTimeDomainStageNumberOfVariables, 309
- getTimeDomainStageObjList, 310
- getTimeDomainStageVarList, 309
- getVariableLowerBounds, 293
- getVariableNames, 291
- getVariableNumber, 291
- getVariableTypes, 292
- getVariableUpperBounds, 293
- getZeroOrderResults, 331
- initForAlgDiff, 333
- initObjGradients, 333
- initializeNonLinearStructures, 324
- matrixHasBase, 304
- printModel, 324
- reverseAD, 330
- setConeNumber, 318
- setConstraintNumber, 314
- setConstraints, 314
- setInstanceCreator, 311
- setInstanceDescription, 311
- setInstanceLicence, 311
- setInstanceName, 310
- setInstanceSource, 310
- setLinearConstraintCoefficients, 315
- setMatrixNumber, 317
- setNonlinearExpressions, 317
- setNumberOfQuadraticTerms, 316
- setObjectiveNumber, 313
- setObjectives, 313
- setQuadraticCoefficients, 316
- setQuadraticTermsInNonlinearExpressions, 316
- setTimeDomainStageConstraintsOrdered, 334
- setTimeDomainStageConstraintsUnordered, 334
- setTimeDomainStageObjectivesOrdered, 334
- setTimeDomainStageObjectivesUnordered, 334
- setTimeDomainStageVariablesOrdered, 333
- setTimeDomainStageVariablesUnordered, 333
- setVariableNumber, 311
- setVariables, 312
- OSInstance.h, 720
- OSMatlab, 334
 - solve, 336
- OSMatrix, 337
 - alignsOnBlockBoundary, 339
 - cloneMatrixNode, 340
 - createConstructorTreeFromPrefix, 337
 - deepCopyFrom, 340
 - expandElements, 338
 - getMatrixNodeInXML, 340
 - getMatrixType, 338
 - getNodeName, 338
 - getNodeType, 338
 - setMatrix, 339
 - setRandom, 340
- OSMatrix.h, 723
- OSMatrixWithMatrixConIdx, 341
 - cloneMatrixNode, 341
 - deepCopyFrom, 342
 - getMatrixNodeInXML, 341
 - setRandom, 342
- OSMatrixWithMatrixObjIdx, 342
 - cloneMatrixNode, 343
 - deepCopyFrom, 343
 - getMatrixNodeInXML, 343
 - setRandom, 343
- OSMatrixWithMatrixVarIdx, 344
 - cloneMatrixNode, 344
 - deepCopyFrom, 345
 - getMatrixNodeInXML, 344
 - setRandom, 345
- OSOption, 442
 - deepCopyFrom, 452
 - getAllOtherConstraintOptions, 467

[getAllOtherObjectiveOptions](#), 463
[getAllOtherOptions](#), 455
[getAllOtherVariableOptions](#), 461
[getAllSolverOptions](#), 468
[getDirectoriesToDelete](#), 457
[getDirectoriesToMake](#), 456
[getFilesToDelete](#), 457
[getFilesToMake](#), 456
[getInitBasisStatusDense](#), 459
[getInitBasisStatusSparse](#), 458
[getInitConValuesDense](#), 464
[getInitConValuesSparse](#), 464
[getInitDualVarLowerBoundsDense](#), 464
[getInitDualVarUpperBoundsDense](#), 466
[getInitDualVarValuesSparse](#), 464
[getInitObjBoundsSparse](#), 462
[getInitObjLowerBoundsDense](#), 462
[getInitObjUpperBoundsDense](#), 462
[getInitObjValuesDense](#), 461
[getInitObjValuesSparse](#), 461
[getInitVarValuesDense](#), 457, 458
[getInitVarValuesSparse](#), 457
[getInitVarValuesStringDense](#), 458
[getInitVarValuesStringSparse](#), 458
[getInitialBasisElements](#), 459
[getInputDirectoriesToMove](#), 456
[getInputFilesToMove](#), 456
[getIntegerVariableBranchingWeightsDense](#), 460
[getIntegerVariableBranchingWeightsSparse](#), 460
[getJobDependencies](#), 455
[getJobID](#), 452
[getNumberOfInitConValues](#), 454
[getNumberOfInitDualVarValues](#), 454
[getNumberOfInitObjBounds](#), 453
[getNumberOfInitObjValues](#), 453
[getNumberOfInitVarValues](#), 452
[getNumberOfInitVarValuesString](#), 452
[getNumberOfInitialBasisElements](#), 459
[getNumberOfIntegerVariableBranchingWeights](#), 452
[getNumberOfOtherConstraintOptions](#), 454
[getNumberOfOtherObjectiveOptions](#), 453
[getNumberOfOtherVariableOptions](#), 453
[getNumberOfSOS](#), 453
[getNumberOfSOSVarBranchingWeights](#), 453
[getNumberOfSolverOptions](#), 454
[getObjectiveInitialBasisStatusDense](#), 463
[getOtherConstraintOption](#), 467
[getOtherConstraintOptions](#), 466
[getOtherGeneralOptions](#), 454
[getOtherJobOptions](#), 455
[getOtherObjectiveOption](#), 463
[getOtherObjectiveOptions](#), 463
[getOtherOptions](#), 455
[getOtherServiceOptions](#), 455
[getOtherSystemOptions](#), 454
[getOtherVariableOption](#), 461
[getOtherVariableOptions](#), 460
[getOutputDirectoriesToMove](#), 456
[getOutputFilesToMove](#), 457
[getProcessesToKill](#), 457
[getRequiredDirectories](#), 455
[getRequiredFiles](#), 456
[getSOSVariableBranchingWeightsSparse](#), 460
[getSlackVariableInitialBasisStatusDense](#), 466
[getSolverOptions](#), 467
[getVariableInitialBasisStatusDense](#), 459
[optionHeader](#), 474
[setAnOtherGeneralOption](#), 468
[setAnotherInitBasisStatus](#), 469
[setHeader](#), 451
[setMinCPUNumber](#), 469
[setMinCPUSpeed](#), 469
[setMinDiskSpace](#), 468
[setMinMemorySize](#), 468
[setOtherConstraintOptionAttributes](#), 473
[setOtherConstraintOptionCon](#), 474
[setOtherGeneralOptions](#), 468
[setOtherObjectiveOptionAttributes](#), 471
[setOtherObjectiveOptionObj](#), 473
[setOtherOptionOrResultEnumeration](#), 471
[setOtherVariableOptionAttributes](#), 469
[setOtherVariableOptionVar](#), 471
[setPathPairs](#), 469
[setRandom](#), 452
[setSolverOptionContent](#), 474
[OSOption.h](#), 731
[OSOptionsStruc.h](#), 733
[OSOutput](#), 478
 [AddChannel](#), 481
 [DeleteChannel](#), 481
 [FindChannel](#), 481
 [OSPrint](#), 479
 [SetPrintLevel](#), 479, 481
[OSOutput.h](#), 734
[OSOutputChannel](#), 482
 [OSOutputChannel](#), 483
 [OSPrintf](#), 484
 [setAllPrintLevels](#), 483
 [setPrintLevel](#), 483
[OSParameters.h](#), 735
 [ENUM_BASIS_STATUS](#), 736
 [ENUM_OUTPUT_AREA](#), 736
 [ENUM_OUTPUT_LEVEL](#), 736
[OSPrint](#)
 [OSOutput](#), 479
[OSPrintf](#)
 [OSOutputChannel](#), 484
[OSReferencedObject](#), 484

- OSReferencer, [486](#)
- OSResult, [486](#)
 - addTimingInformation, [526](#)
 - getAnotherVariableResultNumberOfVar, [507](#)
 - getBasisInformationDense, [506](#)
 - getBasisStatusEI, [506](#)
 - getBasisStatusNumberOfEI, [506](#)
 - getConstraintNumber, [504](#)
 - getGeneralMessage, [502](#)
 - getGeneralStatus, [500](#)
 - getGeneralStatusDescription, [500](#)
 - getGeneralStatusType, [500](#)
 - getGeneralSubstatusDescription, [502](#)
 - getGeneralSubstatusName, [500](#)
 - getInstanceName, [502](#)
 - getJobID, [502](#)
 - getNumberOfGeneralSubstatuses, [500](#)
 - getNumberOfOtherGeneralResults, [503](#)
 - getNumberOfOtherVariableResults, [507](#)
 - getObjectiveNumber, [504](#)
 - getOptimalDualVariableValues, [511](#)
 - getOptimalObjValue, [509](#)
 - getOptimalPrimalVariableValues, [506](#)
 - getOtherConstraintResultArrayDense, [514](#)
 - getOtherConstraintResultArrayType, [513](#)
 - getOtherConstraintResultEnumerationDescription, [513](#)
 - getOtherConstraintResultEnumerationEI, [514](#)
 - getOtherConstraintResultEnumerationNumberOfEI, [513](#)
 - getOtherConstraintResultEnumerationValue, [513](#)
 - getOtherGeneralResultName, [503](#)
 - getOtherObjectiveResultArrayDense, [511](#)
 - getOtherObjectiveResultArrayType, [509](#)
 - getOtherObjectiveResultEnumerationDescription, [509](#)
 - getOtherObjectiveResultEnumerationEI, [511](#)
 - getOtherObjectiveResultEnumerationNumberOfEI, [511](#)
 - getOtherObjectiveResultEnumerationValue, [509](#)
 - getOtherVariableResultArrayDense, [508](#)
 - getOtherVariableResultArrayType, [507](#)
 - getOtherVariableResultEnumerationDescription, [508](#)
 - getOtherVariableResultEnumerationEI, [508](#)
 - getOtherVariableResultEnumerationNumberOfEI, [508](#)
 - getOtherVariableResultEnumerationValue, [507](#)
 - getServiceName, [502](#)
 - getServiceURI, [502](#)
 - getSolutionMessage, [505](#)
 - getSolutionNumber, [504](#)
 - getSolutionStatus, [504](#)
 - getSolutionStatusDescription, [505](#)
 - getSolutionStatusType, [505](#)
 - getSolutionWeightedObjectives, [505](#)
 - getSolverInvoked, [503](#)
 - getTimeNumber, [503](#)
 - getTimeStamp, [503](#)
 - getTimeValue, [503](#)
 - getVariableNumber, [504](#)
 - setActualStartTime, [525](#)
 - setAnotherSolutionResult, [575](#)
 - setAnotherVariableResultDense, [541](#), [542](#)
 - setAnotherVariableResultSparse, [539](#), [541](#)
 - setAvailableCPUNumberDescription, [521](#)
 - setAvailableCPUNumberValue, [521](#)
 - setAvailableCPUSpeedDescription, [520](#)
 - setAvailableCPUSpeedUnit, [520](#)
 - setAvailableCPUSpeedValue, [521](#)
 - setAvailableDiskSpaceDescription, [519](#)
 - setAvailableDiskSpaceUnit, [519](#)
 - setAvailableDiskSpaceValue, [519](#)
 - setAvailableMemoryDescription, [520](#)
 - setAvailableMemoryUnit, [519](#)
 - setAvailableMemoryValue, [520](#)
 - setBasisStatus, [538](#)
 - setConstraintNumber, [531](#)
 - setConstraintValuesDense, [561](#)
 - setCurrentJobCount, [523](#)
 - setCurrentState, [522](#)
 - setDualValue, [561](#)
 - setDualVariableValuesDense, [561](#)
 - setDualVariableValuesSparse, [560](#)
 - setGeneralMessage, [516](#)
 - setGeneralStatus, [514](#)
 - setGeneralStatusDescription, [515](#)
 - setGeneralStatusType, [514](#)
 - setGeneralSubstatusDescription, [515](#)
 - setGeneralSubstatusName, [515](#)
 - setHeader, [499](#)
 - setInstanceName, [516](#)
 - setJobEndTime, [526](#)
 - setJobID, [517](#)
 - setJobStatus, [525](#)
 - setJobSubmitTime, [525](#)
 - setMatrixVarValuesAttributes, [569](#)
 - setMatrixVarValuesBlockElements, [570](#)
 - setMatrixVarValuesBlockStructure, [569](#)
 - setMatrixVariableSolution, [568](#)
 - setMatrixVariablesOtherResultBlockElements, [572](#)
 - setMatrixVariablesOtherResultBlockStructure, [572](#)
 - setMatrixVariablesOtherResultGeneralAttributes, [570](#)
 - setMatrixVariablesOtherResultMatrixAttributes, [571](#)
 - setNumberOfDualValues, [560](#)
 - setNumberOfDualVariableValues, [560](#)
 - setNumberOfGeneralSubstatuses, [515](#)
 - setNumberOfObjValues, [550](#)
 - setNumberOfObjectiveValues, [551](#)

setNumberOfOtherConstraintResults, 559
setNumberOfOtherGeneralResults, 517
setNumberOfOtherJobResults, 530
setNumberOfOtherObjectiveResults, 550
setNumberOfOtherServiceResults, 524
setNumberOfOtherSolutionResults, 573
setNumberOfOtherSystemResults, 521
setNumberOfOtherVariableResults, 539
setNumberOfPrimalVariableValues, 535
setNumberOfSolutionSubstatuses, 533
setNumberOfSolverOutputs, 576
setNumberOfTimes, 527
setNumberOfVarValues, 536
setNumberOfVarValuesString, 538
setObjValue, 552
setObjectiveNumber, 531
setObjectiveValuesDense, 551
setObjectiveValuesSparse, 551
setOtherConstraintResultCategory, 567
setOtherConstraintResultCon, 568
setOtherConstraintResultConIdx, 567
setOtherConstraintResultConName, 567
setOtherConstraintResultConType, 564
setOtherConstraintResultDescription, 566
setOtherConstraintResultEnumType, 565
setOtherConstraintResultName, 564
setOtherConstraintResultNumberOfCon, 563
setOtherConstraintResultNumberOfEnumerations, 563
setOtherConstraintResultSolver, 566
setOtherConstraintResultType, 564
setOtherConstraintResultValue, 565
setOtherGeneralResultDescription, 518
setOtherGeneralResultName, 518
setOtherGeneralResultValue, 518
setOtherJobResultDescription, 531
setOtherJobResultName, 530
setOtherJobResultValue, 530
setOtherObjectiveResultCategory, 557
setOtherObjectiveResultDescription, 555
setOtherObjectiveResultEnumType, 555
setOtherObjectiveResultName, 553
setOtherObjectiveResultNumberOfEnumerations, 553
setOtherObjectiveResultNumberOfObj, 552
setOtherObjectiveResultObj, 559
setOtherObjectiveResultObjIdx, 558
setOtherObjectiveResultObjName, 558
setOtherObjectiveResultObjType, 554
setOtherObjectiveResultSolver, 557
setOtherObjectiveResultType, 554
setOtherObjectiveResultValue, 555
setOtherOptionOrResultEnumeration, 549
setOtherServiceResultDescription, 524
setOtherServiceResultName, 524
setOtherServiceResultValue, 524
setOtherSolutionResultCategory, 574
setOtherSolutionResultDescription, 574
setOtherSolutionResultItem, 575
setOtherSolutionResultName, 573
setOtherSolutionResultNumberOfItems, 575
setOtherSolutionResultValue, 574
setOtherSystemResultDescription, 522
setOtherSystemResultName, 522
setOtherSystemResultValue, 522
setOtherVariableResultCategory, 546
setOtherVariableResultDescription, 546
setOtherVariableResultEnumType, 545
setOtherVariableResultName, 543
setOtherVariableResultNumberOfEnumerations, 543
setOtherVariableResultNumberOfVar, 542
setOtherVariableResultSolver, 546
setOtherVariableResultType, 544
setOtherVariableResultValue, 545
setOtherVariableResultVar, 549
setOtherVariableResultVarIdx, 548
setOtherVariableResultVarName, 548
setOtherVariableResultVarType, 544
setPrimalVariableValuesDense, 536
setPrimalVariableValuesSparse, 535
setRandom, 500
setScheduledStartTime, 525
setServiceName, 516
setServiceURI, 516
setServiceUtilization, 523
setSolutionMessage, 535
setSolutionNumber, 532
setSolutionStatus, 532
setSolutionStatusDescription, 533
setSolutionStatusType, 532
setSolutionSubstatusDescription, 533
setSolutionSubstatusType, 533
setSolutionTargetObjectiveIdx, 534
setSolutionTargetObjectiveName, 534
setSolutionWeightedObjectives, 534
setSolverInvoked, 517
setSolverOutputCategory, 576
setSolverOutputDescription, 576
setSolverOutputItem, 578
setSolverOutputName, 576
setSolverOutputNumberOfItems, 578
setSystemInformation, 518
setTime, 526
setTimeNumber, 527
setTimeServiceStarted, 523
setTimeStamp, 517
setTimingInformation, 526
setTotalJobsSoFar, 523

- setUsedCPUNumberDescription, 529
 - setUsedCPUNumberValue, 530
 - setUsedCPUSpeedDescription, 529
 - setUsedCPUSpeedUnit, 529
 - setUsedCPUSpeedValue, 529
 - setUsedDiskSpaceDescription, 527
 - setUsedDiskSpaceUnit, 527
 - setUsedDiskSpaceValue, 528
 - setUsedMemoryDescription, 528
 - setUsedMemoryUnit, 528
 - setUsedMemoryValue, 528
 - setVarValue, 536
 - setVarValueString, 538
 - setVariableNumber, 531
- OSResult.h, 737
- OSRunSolver.h, 740
 - runSolver, 741, 742
 - selectSolver, 742
- OSSmartPtr
 - ~OSSmartPtr, 589
 - GetRawPtr, 590
 - IsNull, 590
 - IsValid, 590
 - operator!=, 590
 - operator*, 589
 - operator->, 589
 - operator==, 589
- OSSmartPtr< T >, 586
- OSSolverAgent, 590
 - fileUpload, 594
 - getJobID, 593
 - kill, 593
 - knock, 594
 - OSSolverAgent, 592
 - retrieve, 594
 - send, 593
 - solve, 593
- OSSolverAgent.h, 743
- OSStringUtil.h, 743
 - writeStringData, 744
- OSWSUtil.h, 745
- OSgLParserData, 272
 - matrixWithMatrixVarIdx, 273
- OSgLParserData.h, 715
- OSgLWriter.h, 716
 - writeBasisStatus, 718
 - writeDbIVectorData, 718
 - writeGeneralFileHeader, 717
 - writeIntVectorData, 717
 - writeOtherOptionOrResultEnumeration, 718
- OSgams2osil, 271
 - createOSInstance, 271
 - getOSInstance, 272
 - takeOverOSInstance, 272
- OShL, 274
 - getJobID, 275
 - kill, 275
 - knock, 277
 - retrieve, 275
 - send, 275
 - solve, 274
- OShL.h, 718
- OSiLParserData, 277
 - nvarcovered, 280
 - qtermcount, 279
 - stageVariableStartIdx, 279
 - stageVariablesON, 279
 - stageVariablesOrdered, 279
 - timeDomainStages, 279
- OSiLParserData.h, 719
- OSiLReader, 280
 - readOSiL, 280
- OSiLReader.h, 719
- OSiLWriter, 281
 - writeOSiL, 281
- OSiLWriter.h, 720
- OSmps2OS, 345
 - createOSObjects, 346
 - jobID, 347
 - osol, 347
- OSmps2OS.h, 725
- OSmps2osil, 347
 - createOSInstance, 348
- OSmps2osil.h, 726
- OSnLMNode, 352
 - copyNodeAndDescendants, 354
 - createExpressionTreeFromPostfix, 354
 - createExpressionTreeFromPrefix, 353
 - getPostfixFromExpressionTree, 354
 - getPrefixFromExpressionTree, 353
 - postOrderOSnLMNodeTraversal, 354
 - preOrderOSnLMNodeTraversal, 353
- OSnLMNodeDiagonalMatrixFromVector, 355
 - cloneExprNode, 355
 - getTokenName, 355
- OSnLMNodeIdentityMatrix, 356
 - cloneExprNode, 356
 - getTokenName, 356
- OSnLMNodeMatrixCon, 357
 - cloneExprNode, 358
 - copyNodeAndDescendants, 358
 - getNonlinearExpressionInXML, 357
 - getTokenName, 357
 - getTokenNumber, 357
- OSnLMNodeMatrixDiagonal, 358
 - cloneExprNode, 359
 - getTokenName, 359
- OSnLMNodeMatrixDotTimes, 359

- cloneExprNode, 360
- getTokenName, 359
- OSnLMNodeMatrixInverse, 360
 - cloneExprNode, 360
 - getTokenName, 360
- OSnLMNodeMatrixLowerTriangle, 361
 - cloneExprNode, 362
 - copyNodeAndDescendants, 362
 - getNonlinearExpressionInXML, 362
 - getTokenName, 361
- OSnLMNodeMatrixMinus, 362
 - cloneExprNode, 363
 - getTokenName, 363
- OSnLMNodeMatrixNegate, 363
 - cloneExprNode, 364
 - getTokenName, 364
- OSnLMNodeMatrixObj, 364
 - cloneExprNode, 365
 - copyNodeAndDescendants, 365
 - getNonlinearExpressionInXML, 365
 - getTokenName, 365
 - getTokenNumber, 365
- OSnLMNodeMatrixPlus, 366
 - cloneExprNode, 366
 - getTokenName, 366
- OSnLMNodeMatrixProduct, 366
 - cloneExprNode, 367
 - getTokenName, 367
- OSnLMNodeMatrixReference, 368
 - cloneExprNode, 369
 - copyNodeAndDescendants, 369
 - getNonlinearExpressionInXML, 369
 - getTokenName, 369
 - getTokenNumber, 369
- OSnLMNodeMatrixScalarTimes, 369
 - cloneExprNode, 370
 - getTokenName, 370
- OSnLMNodeMatrixSubmatrixAt, 370
 - cloneExprNode, 371
 - getTokenName, 371
- OSnLMNodeMatrixSum, 371
 - cloneExprNode, 372
 - getTokenName, 372
- OSnLMNodeMatrixTimes, 372
 - cloneExprNode, 373
 - getTokenName, 373
- OSnLMNodeMatrixTranspose, 373
 - cloneExprNode, 374
 - getTokenName, 373
- OSnLMNodeMatrixUpperTriangle, 374
 - cloneExprNode, 375
 - copyNodeAndDescendants, 375
 - getNonlinearExpressionInXML, 375
 - getTokenName, 375
- OSnLMNodeMatrixVar, 375
 - cloneExprNode, 376
 - copyNodeAndDescendants, 377
 - getNonlinearExpressionInXML, 376
 - getTokenName, 376
 - getTokenNumber, 376
- OSnLNode, 377
 - calculateFunction, 379
 - constructADTape, 379
 - copyNodeAndDescendants, 381
 - createExpressionTreeFromPostfix, 380
 - createExpressionTreeFromPrefix, 379
 - getPostfixFromExpressionTree, 380
 - getPrefixFromExpressionTree, 380
 - getVariableIndexMap, 378
 - postOrderOSnLNodeTraversal, 381
 - preOrderOSnLNodeTraversal, 380
- OSnLNode.h, 726
- OSnLNodeAbs, 381
 - calculateFunction, 382
 - cloneExprNode, 383
 - constructADTape, 383
 - getTokenName, 382
- OSnLNodeAllDiff, 383
 - calculateFunction, 384
 - cloneExprNode, 385
 - constructADTape, 385
 - getTokenName, 384
- OSnLNodeCos, 385
 - calculateFunction, 386
 - cloneExprNode, 387
 - constructADTape, 387
 - getTokenName, 386
- OSnLNodeDivide, 387
 - calculateFunction, 388
 - cloneExprNode, 389
 - constructADTape, 389
 - getTokenName, 388
- OSnLNodeE, 389
 - calculateFunction, 391
 - cloneExprNode, 391
 - constructADTape, 391
 - getNonlinearExpressionInXML, 390
 - getTokenName, 390
 - getTokenNumber, 390
- OSnLNodeErf, 391
 - calculateFunction, 392
 - cloneExprNode, 393
 - constructADTape, 393
 - getTokenName, 392
- OSnLNodeExp, 393
 - calculateFunction, 394
 - cloneExprNode, 395
 - constructADTape, 395

- getTokenName, 394
- OSnLNodeIf, 395
 - calculateFunction, 396
 - cloneExprNode, 397
 - constructADTape, 397
 - getTokenName, 396
- OSnLNodeLn, 397
 - calculateFunction, 398
 - cloneExprNode, 399
 - constructADTape, 399
 - getTokenName, 398
- OSnLNodeMatrixDeterminant, 399
 - calculateFunction, 400
 - cloneExprNode, 401
 - constructADTape, 400
 - getTokenName, 400
- OSnLNodeMatrixToScalar, 401
 - calculateFunction, 402
 - cloneExprNode, 402
 - constructADTape, 402
 - getTokenName, 402
- OSnLNodeMatrixTrace, 403
 - calculateFunction, 404
 - cloneExprNode, 404
 - constructADTape, 404
 - getTokenName, 404
- OSnLNodeMax, 405
 - calculateFunction, 406
 - cloneExprNode, 406
 - constructADTape, 406
 - getTokenName, 406
- OSnLNodeMin, 407
 - calculateFunction, 408
 - cloneExprNode, 408
 - constructADTape, 408
 - getTokenName, 408
- OSnLNodeMinus, 409
 - calculateFunction, 410
 - cloneExprNode, 410
 - constructADTape, 410
 - getTokenName, 410
- OSnLNodeNegate, 411
 - calculateFunction, 412
 - cloneExprNode, 412
 - constructADTape, 412
 - getTokenName, 412
- OSnLNodeNumber, 413
 - calculateFunction, 414
 - cloneExprNode, 414
 - constructADTape, 415
 - copyNodeAndDescendants, 415
 - getNonlinearExpressionInXML, 414
 - getTokenName, 414
 - getTokenNumber, 414
- id, 415
- OSnLNodePI, 415
 - calculateFunction, 417
 - cloneExprNode, 417
 - constructADTape, 417
 - getNonlinearExpressionInXML, 417
 - getTokenName, 416
 - getTokenNumber, 416
- OSnLNodePlus, 418
 - calculateFunction, 419
 - cloneExprNode, 419
 - constructADTape, 419
 - getTokenName, 419
- OSnLNodePower, 419
 - calculateFunction, 420
 - cloneExprNode, 421
 - constructADTape, 421
 - getTokenName, 420
- OSnLNodeProduct, 421
 - calculateFunction, 422
 - cloneExprNode, 423
 - constructADTape, 423
 - getTokenName, 422
- OSnLNodeSin, 423
 - calculateFunction, 424
 - cloneExprNode, 425
 - constructADTape, 425
 - getTokenName, 424
- OSnLNodeSqrt, 425
 - calculateFunction, 426
 - cloneExprNode, 427
 - constructADTape, 427
 - getTokenName, 426
- OSnLNodeSquare, 427
 - calculateFunction, 428
 - cloneExprNode, 429
 - constructADTape, 429
 - getTokenName, 428
- OSnLNodeSum, 429
 - calculateFunction, 430
 - cloneExprNode, 431
 - constructADTape, 431
 - getTokenName, 430
- OSnLNodeTimes, 431
 - calculateFunction, 432
 - cloneExprNode, 433
 - constructADTape, 433
 - getTokenName, 432
- OSnLNodeVariable, 433
 - calculateFunction, 435
 - cloneExprNode, 435
 - constructADTape, 436
 - copyNodeAndDescendants, 435
 - getNonlinearExpressionInXML, 435

- getTokenName, [435](#)
- getTokenNumber, [435](#)
- getVariableIndexMap, [434](#)
- OSnLParserData, [436](#)
 - nINodePoint, [438](#)
 - tmpnlcount, [438](#)
- OSnLParserData.h, [729](#)
- OSnI2OS, [348](#)
 - createOSObjects, [350](#)
 - getASL, [350](#)
 - jobID, [351](#)
 - OSnI2OS, [350](#)
 - osol, [351](#)
 - readNI, [350](#)
 - setASL, [350](#)
 - setIBVar, [351](#)
 - setVar, [351](#)
 - walkTree, [351](#)
- OSoLParserData, [439](#)
- OSoLParserData.h, [729](#)
- OSoLReader, [440](#)
 - readOSoL, [441](#)
- OSoLReader.h, [730](#)
- OSoLWriter, [441](#)
 - writeOSoL, [442](#)
- OSoLWriter.h, [730](#)
- OSosl2ampl, [477](#)
 - writeSolFile, [478](#)
- OSosl2ampl.h, [734](#)
- OSrL2Gams, [578](#)
 - OSrL2Gams, [579](#)
 - writeSolution, [579](#)
- OSrLParserData, [579](#)
 - categoryAttribute, [582](#)
 - kounter, [582](#)
 - numberAttributePresent, [582](#)
- OSrLParserData.h, [739](#)
- OSrLReader, [583](#)
 - readOSrL, [584](#)
- OSrLReader.h, [739](#)
- OSrLWriter, [585](#)
 - writeOSrL, [585](#)
- OSrLWriter.h, [740](#)
- ObjCoef, [242](#)
- ObjReferenceMatrixElements, [249](#)
 - alignsOnBlockBoundary, [250](#)
 - cloneMatrixNode, [251](#)
 - deepCopyFrom, [251](#)
 - getMatrixNodeInXML, [250](#)
 - getMatrixType, [250](#)
 - getNodeName, [250](#)
 - getNodeType, [250](#)
 - setRandom, [251](#)
- ObjReferenceMatrixValues, [251](#)
 - deepCopyFrom, [252](#)
 - setRandom, [252](#)
- ObjValue, [253](#)
 - setRandom, [254](#)
- Objective, [243](#)
- ObjectiveOption, [244](#)
 - addOther, [245](#)
 - deepCopyFrom, [245](#)
 - setOther, [245](#)
 - setRandom, [245](#)
- ObjectiveSolution, [246](#)
 - setRandom, [247](#)
- ObjectiveValues, [248](#)
 - setRandom, [249](#)
- Objectives, [246](#)
- operator!=
 - OSSmartPtr, [590](#)
- operator*
 - OSSmartPtr, [589](#)
- operator->
 - OSSmartPtr, [589](#)
- operator==
 - OSSmartPtr, [589](#)
- OptimizationOption, [255](#)
 - deepCopyFrom, [256](#)
 - setRandom, [256](#)
- OptimizationResult, [257](#)
 - setRandom, [258](#)
- OptimizationSolution, [258](#)
 - setRandom, [259](#)
- OptimizationSolutionStatus, [260](#)
 - setRandom, [261](#)
- OptimizationSolutionSubstatus, [261](#)
 - setRandom, [262](#)
- optimize
 - LindoSolver, [176](#)
- optionHeader
 - OSOption, [474](#)
- OrthantCone, [262](#)
 - deepCopyFrom, [264](#)
 - getConeInXML, [263](#)
 - getConeName, [263](#)
 - setRandom, [263](#)
- osOptionsStruc, [475](#)
 - browser, [477](#)
 - serviceMethod, [477](#)
 - solverName, [477](#)
- osilOutputFile
 - OSCommandLine, [267](#)
- osinstance
 - OSCommandLine, [267](#)
- osol
 - OSmps2OS, [347](#)
 - OSnI2OS, [351](#)

- osolOutputFile
 - OSCommandLine, [267](#)
- osoption
 - OSCommandLine, [267](#)
- osplOutputFile
 - OSCommandLine, [268](#)
- OtherConOption, [595](#)
 - deepCopyFrom, [596](#)
 - setRandom, [596](#)
- OtherConResult, [596](#)
 - setRandom, [597](#)
- OtherConstraintOption, [597](#)
 - addCon, [600](#)
 - deepCopyFrom, [599](#)
 - setCon, [599](#)
 - setRandom, [599](#)
- OtherConstraintResult, [600](#)
 - setRandom, [601](#)
- OtherMatrixVariableResult, [602](#)
 - enumeration, [603](#)
 - name, [603](#)
- OtherObjOption, [607](#)
 - deepCopyFrom, [608](#)
 - setRandom, [608](#)
- OtherObjResult, [609](#)
 - setRandom, [610](#)
- OtherObjectiveOption, [603](#)
 - addObj, [605](#)
 - deepCopyFrom, [605](#)
 - setObj, [605](#)
 - setRandom, [604](#)
- OtherObjectiveResult, [605](#)
 - setRandom, [607](#)
- OtherOption, [610](#)
 - deepCopyFrom, [611](#)
 - setRandom, [611](#)
- OtherOptionOrResultEnumeration, [612](#)
 - deepCopyFrom, [614](#)
 - setOtherOptionOrResultEnumeration, [614](#)
 - setRandom, [613](#)
- OtherOptions, [614](#)
 - addOther, [616](#)
 - deepCopyFrom, [616](#)
 - setOther, [616](#)
 - setRandom, [615](#)
- OtherResult, [616](#)
 - setRandom, [617](#)
- OtherResults, [618](#)
 - setRandom, [619](#)
- OtherSolutionResult, [619](#)
 - setRandom, [620](#)
- OtherSolutionResults, [620](#)
 - setRandom, [621](#)
- OtherSolverOutput, [622](#)
 - setRandom, [623](#)
- otherVarIndex
 - OtherVariableResultStruct, [627](#)
- OtherVarOption, [628](#)
 - deepCopyFrom, [629](#)
 - setRandom, [629](#)
- OtherVarResult, [630](#)
 - setRandom, [631](#)
- otherVarText
 - OtherVariableResultStruct, [627](#)
- OtherVariableOption, [623](#)
 - addVar, [625](#)
 - deepCopyFrom, [624](#)
 - setRandom, [624](#)
 - setVar, [625](#)
- OtherVariableResult, [625](#)
 - setRandom, [626](#)
- OtherVariableResultStruct, [627](#)
 - otherVarIndex, [627](#)
 - otherVarText, [627](#)
- parseString
 - OSCommandLineReader, [269](#)
- PathPair, [632](#)
 - deepCopyFrom, [633](#)
 - setRandom, [633](#)
- PathPairs, [633](#)
 - addPathPair, [636](#)
 - deepCopyFrom, [636](#)
 - setPathPair, [636](#)
 - setRandom, [635](#)
- PolarCone, [637](#)
 - deepCopyFrom, [638](#)
 - getConeName, [638](#)
 - numberOfOtherIndexes, [638](#)
 - setRandom, [638](#)
- PolyhedralCone, [638](#)
 - deepCopyFrom, [640](#)
 - getConeInXML, [639](#)
 - getConeName, [639](#)
 - numberOfOtherIndexes, [640](#)
 - setRandom, [640](#)
- postOrderMatrixNodeTraversal
 - MatrixNode, [206](#)
- postOrderOSnLNodeTraversal
 - ExprNode, [101](#)
 - OSnLMNode, [354](#)
 - OSnLNode, [381](#)
- preOrderOSnLNodeTraversal
 - ExprNode, [100](#)
 - OSnLMNode, [353](#)
 - OSnLNode, [380](#)
- printExpandedMatrix
 - MatrixType, [219](#)

- printModel
 - OSInstance, [324](#)
- processBlockPartition
 - MatrixType, [221](#)
- processBlocks
 - MatrixType, [221](#)
- processConstraints
 - LindoSolver, [176](#)
- processNonlinearExpressions
 - LindoSolver, [177](#)
- processQuadraticTerms
 - LindoSolver, [177](#)
- processVariables
 - LindoSolver, [176](#)
- Processes, [640](#)
 - addProcess, [643](#)
 - deepCopyFrom, [643](#)
 - setProcess, [643](#)
 - setRandom, [642](#)
- ProductCone, [643](#)
 - deepCopyFrom, [645](#)
 - getConeInXML, [644](#)
 - getConeName, [644](#)
 - numberOfOtherIndexes, [645](#)
 - setRandom, [645](#)
- qtermcount
 - OSILParserData, [279](#)
- QuadraticCoefficients, [645](#)
- QuadraticCone, [646](#)
 - axisDirection, [648](#)
 - deepCopyFrom, [648](#)
 - getConeInXML, [647](#)
 - getConeName, [647](#)
 - normScaleFactor, [648](#)
 - numberOfOtherIndexes, [648](#)
 - setRandom, [647](#)
- QuadraticTerm, [649](#)
- QuadraticTerms, [649](#)
 - rowIndexes, [650](#)
- quit
 - OSCommandLine, [268](#)
- readCommandLine
 - OSCommandLineReader, [269](#)
- readNI
 - OSnl2OS, [350](#)
- readOSiL
 - OSiLReader, [280](#)
- readOSoL
 - OSoLReader, [441](#)
- readOSrL
 - OSrLReader, [584](#)
- retrieve
 - OSSolverAgent, [594](#)
 - OShL, [275](#)
- reverseAD
 - OSInstance, [330](#)
- RotatedQuadraticCone, [650](#)
 - deepCopyFrom, [652](#)
 - firstAxisDirection, [652](#)
 - getConeInXML, [651](#)
 - getConeName, [651](#)
 - normScaleFactor, [652](#)
 - numberOfOtherIndexes, [652](#)
 - setRandom, [652](#)
- rowIndexes
 - QuadraticTerms, [650](#)
- runSolver
 - OSRunSolver.h, [741](#), [742](#)
- SOAPify
 - WSUtil, [707](#)
- SOSVariableBranchingWeights, [665](#)
 - addSOS, [667](#)
 - deepCopyFrom, [666](#)
 - setRandom, [666](#)
 - setSOS, [667](#)
- SOSWeights, [667](#)
 - addVar, [669](#)
 - deepCopyFrom, [669](#)
 - setRandom, [668](#)
 - setVar, [669](#)
- sSolverName
 - DefaultSolver, [87](#)
- ScalarExpressionTree, [653](#)
 - calculateFunction, [654](#)
 - getPostfixFromExpressionTree, [654](#)
 - getPrefixFromExpressionTree, [654](#)
 - getVariableIndicesMap, [654](#)
- selectSolver
 - OSRunSolver.h, [742](#)
- SemidefiniteCone, [655](#)
 - deepCopyFrom, [656](#)
 - getConeInXML, [656](#)
 - getConeName, [656](#)
 - numberOfOtherIndexes, [656](#)
 - setRandom, [656](#)
- send
 - OSSolverAgent, [593](#)
 - OShL, [275](#)
- sendSOAPMessage
 - WSUtil, [707](#)
- serviceMethod
 - OSCommandLine, [267](#)
 - osOptionsStruc, [477](#)
- ServiceOption, [657](#)
 - deepCopyFrom, [658](#)
 - setRandom, [658](#)

- ServiceResult, [658](#)
 - setRandom, [659](#)
- setASL
 - OSnl2OS, [350](#)
- setActualStartTime
 - OSResult, [525](#)
- setAllPrintLevels
 - OSOutputChannel, [483](#)
- setAnOtherGeneralOption
 - OSOption, [468](#)
- setAnOtherSolutionResult
 - OSResult, [575](#)
- setAnOtherVariableResultDense
 - OSResult, [541](#), [542](#)
- setAnOtherVariableResultSparse
 - OSResult, [539](#), [541](#)
- setAnotherInitBasisStatus
 - OSOption, [469](#)
- setAvailableCPUNumberDescription
 - OSResult, [521](#)
- setAvailableCPUNumberValue
 - OSResult, [521](#)
- setAvailableCPUSpeedDescription
 - OSResult, [520](#)
- setAvailableCPUSpeedUnit
 - OSResult, [520](#)
- setAvailableCPUSpeedValue
 - OSResult, [521](#)
- setAvailableDiskSpaceDescription
 - OSResult, [519](#)
- setAvailableDiskSpaceUnit
 - OSResult, [519](#)
- setAvailableDiskSpaceValue
 - OSResult, [519](#)
- setAvailableMemoryDescription
 - OSResult, [520](#)
- setAvailableMemoryUnit
 - OSResult, [519](#)
- setAvailableMemoryValue
 - OSResult, [520](#)
- setBasisStatus
 - OSResult, [538](#)
- setCoinPackedMatrix
 - CoinSolver, [55](#)
- setCon
 - InitConstraintValues, [123](#)
 - InitDualVariableValues, [127](#)
 - OtherConstraintOption, [599](#)
- setConeNumber
 - OSInstance, [318](#)
- setConstraintNumber
 - OSInstance, [314](#)
 - OSResult, [531](#)
- setConstraintValuesDense
 - OSResult, [561](#)
- setConstraints
 - OSInstance, [314](#)
- setCurrentJobCount
 - OSResult, [523](#)
- setCurrentState
 - OSResult, [522](#)
- setDualValue
 - OSResult, [561](#)
- setDualVariableValuesDense
 - OSResult, [561](#)
- setDualVariableValuesSparse
 - OSResult, [560](#)
- setGeneralMessage
 - OSResult, [516](#)
- setGeneralStatus
 - OSResult, [514](#)
- setGeneralStatusDescription
 - OSResult, [515](#)
- setGeneralStatusType
 - OSResult, [514](#)
- setGeneralSubstatusDescription
 - OSResult, [515](#)
- setGeneralSubstatusName
 - OSResult, [515](#)
- setHeader
 - GeneralFileHeader, [106](#)
 - OSOption, [451](#)
 - OSResult, [499](#)
- setIBVar
 - OSnl2OS, [351](#)
- setInstanceCreator
 - OSInstance, [311](#)
- setInstanceDescription
 - OSInstance, [311](#)
- setInstanceLicence
 - OSInstance, [311](#)
- setInstanceName
 - OSInstance, [310](#)
 - OSResult, [516](#)
- setInstanceSource
 - OSInstance, [310](#)
- setIntVector
 - BasisStatus, [46](#)
 - IntVector, [162](#)
- setJobEndTime
 - OSResult, [526](#)
- setJobID
 - JobDependencies, [167](#)
 - OSResult, [517](#)
- setJobStatus
 - OSResult, [525](#)
- setJobSubmitTime
 - OSResult, [525](#)

- setLinearConstraintCoefficients
 - OSInstance, [315](#)
- setMatrix
 - OSMatrix, [339](#)
- setMatrixNumber
 - OSInstance, [317](#)
- setMatrixVarValuesAttributes
 - OSResult, [569](#)
- setMatrixVarValuesBlockElements
 - OSResult, [570](#)
- setMatrixVarValuesBlockStructure
 - OSResult, [569](#)
- setMatrixVariableSolution
 - OSResult, [568](#)
- setMatrixVariablesOtherResultBlockElements
 - OSResult, [572](#)
- setMatrixVariablesOtherResultBlockStructure
 - OSResult, [572](#)
- setMatrixVariablesOtherResultGeneralAttributes
 - OSResult, [570](#)
- setMatrixVariablesOtherResultMatrixAttributes
 - OSResult, [571](#)
- setMinCPUNumber
 - OSOption, [469](#)
- setMinCPUSpeed
 - OSOption, [469](#)
- setMinDiskSpace
 - OSOption, [468](#)
- setMinMemorySize
 - OSOption, [468](#)
- setNonlinearExpressions
 - OSInstance, [317](#)
- setNumberOfDualValues
 - OSResult, [560](#)
- setNumberOfDualVariableValues
 - OSResult, [560](#)
- setNumberOfGeneralSubstatuses
 - OSResult, [515](#)
- setNumberOfObjValues
 - OSResult, [550](#)
- setNumberOfObjectiveValues
 - OSResult, [551](#)
- setNumberOfOtherConstraintResults
 - OSResult, [559](#)
- setNumberOfOtherGeneralResults
 - OSResult, [517](#)
- setNumberOfOtherJobResults
 - OSResult, [530](#)
- setNumberOfOtherObjectiveResults
 - OSResult, [550](#)
- setNumberOfOtherServiceResults
 - OSResult, [524](#)
- setNumberOfOtherSolutionResults
 - OSResult, [573](#)
- setNumberOfOtherSystemResults
 - OSResult, [521](#)
- setNumberOfOtherVariableResults
 - OSResult, [539](#)
- setNumberOfPrimalVariableValues
 - OSResult, [535](#)
- setNumberOfQuadraticTerms
 - OSInstance, [316](#)
- setNumberOfSolutionSubstatuses
 - OSResult, [533](#)
- setNumberOfSolverOutputs
 - OSResult, [576](#)
- setNumberOfTimes
 - OSResult, [527](#)
- setNumberOfVarValues
 - OSResult, [536](#)
- setNumberOfVarValuesString
 - OSResult, [538](#)
- setObj
 - InitObjectiveBounds, [135](#), [136](#)
 - InitObjectiveValues, [139](#)
 - OtherObjectiveOption, [605](#)
- setObjValue
 - OSResult, [552](#)
- setObjectiveNumber
 - OSInstance, [313](#)
 - OSResult, [531](#)
- setObjectiveValuesDense
 - OSResult, [551](#)
- setObjectiveValuesSparse
 - OSResult, [551](#)
- setObjectives
 - OSInstance, [313](#)
- setOther
 - ConstraintOption, [74](#)
 - ObjectiveOption, [245](#)
 - OtherOptions, [616](#)
 - VariableOption, [693](#)
- setOtherConstraintOptionAttributes
 - OSOption, [473](#)
- setOtherConstraintOptionCon
 - OSOption, [474](#)
- setOtherConstraintResultCategory
 - OSResult, [567](#)
- setOtherConstraintResultCon
 - OSResult, [568](#)
- setOtherConstraintResultConIdx
 - OSResult, [567](#)
- setOtherConstraintResultConName
 - OSResult, [567](#)
- setOtherConstraintResultConType
 - OSResult, [564](#)
- setOtherConstraintResultDescription
 - OSResult, [566](#)

- setOtherConstraintResultEnumType
 - OSResult, [565](#)
- setOtherConstraintResultName
 - OSResult, [564](#)
- setOtherConstraintResultNumberOfCon
 - OSResult, [563](#)
- setOtherConstraintResultNumberOfEnumerations
 - OSResult, [563](#)
- setOtherConstraintResultSolver
 - OSResult, [566](#)
- setOtherConstraintResultType
 - OSResult, [564](#)
- setOtherConstraintResultValue
 - OSResult, [565](#)
- setOtherGeneralOptions
 - OSOption, [468](#)
- setOtherGeneralResultDescription
 - OSResult, [518](#)
- setOtherGeneralResultName
 - OSResult, [518](#)
- setOtherGeneralResultValue
 - OSResult, [518](#)
- setOtherJobResultDescription
 - OSResult, [531](#)
- setOtherJobResultName
 - OSResult, [530](#)
- setOtherJobResultValue
 - OSResult, [530](#)
- setOtherObjectiveOptionAttributes
 - OSOption, [471](#)
- setOtherObjectiveOptionObj
 - OSOption, [473](#)
- setOtherObjectiveResultCategory
 - OSResult, [557](#)
- setOtherObjectiveResultDescription
 - OSResult, [555](#)
- setOtherObjectiveResultEnumType
 - OSResult, [555](#)
- setOtherObjectiveResultName
 - OSResult, [553](#)
- setOtherObjectiveResultNumberOfEnumerations
 - OSResult, [553](#)
- setOtherObjectiveResultNumberOfObj
 - OSResult, [552](#)
- setOtherObjectiveResultObj
 - OSResult, [559](#)
- setOtherObjectiveResultObjIdx
 - OSResult, [558](#)
- setOtherObjectiveResultObjName
 - OSResult, [558](#)
- setOtherObjectiveResultObjType
 - OSResult, [554](#)
- setOtherObjectiveResultSolver
 - OSResult, [557](#)
- setOtherObjectiveResultType
 - OSResult, [554](#)
- setOtherObjectiveResultValue
 - OSResult, [555](#)
- setOtherOptionOrResultEnumeration
 - OSOption, [471](#)
 - OSResult, [549](#)
 - OtherOptionOrResultEnumeration, [614](#)
- setOtherServiceResultDescription
 - OSResult, [524](#)
- setOtherServiceResultName
 - OSResult, [524](#)
- setOtherServiceResultValue
 - OSResult, [524](#)
- setOtherSolutionResultCategory
 - OSResult, [574](#)
- setOtherSolutionResultDescription
 - OSResult, [574](#)
- setOtherSolutionResultItem
 - OSResult, [575](#)
- setOtherSolutionResultName
 - OSResult, [573](#)
- setOtherSolutionResultNumberOfItems
 - OSResult, [575](#)
- setOtherSolutionResultValue
 - OSResult, [574](#)
- setOtherSystemResultDescription
 - OSResult, [522](#)
- setOtherSystemResultName
 - OSResult, [522](#)
- setOtherSystemResultValue
 - OSResult, [522](#)
- setOtherVariableOptionAttributes
 - OSOption, [469](#)
- setOtherVariableOptionVar
 - OSOption, [471](#)
- setOtherVariableResultCategory
 - OSResult, [546](#)
- setOtherVariableResultDescription
 - OSResult, [546](#)
- setOtherVariableResultEnumType
 - OSResult, [545](#)
- setOtherVariableResultName
 - OSResult, [543](#)
- setOtherVariableResultNumberOfEnumerations
 - OSResult, [543](#)
- setOtherVariableResultNumberOfVar
 - OSResult, [542](#)
- setOtherVariableResultSolver
 - OSResult, [546](#)
- setOtherVariableResultType
 - OSResult, [544](#)
- setOtherVariableResultValue
 - OSResult, [545](#)

- setOtherVariableResultVar
 - OSResult, [549](#)
- setOtherVariableResultVarIdx
 - OSResult, [548](#)
- setOtherVariableResultVarName
 - OSResult, [548](#)
- setOtherVariableResultVarType
 - OSResult, [544](#)
- setPath
 - DirectoriesAndFiles, [89](#)
- setPathPair
 - PathPairs, [636](#)
- setPathPairs
 - OSOption, [469](#)
- setPrimalVariableValuesDense
 - OSResult, [536](#)
- setPrimalVariableValuesSparse
 - OSResult, [535](#)
- SetPrintLevel
 - OSOutput, [479](#), [481](#)
- setPrintLevel
 - OSOutputChannel, [483](#)
- setProcess
 - Processes, [643](#)
- setQuadraticCoefficients
 - OSInstance, [316](#)
- setQuadraticTermsInNonlinearExpressions
 - OSInstance, [316](#)
- setRandom
 - BasisStatus, [45](#)
 - BranchingWeight, [53](#)
 - CPUNumber, [82](#)
 - CPU Speed, [84](#)
 - CompletelyPositiveMatricesCone, [57](#)
 - ConReferenceMatrixElement, [62](#)
 - ConReferenceMatrixElements, [65](#)
 - ConReferenceMatrixValues, [67](#)
 - Cone, [59](#)
 - Cones, [61](#)
 - ConstantMatrixElements, [70](#)
 - ConstantMatrixValues, [71](#)
 - ConstraintOption, [74](#)
 - ConstraintSolution, [77](#)
 - ContactOption, [78](#)
 - CopositiveMatricesCone, [79](#)
 - DirectoriesAndFiles, [89](#)
 - DualCone, [91](#)
 - DualVarValue, [94](#)
 - DualVariableValues, [93](#)
 - GeneralFileHeader, [105](#)
 - GeneralMatrixElements, [108](#)
 - GeneralMatrixValues, [109](#)
 - GeneralOption, [111](#)
 - GeneralResult, [113](#)
 - GeneralStatus, [117](#)
 - GeneralSubstatus, [118](#)
 - InitBasStatus, [120](#)
 - InitConValue, [125](#)
 - InitConstraintValues, [122](#)
 - InitDualVarValue, [129](#)
 - InitDualVariableValues, [127](#)
 - InitObjBound, [133](#)
 - InitObjValue, [141](#)
 - InitObjectiveBounds, [135](#)
 - InitObjectiveValues, [138](#)
 - InitVarValue, [149](#)
 - InitVarValueString, [151](#)
 - InitVariableValues, [143](#)
 - InitVariableValuesString, [146](#)
 - InitialBasisStatus, [131](#)
 - InstanceLocationOption, [154](#)
 - IntVector, [162](#)
 - IntegerVariableBranchingWeights, [155](#)
 - IntersectionCone, [159](#)
 - JobDependencies, [167](#)
 - JobOption, [170](#)
 - JobResult, [172](#)
 - LinearMatrixElement, [179](#)
 - LinearMatrixElementTerm, [183](#)
 - LinearMatrixElements, [182](#)
 - LinearMatrixValues, [184](#)
 - Matrices, [188](#)
 - MatrixBlock, [190](#)
 - MatrixBlocks, [195](#)
 - MatrixNode, [208](#)
 - MatrixProgramming, [212](#)
 - MatrixProgrammingSolution, [214](#)
 - MatrixTransformation, [216](#)
 - MatrixType, [224](#)
 - MixedRowReferenceMatrixElements, [236](#)
 - NonnegativeCone, [240](#)
 - NonpositiveCone, [241](#)
 - OSMatrix, [340](#)
 - OSMatrixWithMatrixConIdx, [342](#)
 - OSMatrixWithMatrixObjIdx, [343](#)
 - OSMatrixWithMatrixVarIdx, [345](#)
 - OSOption, [452](#)
 - OSResult, [500](#)
 - ObjReferenceMatrixElements, [251](#)
 - ObjReferenceMatrixValues, [252](#)
 - ObjValue, [254](#)
 - ObjectiveOption, [245](#)
 - ObjectiveSolution, [247](#)
 - ObjectiveValues, [249](#)
 - OptimizationOption, [256](#)
 - OptimizationResult, [258](#)
 - OptimizationSolution, [259](#)
 - OptimizationSolutionStatus, [261](#)

- OptimizationSolutionSubstatus, [262](#)
- OrthantCone, [263](#)
- OtherConOption, [596](#)
- OtherConResult, [597](#)
- OtherConstraintOption, [599](#)
- OtherConstraintResult, [601](#)
- OtherObjOption, [608](#)
- OtherObjResult, [610](#)
- OtherObjectiveOption, [604](#)
- OtherObjectiveResult, [607](#)
- OtherOption, [611](#)
- OtherOptionOrResultEnumeration, [613](#)
- OtherOptions, [615](#)
- OtherResult, [617](#)
- OtherResults, [619](#)
- OtherSolutionResult, [620](#)
- OtherSolutionResults, [621](#)
- OtherSolverOutput, [623](#)
- OtherVarOption, [629](#)
- OtherVarResult, [631](#)
- OtherVariableOption, [624](#)
- OtherVariableResult, [626](#)
- PathPair, [633](#)
- PathPairs, [635](#)
- PolarCone, [638](#)
- PolyhedralCone, [640](#)
- Processes, [642](#)
- ProductCone, [645](#)
- QuadraticCone, [647](#)
- RotatedQuadraticCone, [652](#)
- SOSVariableBranchingWeights, [666](#)
- SOSWeights, [668](#)
- SemidefiniteCone, [656](#)
- ServiceOption, [658](#)
- ServiceResult, [659](#)
- SolverOption, [661](#)
- SolverOptions, [663](#)
- SolverOutput, [665](#)
- StorageCapacity, [676](#)
- SystemOption, [678](#)
- SystemResult, [680](#)
- TimeMeasurement, [687](#)
- TimeSpan, [688](#)
- TimingInformation, [690](#)
- VarReferenceMatrixElements, [700](#)
- VarReferenceMatrixValues, [701](#)
- VarValue, [703](#)
- VarValueString, [704](#)
- VariableOption, [693](#)
- VariableSolution, [695](#)
- VariableValues, [697](#)
- VariableValuesString, [698](#)
- setSOS
 - SOSVariableBranchingWeights, [667](#)
- setScheduledStartTime
 - OSResult, [525](#)
- setServiceName
 - OSResult, [516](#)
- setServiceURI
 - OSResult, [516](#)
- setServiceUtilization
 - OSResult, [523](#)
- setSolutionMessage
 - OSResult, [535](#)
- setSolutionNumber
 - OSResult, [532](#)
- setSolutionStatus
 - OSResult, [532](#)
- setSolutionStatusDescription
 - OSResult, [533](#)
- setSolutionStatusType
 - OSResult, [532](#)
- setSolutionSubstatusDescription
 - OSResult, [533](#)
- setSolutionSubstatusType
 - OSResult, [533](#)
- setSolutionTargetObjectiveIdx
 - OSResult, [534](#)
- setSolutionTargetObjectiveName
 - OSResult, [534](#)
- setSolutionWeightedObjectives
 - OSResult, [534](#)
- setSolverInvoked
 - OSResult, [517](#)
- setSolverOptionContent
 - OSOption, [474](#)
- setSolverOptions
 - BonminSolver, [52](#)
 - CoinSolver, [55](#)
 - CouenneSolver, [81](#)
 - CsdpSolver, [86](#)
 - IloptSolver, [166](#)
 - KnitroSolver, [174](#)
 - LindoSolver, [176](#)
 - SolverOptions, [663](#)
- setSolverOutputCategory
 - OSResult, [576](#)
- setSolverOutputDescription
 - OSResult, [576](#)
- setSolverOutputItem
 - OSResult, [578](#)
- setSolverOutputName
 - OSResult, [576](#)
- setSolverOutputNumberOfItems
 - OSResult, [578](#)
- setSystemInformation
 - OSResult, [518](#)
- setTime

- OSResult, [526](#)
- setTimeDomainStageConstraintsOrdered
 - OSInstance, [334](#)
- setTimeDomainStageConstraintsUnordered
 - OSInstance, [334](#)
- setTimeDomainStageObjectivesOrdered
 - OSInstance, [334](#)
- setTimeDomainStageObjectivesUnordered
 - OSInstance, [334](#)
- setTimeDomainStageVariablesOrdered
 - OSInstance, [333](#)
- setTimeDomainStageVariablesUnordered
 - OSInstance, [333](#)
- setTimeNumber
 - OSResult, [527](#)
- setTimeServiceStarted
 - OSResult, [523](#)
- setTimeStamp
 - OSResult, [517](#)
- setTimingInformation
 - OSResult, [526](#)
- setTotalJobsSoFar
 - OSResult, [523](#)
- setUsedCPUNumberDescription
 - OSResult, [529](#)
- setUsedCPUNumberValue
 - OSResult, [530](#)
- setUsedCPUSpeedDescription
 - OSResult, [529](#)
- setUsedCPUSpeedUnit
 - OSResult, [529](#)
- setUsedCPUSpeedValue
 - OSResult, [529](#)
- setUsedDiskSpaceDescription
 - OSResult, [527](#)
- setUsedDiskSpaceUnit
 - OSResult, [527](#)
- setUsedDiskSpaceValue
 - OSResult, [528](#)
- setUsedMemoryDescription
 - OSResult, [528](#)
- setUsedMemoryUnit
 - OSResult, [528](#)
- setUsedMemoryValue
 - OSResult, [528](#)
- setVar
 - InitVariableValues, [143](#), [144](#)
 - InitVariableValuesString, [147](#)
 - InitialBasisStatus, [131](#)
 - IntegerVariableBranchingWeights, [156](#)
 - OSnl2OS, [351](#)
 - OtherVariableOption, [625](#)
 - SOSWeights, [669](#)
- setVarValue
 - OSResult, [536](#)
- setVarValueString
 - OSResult, [538](#)
- setVariableNumber
 - OSInstance, [311](#)
 - OSResult, [531](#)
- setVariables
 - OSInstance, [312](#)
- shape
 - MatrixExpression, [202](#)
 - MatrixTransformation, [217](#)
 - NI, [238](#)
- solve
 - CoinSolver, [55](#)
 - OSMatlab, [336](#)
 - OSSolverAgent, [593](#)
 - OShL, [274](#)
- solverName
 - osOptionsStruc, [477](#)
- SolverOption, [660](#)
 - deepCopyFrom, [661](#)
 - setRandom, [661](#)
- SolverOptions, [662](#)
 - addSolverOption, [663](#)
 - deepCopyFrom, [663](#)
 - setRandom, [663](#)
 - setSolverOptions, [663](#)
- SolverOutput, [664](#)
 - setRandom, [665](#)
- SparseHessianMatrix, [669](#)
 - SparseHessianMatrix, [670](#)
- SparseIntVector, [670](#)
 - indexes, [671](#)
 - SparseIntVector, [671](#)
- SparseJacobianMatrix, [672](#)
 - SparseJacobianMatrix, [672](#)
- SparseMatrix, [673](#)
 - display, [674](#)
 - indexes, [674](#)
 - isColumnMajor, [674](#)
 - SparseMatrix, [674](#)
- SparseVector, [674](#)
 - SparseVector, [675](#)
- stageVariableStartIdx
 - OSILParserData, [279](#)
- stageVariablesON
 - OSILParserData, [279](#)
- stageVariablesOrdered
 - OSILParserData, [279](#)
- StorageCapacity, [675](#)
 - deepCopyFrom, [676](#)
 - setRandom, [676](#)
- symmetry
 - GeneralSparseMatrix, [115](#)

- MatrixType, 225
- SystemOption, 677
 - deepCopyFrom, 678
 - setRandom, 678
- SystemResult, 679
 - setRandom, 680
- takeOverOSInstance
 - OSgams2osil, 272
- TimeDomain, 680
- TimeDomainInterval, 681
- TimeDomainStage, 681
- TimeDomainStageCon, 682
- TimeDomainStageConstraints, 682
- TimeDomainStageObj, 683
- TimeDomainStageObjectives, 684
- TimeDomainStageVar, 685
- TimeDomainStageVariables, 686
- TimeDomainStages, 684
- timeDomainStages
 - OSILParserData, 279
- TimeMeasurement, 686
 - setRandom, 687
- TimeSpan, 687
 - deepCopyFrom, 689
 - setRandom, 688
- TimingInformation, 689
 - setRandom, 690
- tmpnlcount
 - OSnLPParserData, 438
- type
 - MatrixType, 225
 - Variable, 691
- ub
 - Variable, 691
- usedMemory
 - JobResult, 173
- vType
 - GeneralSparseMatrix, 115
- value
 - GeneralSparseMatrix, 115
 - LinearMatrixElements, 182
 - MixedRowReferenceMatrixElements, 236
- valueType
 - ConReferenceMatrixElement, 63
- VarReferenceMatrixElements, 698
 - alignsOnBlockBoundary, 700
 - cloneMatrixNode, 700
 - deepCopyFrom, 700
 - getMatrixNodeInXML, 699
 - getMatrixType, 699
 - getNodeName, 699
 - getNodeType, 699
 - setRandom, 700
- VarReferenceMatrixValues, 701
 - deepCopyFrom, 702
 - setRandom, 701
- varType
 - MatrixVar, 226
- VarValue, 702
 - setRandom, 703
- VarValueString, 703
 - setRandom, 704
- Variable, 690
 - lb, 691
 - type, 691
 - ub, 691
- VariableOption, 691
 - addOther, 693
 - deepCopyFrom, 693
 - setOther, 693
 - setRandom, 693
- VariableSolution, 694
 - setRandom, 695
- VariableValues, 696
 - setRandom, 697
- VariableValuesString, 697
 - setRandom, 698
- Variables, 694
- WSUtil, 705
 - createFormDataUpload, 708
 - createSOAPMessage, 707
 - deSOAPify, 707
 - getOSxL, 708
 - SOAPify, 707
 - sendSOAPMessage, 707
 - WSUtil, 706
- walkTree
 - OSnl2OS, 351
- writeBasisStatus
 - OSgLWriter.h, 718
- writeDbfVectorData
 - OSgLWriter.h, 718
- writeFileFromChar
 - FileUtil, 104
- writeFileFromString
 - FileUtil, 103
- writeGeneralFileHeader
 - OSgLWriter.h, 717
- writeIntVectorData
 - OSgLWriter.h, 717
- writeOSiL
 - OSiLWriter, 281
- writeOSoL
 - OSoLWriter, 442
- writeOSrL

- OSrLWriter, [585](#)
- writeOtherOptionOrResultEnumeration
 - OSgLWriter.h, [718](#)
- writeSolFile
 - OSosl2ampl, [478](#)
- writeSolution
 - OSrL2Gams, [579](#)
- writeStringData
 - OSStringUtil.h, [744](#)
- YYLTYPE, [709](#)
- YYSTYPE, [709](#)