

## Macsur Adapter

Generated by Doxygen 1.8.3.1

Tue Jun 4 2013 12:58:10



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	Ui Namespace Reference . . . . .	9
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
6.1	MadData Class Reference . . . . .	11
6.1.1	Detailed Description . . . . .	12
6.1.2	Constructor & Destructor Documentation . . . . .	12
6.1.2.1	MadData . . . . .	12
6.1.2.2	MadData . . . . .	13
6.1.3	Member Function Documentation . . . . .	13
6.1.3.1	description . . . . .	13
6.1.3.2	fromXml . . . . .	14
6.1.3.3	fromXmlFile . . . . .	15
6.1.3.4	guid . . . . .	16
6.1.3.5	imageFile . . . . .	16
6.1.3.6	name . . . . .	16
6.1.3.7	operator= . . . . .	17
6.1.3.8	setDescription . . . . .	18
6.1.3.9	setGuid . . . . .	18
6.1.3.10	setImageFile . . . . .	18
6.1.3.11	setName . . . . .	19

6.1.3.12	toHtml	19
6.1.3.13	toText	19
6.1.3.14	toXml	20
6.1.3.15	toXmlFile	21
6.2	MadDataClassification Class Reference	21
6.2.1	Detailed Description	22
6.2.2	Constructor & Destructor Documentation	22
6.2.2.1	MadDataClassification	22
6.2.3	Member Function Documentation	27
6.2.3.1	changeEvent	27
6.3	MadDataClassificationCultivation Class Reference	27
6.3.1	Detailed Description	29
6.3.2	Constructor & Destructor Documentation	29
6.3.2.1	MadDataClassificationCultivation	29
6.3.2.2	MadDataClassificationCultivation	29
6.3.3	Member Function Documentation	30
6.3.3.1	fertilisation	30
6.3.3.2	fromXml	31
6.3.3.3	fromXmlFile	32
6.3.3.4	guid	32
6.3.3.5	harvest	33
6.3.3.6	irrigation	33
6.3.3.7	operator=	33
6.3.3.8	seedDensity	34
6.3.3.9	setFertilisation	35
6.3.3.10	setGuid	35
6.3.3.11	setHarvest	36
6.3.3.12	setIrrigation	36
6.3.3.13	setSeedDensity	37
6.3.3.14	setSowing	37
6.3.3.15	setTillage	37
6.3.3.16	setVariety	38
6.3.3.17	setYield	38
6.3.3.18	sowing	38
6.3.3.19	tillage	39
6.3.3.20	toHtml	39
6.3.3.21	toText	40
6.3.3.22	toXml	40
6.3.3.23	toXmlFile	41
6.3.3.24	variety	42

6.3.3.25	yield	42
6.4	MadDataClassificationInitialValues Class Reference	43
6.4.1	Detailed Description	44
6.4.2	Constructor & Destructor Documentation	44
6.4.2.1	MadDataClassificationInitialValues	44
6.4.2.2	MadDataClassificationInitialValues	45
6.4.3	Member Function Documentation	45
6.4.3.1	fromXml	46
6.4.3.2	fromXmlFile	46
6.4.3.3	guid	47
6.4.3.4	nitrogenMin	47
6.4.3.5	operator=	48
6.4.3.6	setGuid	48
6.4.3.7	setNitrogenMin	49
6.4.3.8	setSoilMoisture	49
6.4.3.9	soilMoisture	50
6.4.3.10	toHtml	50
6.4.3.11	toText	51
6.4.3.12	toXml	51
6.4.3.13	toXmlFile	52
6.5	MadDataClassificationPhenology Class Reference	53
6.5.1	Detailed Description	54
6.5.2	Constructor & Destructor Documentation	55
6.5.2.1	MadDataClassificationPhenology	55
6.5.2.2	MadDataClassificationPhenology	55
6.5.3	Member Function Documentation	56
6.5.3.1	earEmergence	56
6.5.3.2	emergence	57
6.5.3.3	flowering	57
6.5.3.4	fromXml	58
6.5.3.5	fromXmlFile	58
6.5.3.6	guid	59
6.5.3.7	operator=	59
6.5.3.8	setEarEmergence	60
6.5.3.9	setEmergence	60
6.5.3.10	setFlowering	61
6.5.3.11	setGuid	61
6.5.3.12	setStemElongation	62
6.5.3.13	setYellowRipeness	62
6.5.3.14	stemElongation	63

6.5.3.15	toHtml	63
6.5.3.16	toText	63
6.5.3.17	toXml	64
6.5.3.18	toXmlFile	65
6.5.3.19	yellowRipeness	66
6.6	MadDataClassificationPrevCrop Class Reference	66
6.6.1	Detailed Description	67
6.6.2	Constructor & Destructor Documentation	68
6.6.2.1	MadDataClassificationPrevCrop	68
6.6.2.2	MadDataClassificationPrevCrop	68
6.6.3	Member Function Documentation	69
6.6.3.1	crop	69
6.6.3.2	fertilisation	70
6.6.3.3	fromXml	70
6.6.3.4	fromXmlFile	71
6.6.3.5	guid	72
6.6.3.6	harvestDate	72
6.6.3.7	irrigation	72
6.6.3.8	operator=	73
6.6.3.9	residueMgmt	74
6.6.3.10	setCrop	75
6.6.3.11	setFertilisation	75
6.6.3.12	setGuid	76
6.6.3.13	setHarvestDate	76
6.6.3.14	setIrrigation	77
6.6.3.15	setResidueMgmt	77
6.6.3.16	setSowingDate	77
6.6.3.17	setYield	78
6.6.3.18	sowingDate	78
6.6.3.19	toHtml	78
6.6.3.20	toText	79
6.6.3.21	toXml	79
6.6.3.22	toXmlFile	80
6.6.3.23	yield	81
6.7	MadDataClassificationSiteData Class Reference	82
6.7.1	Detailed Description	83
6.7.2	Constructor & Destructor Documentation	83
6.7.2.1	MadDataClassificationSiteData	83
6.7.2.2	MadDataClassificationSiteData	83
6.7.3	Member Function Documentation	84

6.7.3.1	altitude	84
6.7.3.2	fromXml	85
6.7.3.3	fromXmlFile	85
6.7.3.4	guid	86
6.7.3.5	latitude	86
6.7.3.6	longitude	87
6.7.3.7	operator=	87
6.7.3.8	setAltitude	88
6.7.3.9	setGuid	88
6.7.3.10	setLatitude	89
6.7.3.11	setLongitude	89
6.7.3.12	toHtml	90
6.7.3.13	toText	90
6.7.3.14	toXml	91
6.7.3.15	toXmlFile	92
6.8	MadDataClassificationSoil Class Reference	93
6.8.1	Detailed Description	94
6.8.2	Constructor & Destructor Documentation	94
6.8.2.1	MadDataClassificationSoil	94
6.8.2.2	MadDataClassificationSoil	95
6.8.3	Member Function Documentation	96
6.8.3.1	bulkDensity	96
6.8.3.2	carbonOrganic	97
6.8.3.3	fieldCapacityMeas	97
6.8.3.4	fromXml	98
6.8.3.5	fromXmlFile	99
6.8.3.6	guid	99
6.8.3.7	hydrCondCurve	100
6.8.3.8	nitrogenOrganic	100
6.8.3.9	operator=	100
6.8.3.10	pfCurve	101
6.8.3.11	pH	102
6.8.3.12	setBulkDensity	102
6.8.3.13	setCarbonOrganic	103
6.8.3.14	setFieldCapacityMeas	103
6.8.3.15	setGuid	104
6.8.3.16	setHydrCondCurve	104
6.8.3.17	setNitrogenOrganic	105
6.8.3.18	setPfCurve	105
6.8.3.19	setPh	105

6.8.3.20	<a href="#">setTexture</a>	106
6.8.3.21	<a href="#">setWiltingPointMeas</a>	106
6.8.3.22	<a href="#">texture</a>	107
6.8.3.23	<a href="#">toHtml</a>	107
6.8.3.24	<a href="#">toText</a>	107
6.8.3.25	<a href="#">toXml</a>	108
6.8.3.26	<a href="#">toXmlFile</a>	109
6.8.3.27	<a href="#">wiltingPointMeas</a>	110
6.9	<a href="#">MadDataClassificationWeather Class Reference</a>	110
6.9.1	<a href="#">Detailed Description</a>	112
6.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	112
6.9.2.1	<a href="#">MadDataClassificationWeather</a>	112
6.9.2.2	<a href="#">MadDataClassificationWeather</a>	112
6.9.3	<a href="#">Member Function Documentation</a>	114
6.9.3.1	<a href="#">fromXml</a>	114
6.9.3.2	<a href="#">fromXmlFile</a>	115
6.9.3.3	<a href="#">globalRadiation</a>	115
6.9.3.4	<a href="#">guid</a>	116
6.9.3.5	<a href="#">leafWetness</a>	116
6.9.3.6	<a href="#">minData</a>	117
6.9.3.7	<a href="#">operator=</a>	117
6.9.3.8	<a href="#">precipitation</a>	118
6.9.3.9	<a href="#">relativeHumidity</a>	119
6.9.3.10	<a href="#">setGlobalRadiation</a>	119
6.9.3.11	<a href="#">setGuid</a>	120
6.9.3.12	<a href="#">setLeafWetness</a>	120
6.9.3.13	<a href="#">setMinData</a>	121
6.9.3.14	<a href="#">setPrecipitation</a>	121
6.9.3.15	<a href="#">setRelativeHumidity</a>	121
6.9.3.16	<a href="#">setSoilTemp</a>	121
6.9.3.17	<a href="#">setSunshineHours</a>	122
6.9.3.18	<a href="#">setTAve</a>	122
6.9.3.19	<a href="#">setTMax</a>	122
6.9.3.20	<a href="#">setTMin</a>	123
6.9.3.21	<a href="#">setWindSpeed</a>	123
6.9.3.22	<a href="#">soilTemp</a>	124
6.9.3.23	<a href="#">sunshineHours</a>	124
6.9.3.24	<a href="#">tAve</a>	124
6.9.3.25	<a href="#">tMax</a>	125
6.9.3.26	<a href="#">tMin</a>	125



6.9.3.27	toHtml	126
6.9.3.28	toText	126
6.9.3.29	toXml	127
6.9.3.30	toXmlFile	128
6.9.3.31	windSpeed	129
6.10	MadDataset Class Reference	129
6.10.1	Detailed Description	131
6.10.2	Constructor & Destructor Documentation	131
6.10.2.1	MadDataset	131
6.10.2.2	MadDataset	131
6.10.3	Member Function Documentation	132
6.10.3.1	cultivation	133
6.10.3.2	description	133
6.10.3.3	fromXml	133
6.10.3.4	fromXmlFile	134
6.10.3.5	guid	135
6.10.3.6	initialValues	135
6.10.3.7	name	136
6.10.3.8	operator=	136
6.10.3.9	phenology	137
6.10.3.10	prevCrop	138
6.10.3.11	setCultivation	138
6.10.3.12	setDescription	138
6.10.3.13	setGuid	139
6.10.3.14	setInitialValues	139
6.10.3.15	setName	139
6.10.3.16	setPhenology	140
6.10.3.17	setPrevCrop	140
6.10.3.18	setSiteData	140
6.10.3.19	setSoil	140
6.10.3.20	setStateVars	140
6.10.3.21	setWeather	140
6.10.3.22	siteData	141
6.10.3.23	soil	141
6.10.3.24	stateVars	141
6.10.3.25	toHtml	142
6.10.3.26	toText	142
6.10.3.27	toXml	143
6.10.3.28	toXmlFile	144
6.10.3.29	weather	145

6.11 MadGuid Class Reference . . . . .	145
6.11.1 Detailed Description . . . . .	147
6.11.2 Constructor & Destructor Documentation . . . . .	147
6.11.2.1 MadGuid . . . . .	147
6.11.3 Member Function Documentation . . . . .	147
6.11.3.1 guid . . . . .	147
6.11.3.2 setGuid . . . . .	147
6.12 MadMainWindow Class Reference . . . . .	148
6.12.1 Detailed Description . . . . .	149
6.12.2 Constructor & Destructor Documentation . . . . .	149
6.12.2.1 MadMainWindow . . . . .	149
6.12.3 Member Function Documentation . . . . .	150
6.12.3.1 changeEvent . . . . .	150
6.12.3.2 modelText . . . . .	150
6.12.3.3 setModelText . . . . .	150
6.13 MadModel Class Reference . . . . .	150
6.13.1 Detailed Description . . . . .	152
6.13.2 Constructor & Destructor Documentation . . . . .	152
6.13.2.1 MadModel . . . . .	152
6.13.2.2 MadModel . . . . .	152
6.13.3 Member Function Documentation . . . . .	153
6.13.3.1 description . . . . .	153
6.13.3.2 fromXml . . . . .	153
6.13.3.3 fromXmlFile . . . . .	154
6.13.3.4 guid . . . . .	155
6.13.3.5 imageFile . . . . .	155
6.13.3.6 name . . . . .	156
6.13.3.7 operator= . . . . .	156
6.13.3.8 setDescription . . . . .	157
6.13.3.9 setGuid . . . . .	157
6.13.3.10 setImageFile . . . . .	158
6.13.3.11 setName . . . . .	158
6.13.3.12 toHtml . . . . .	159
6.13.3.13 toText . . . . .	159
6.13.3.14 toXml . . . . .	160
6.13.3.15 toXmlFile . . . . .	161
6.14 MadSerialisable Class Reference . . . . .	161
6.14.1 Detailed Description . . . . .	163
6.14.2 Constructor & Destructor Documentation . . . . .	163
6.14.2.1 MadSerialisable . . . . .	163

6.14.3	Member Function Documentation	163
6.14.3.1	fromXml	163
6.14.3.2	fromXmlFile	164
6.14.3.3	toXml	165
6.14.3.4	toXmlFile	165
6.15	MadStateVars Class Reference	166
6.15.1	Detailed Description	167
6.15.2	Constructor & Destructor Documentation	168
6.15.2.1	MadStateVars	168
6.15.2.2	MadStateVars	168
6.15.3	Member Function Documentation	169
6.15.3.1	cropCategories	169
6.15.3.2	fromXml	170
6.15.3.3	fromXmlFile	170
6.15.3.4	guid	171
6.15.3.5	observationCategories	171
6.15.3.6	operator=	172
6.15.3.7	setCropCategories	172
6.15.3.8	setGuid	173
6.15.3.9	setObservationCategories	173
6.15.3.10	setSoilCategories	174
6.15.3.11	setSurfaceFluxesCategories	174
6.15.3.12	soilCategories	175
6.15.3.13	surfaceFluxesCategories	175
6.15.3.14	toHtml	175
6.15.3.15	toText	175
6.15.3.16	toXml	176
6.15.3.17	toXmlFile	177
6.16	MadSubCategory Class Reference	178
6.16.1	Detailed Description	179
6.16.2	Constructor & Destructor Documentation	180
6.16.2.1	MadSubCategory	180
6.16.2.2	MadSubCategory	180
6.16.3	Member Function Documentation	181
6.16.3.1	depth	181
6.16.3.2	fromXml	181
6.16.3.3	fromXmlFile	182
6.16.3.4	guid	183
6.16.3.5	minData	183
6.16.3.6	observations	184

6.16.3.7	operator=	184
6.16.3.8	replicates	185
6.16.3.9	setDepth	185
6.16.3.10	setGuid	186
6.16.3.11	setMinData	186
6.16.3.12	setObservations	187
6.16.3.13	setReplicates	187
6.16.3.14	setWeightPoints	187
6.16.3.15	toHtml	187
6.16.3.16	toText	188
6.16.3.17	toXml	188
6.16.3.18	toXmlFile	189
6.16.3.19	weightPoints	190
6.17	MadSVCrop Class Reference	190
6.17.1	Detailed Description	192
6.17.2	Constructor & Destructor Documentation	192
6.17.2.1	MadSVCrop	192
6.17.2.2	MadSVCrop	193
6.17.3	Member Function Documentation	194
6.17.3.1	agrBiomass	194
6.17.3.2	fromXml	194
6.17.3.3	fromXmlFile	195
6.17.3.4	guid	196
6.17.3.5	lai	196
6.17.3.6	nInAGrBiomass	197
6.17.3.7	nInOrgans	197
6.17.3.8	operator=	198
6.17.3.9	rootBiomass	198
6.17.3.10	setAgrBiomass	199
6.17.3.11	setGuid	199
6.17.3.12	setLai	200
6.17.3.13	setNInAGrBiomass	200
6.17.3.14	setNInOrgans	201
6.17.3.15	setRootBiomass	201
6.17.3.16	setWeightOrgans	202
6.17.3.17	toHtml	202
6.17.3.18	toText	203
6.17.3.19	toXml	203
6.17.3.20	toXmlFile	204
6.17.3.21	weightOrgans	205

6.18 MadSVObservations Class Reference	206
6.18.1 Detailed Description	207
6.18.2 Constructor & Destructor Documentation	207
6.18.2.1 MadSVObservations	207
6.18.2.2 MadSVObservations	207
6.18.3 Member Function Documentation	208
6.18.3.1 damages	208
6.18.3.2 fromXml	209
6.18.3.3 fromXmlFile	210
6.18.3.4 guid	210
6.18.3.5 lodging	211
6.18.3.6 operator=	211
6.18.3.7 pestsOrDiseases	212
6.18.3.8 setDamages	212
6.18.3.9 setGuid	213
6.18.3.10 setLodging	214
6.18.3.11 setPestsOrDiseases	214
6.18.3.12 toHtml	214
6.18.3.13 toText	215
6.18.3.14 toXml	215
6.18.3.15 toXmlFile	216
6.19 MadSVSoil Class Reference	217
6.19.1 Detailed Description	218
6.19.2 Constructor & Destructor Documentation	219
6.19.2.1 MadSVSoil	219
6.19.2.2 MadSVSoil	219
6.19.3 Member Function Documentation	220
6.19.3.1 fromXml	220
6.19.3.2 fromXmlFile	221
6.19.3.3 guid	222
6.19.3.4 nitrogenFluxBottomRoot	222
6.19.3.5 nMin	223
6.19.3.6 operator=	223
6.19.3.7 pressureHeads	224
6.19.3.8 setGuid	224
6.19.3.9 setNitrogenFluxBottomRoot	225
6.19.3.10 setNMin	226
6.19.3.11 setPressureHeads	226
6.19.3.12 setSoilWaterGrav	226
6.19.3.13 setSoilWaterSensorCal	227

6.19.3.14	setWaterFluxBottomRoot	227
6.19.3.15	soilWaterGrav	227
6.19.3.16	soilWaterSensorCal	228
6.19.3.17	toHtml	228
6.19.3.18	toText	229
6.19.3.19	toXml	229
6.19.3.20	toXmlFile	230
6.19.3.21	waterFluxBottomRoot	231
6.20	MadSVSurfaceFluxes Class Reference	231
6.20.1	Detailed Description	233
6.20.2	Constructor & Destructor Documentation	233
6.20.2.1	MadSVSurfaceFluxes	233
6.20.2.2	MadSVSurfaceFluxes	234
6.20.3	Member Function Documentation	235
6.20.3.1	ch4Loss	235
6.20.3.2	et	236
6.20.3.3	fromXml	237
6.20.3.4	fromXmlFile	237
6.20.3.5	guid	238
6.20.3.6	n2Loss	238
6.20.3.7	n2oLoss	239
6.20.3.8	nh3Loss	239
6.20.3.9	operator=	240
6.20.3.10	setCh4Loss	241
6.20.3.11	setEt	242
6.20.3.12	setGuid	242
6.20.3.13	setN2Loss	243
6.20.3.14	setN2oLoss	243
6.20.3.15	setNh3Loss	244
6.20.3.16	toHtml	244
6.20.3.17	toText	245
6.20.3.18	toXml	245
6.20.3.19	toXmlFile	246
6.21	MadTextDisplayForm Class Reference	247
6.21.1	Detailed Description	248
6.21.2	Constructor & Destructor Documentation	248
6.21.2.1	MadTextDisplayForm	248
6.21.2.2	~MadTextDisplayForm	249
6.21.3	Member Function Documentation	249
6.21.3.1	setText	249

6.22	MadUtils Class Reference	249
6.22.1	Detailed Description	250
6.22.2	Member Typedef Documentation	250
6.22.2.1	ModelMap	250
6.22.3	Constructor & Destructor Documentation	250
6.22.3.1	MadUtils	250
6.22.4	Member Function Documentation	250
6.22.4.1	createTextFile	250
6.22.4.2	getAvailableModels	251
6.22.4.3	getModel	252
6.22.4.4	getModelOutputDir	252
6.22.4.5	getStandardCss	252
6.22.4.6	openGraphicFile	253
6.22.4.7	saveFile	253
6.22.4.8	sortList	254
6.22.4.9	uniqueList	254
6.22.4.10	userConversionTablesDirPath	254
6.22.4.11	userImagesDirPath	255
6.22.4.12	userModelParametersDirPath	255
6.22.4.13	userModelProfilesDirPath	256
6.22.4.14	userSettingsDirPath	256
6.22.4.15	xmlDecode	257
6.22.4.16	xmlEncode	257
6.23	QDialog Class Reference	259
6.24	QMainWindow Class Reference	259
<b>7</b>	<b>File Documentation</b>	<b>261</b>
7.1	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.cpp File Reference	261
7.1.1	Function Documentation	261
7.1.1.1	makeString	261
7.2	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.h File Reference	261
7.3	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.cpp File Reference	262
7.4	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.h File Reference	263
7.5	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.cpp File Reference	264
7.6	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.h File Reference	265

7.7	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.cpp	
	File Reference . . . . .	266
7.8	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.h	
	File Reference . . . . .	266
7.9	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.cpp	
	File Reference . . . . .	267
7.10	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.h	
	File Reference . . . . .	268
7.11	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.cpp	
	File Reference . . . . .	269
7.12	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.h	
	File Reference . . . . .	269
7.13	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationssitedata.cpp	
	File Reference . . . . .	270
7.14	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationssitedata.h	
	File Reference . . . . .	271
7.15	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.cpp	
	File Reference . . . . .	272
7.16	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.h	
	File Reference . . . . .	272
7.17	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.cpp	
	File Reference . . . . .	273
7.18	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.h	
	File Reference . . . . .	274
7.19	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.cpp	
	File Reference . . . . .	275
7.20	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.h	
	File Reference . . . . .	275
7.21	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.cpp	
	File Reference . . . . .	276
7.22	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.h	
	File Reference . . . . .	277
7.23	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.cpp	
	File Reference . . . . .	278
7.24	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.h	
	File Reference . . . . .	279
7.24.1	Detailed Description . . . . .	280
7.25	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.cpp	
	File Reference . . . . .	280
7.26	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.h	
	File Reference . . . . .	280
7.27	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.cpp	
	File Reference . . . . .	281
7.28	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.h	
	File Reference . . . . .	282
7.29	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.cpp	
	File Reference . . . . .	283



7.30	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.h File Reference	284
7.31	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/mad.h File Reference	286
7.31.1	Typedef Documentation	287
7.31.1.1	MadModelInfo	287
7.31.1.2	MadTripleMap	287
7.31.2	Enumeration Type Documentation	287
7.31.2.1	AreaUnits	287
7.31.2.2	DataClass	287
7.31.2.3	EnergyType	288
7.31.2.4	FileType	288
7.31.2.5	ModelTheme	288
7.31.2.6	Nuts	289
7.31.2.7	Scale	289
7.32	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.cpp File Reference	289
7.33	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.h File Reference	290
7.34	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.cpp File Reference	290
7.35	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.h File Reference	291
7.36	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.cpp File Reference	292
7.37	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.h File Reference	293
7.38	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.cpp File Reference	293
7.39	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.h File Reference	294
7.40	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.cpp File Reference	295
7.41	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.h File Reference	295
7.42	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.cpp File Reference	296
7.43	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.h File Reference	297
7.44	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madversion.h File Reference	298
7.44.1	Macro Definition Documentation	298
7.44.1.1	VERSION	298
7.45	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.cpp File Reference	298
7.46	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.h File Reference	299
7.47	/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/main.cpp File Reference	299
7.47.1	Function Documentation	300
7.47.1.1	bundleclicked	300
7.47.1.2	main	300



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Ui .....	9
----------	---



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

MadDataClassification	
MadDataClassification	21
MadGuid	145
MadData	11
MadDataClassificationCultivation	27
MadDataClassificationInitialValues	43
MadDataClassificationPhenology	53
MadDataClassificationPrevCrop	66
MadDataClassificationSiteData	82
MadDataClassificationSoil	93
MadDataClassificationWeather	110
MadDataset	129
MadModel	150
MadStateVars	166
MadSubCategory	178
MadSVCrop	190
MadSVObservations	206
MadSVSoil	217
MadSVSurfaceFluxes	231
MadMainWindow	
MadMainWindow	148
MadSerialisable	161
MadData	11
MadDataClassificationCultivation	27
MadDataClassificationInitialValues	43
MadDataClassificationPhenology	53
MadDataClassificationPrevCrop	66
MadDataClassificationSiteData	82
MadDataClassificationSoil	93
MadDataClassificationWeather	110
MadDataset	129
MadModel	150
MadStateVars	166
MadSubCategory	178
MadSVCrop	190
MadSVObservations	206
MadSVSoil	217

MadSVSurfaceFluxes . . . . .	231
MadUtils . . . . .	249
QDialog . . . . .	259
MadDataClassification . . . . .	21
MadTextDisplayForm . . . . .	247
QMainWindow . . . . .	259
MadMainWindow . . . . .	148

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">MadData</a>	
The <a href="#">MadData</a> class	11
<a href="#">MadDataClassification</a>	21
<a href="#">MadDataClassificationCultivation</a>	27
<a href="#">MadDataClassificationInitialValues</a>	43
<a href="#">MadDataClassificationPhenology</a>	53
<a href="#">MadDataClassificationPrevCrop</a>	66
<a href="#">MadDataClassificationSiteData</a>	82
<a href="#">MadDataClassificationSoil</a>	93
<a href="#">MadDataClassificationWeather</a>	110
<a href="#">MadDataset</a>	129
<a href="#">MadGuid</a>	
The <a href="#">MadGuid</a> class	An abstract base class that has a Globally Unique Identifier (GUID) to represent a unique instance
	145
<a href="#">MadMainWindow</a>	148
<a href="#">MadModel</a>	
The <a href="#">MadModel</a> class,	to represent a ModelTheme
	150
<a href="#">MadSerialisable</a>	161
<a href="#">MadStateVars</a>	166
<a href="#">MadSubCategory</a>	178
<a href="#">MadSVCrop</a>	190
<a href="#">MadSVObservations</a>	206
<a href="#">MadSVSoil</a>	217
<a href="#">MadSVSurfaceFluxes</a>	
The <a href="#">MadSVSurfaceFluxes</a> class	231
<a href="#">MadTextDisplayForm</a>	247
<a href="#">MadUtils</a>	249
<a href="#">QDialog</a>	259
<a href="#">QMainWindow</a>	259





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.cpp	298
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.h	299
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/main.cpp	299
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.cpp	261
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.h	261
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.cpp	262
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.h	263
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/mad.h	286
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.cpp	289
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.h	290
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.cpp	290
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.h	291
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.cpp	292
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.h	293
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.cpp	293
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.h	294
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.cpp	295
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.h	295
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.cpp	296
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.h	297
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madversion.h	298
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.- cpp	264
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.- h	265
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.- cpp	266
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.- h	266
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.- cpp	267
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.- h	268
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.- cpp	269
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.- h	269

/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationssitedata.- cpp	270
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationssitedata.- h	271
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.- cpp	272
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.- h	272
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.- cpp	273
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.- h	274
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.- cpp	275
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.h	275
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.- cpp	276
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.- h	277
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.- cpp	278
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.- h	
The <b>MadStateVars</b> class. This contains 4 sub categories	279
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.- cpp	280
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.- h	280
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.- cpp	281
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.- h	282
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.- cpp	283
/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.- h	284

## **Chapter 5**

# **Namespace Documentation**

### **5.1 Ui Namespace Reference**



## Chapter 6

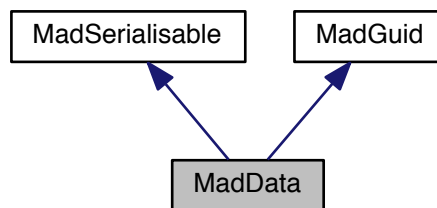
# Class Documentation

### 6.1 MadData Class Reference

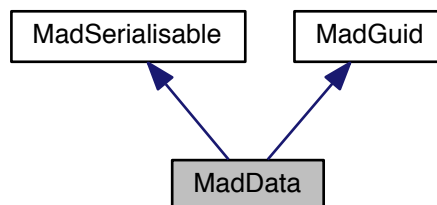
The [MadData](#) class.

```
#include <maddata.h>
```

Inheritance diagram for MadData:



Collaboration diagram for MadData:



## Public Member Functions

- [MadData](#) ()
- [MadData](#) (const [MadData](#) &theData)
- [MadData](#) & operator= (const [MadData](#) &theData)
- [QString](#) [name](#) () const  
*name (accessor) this is the dataset's name*
- [QString](#) [description](#) () const  
*description (accessor) this is the dataset's description*
- [QString](#) [imageFile](#) () const
- void [setName](#) ([QString](#) theName)
- void [setDescription](#) ([QString](#) theDescription)
- void [setImageFile](#) ([QString](#) theImageFileName)
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- bool [fromXml](#) (const [QString](#) theXml)
- virtual bool [toXmlFile](#) (const [QString](#) theFileName)  
*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*
- virtual bool [fromXmlFile](#) (const [QString](#) theFileName)  
*fromXmlFile Read this object from xml in a file*
- [QString](#) [guid](#) () const  
*[MadGuid::guid](#).*
- void [setGuid](#) ([QString](#) theGuid="")  
*[MadGuid::setGuid](#).*

### 6.1.1 Detailed Description

The [MadData](#) class.

Definition at line 38 of file maddata.h.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 [MadData::MadData](#) ( )

Definition at line 32 of file maddata.cpp.

```

32             : MadSerialisable() , MadGuid()
33 {
34     setGuid();
35     mName="No Name Set";
36     mDescription="Not Set";
37 }
```

Here is the call graph for this function:



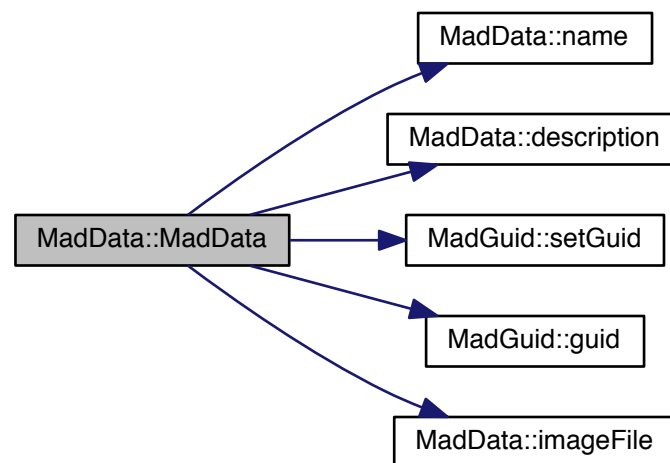
### 6.1.2.2 MadData::MadData ( const MadData & *theData* )

copy constructor

Definition at line 39 of file maddata.cpp.

```
40 {  
41     mName=theData.name();  
42     mDescription=theData.description();  
43     setGuid(theData.guid());  
44     mImageFile=theData.imageFile();  
45 }
```

Here is the call graph for this function:



## 6.1.3 Member Function Documentation

### 6.1.3.1 QString MadData::description ( ) const

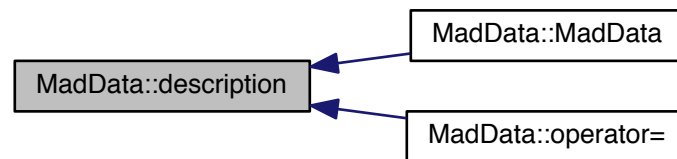
description (accessor) this is the dataset's description

## Returns

Definition at line 65 of file maddata.cpp.

```
66 {
67     return mDescription;
68 }
```

Here is the caller graph for this function:



### 6.1.3.2 bool MadData::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

## See Also

[MadSerialisable](#)

## Note

this class inherits the serialisable interface so it MUST implement this

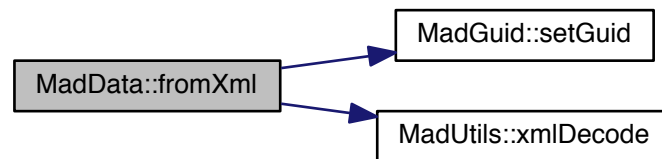
Implements [MadSerialisable](#).

Definition at line 94 of file maddata.cpp.

```
95 {
96     QDomDocument myDocument("mydocument");
97     myDocument.setContent(theXml);
98     QDomElement myTopElement = myDocument.firstChildElement("model");
99     if (myTopElement.isNull())
100     {
101         //TODO - just make this a warning
102         qDebug("the top element couldn't be found!");
103         setGuid(myTopElement.attribute("guid"));
104         mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
105         mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").
106     text());
107         mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
108         return true;
109     }
110     else
111         return false;
112 }
```



Here is the call graph for this function:



**6.1.3.3** `bool MadSerialisable::fromXmlFile ( const QString theFileName )` `[virtual], [inherited]`

`fromXmlFile` Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

result as true for success, false for failure.

Definition at line 76 of file `madserialisable.cpp`.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
  
```

Here is the call graph for this function:



#### 6.1.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

##### Returns

Definition at line 40 of file madguid.cpp.

```
41 {  
42     return mGuid;  
43 }
```

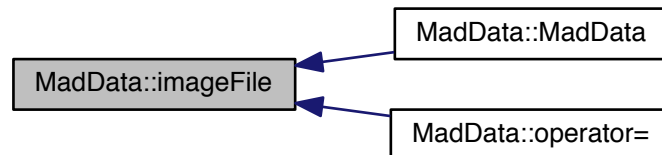
#### 6.1.3.5 QString MadData::imageFile ( ) const

The image file associated with the dataset

Definition at line 70 of file maddata.cpp.

```
71 {  
72     return mImageFile;  
73 }
```

Here is the caller graph for this function:



#### 6.1.3.6 QString MadData::name ( ) const

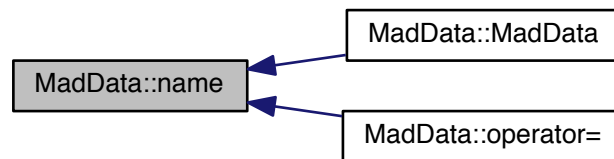
name (accessor) this is the dataset's name

##### Returns

Definition at line 60 of file maddata.cpp.

```
61 {  
62     return mName;  
63 }
```

Here is the caller graph for this function:



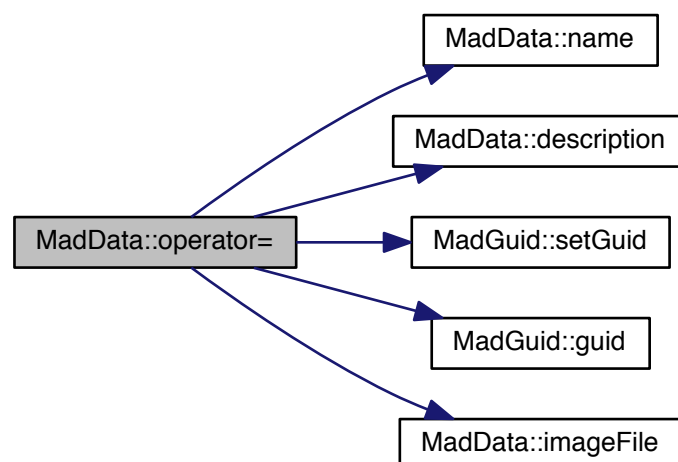
#### 6.1.3.7 MadData & MadData::operator= ( const MadData & theData )

Assignment operator

Definition at line 47 of file maddata.cpp.

```
48 {  
49     if (this == &theData) return *this; // gracefully handles self assignment  
50  
51     mName=theData.name();  
52     mDescription=theData.description();  
53     setGuid(theData.guid());  
54     mImageFile=theData.imageFile();  
55     return *this;  
56 }
```

Here is the call graph for this function:



### 6.1.3.8 void MadData::setDescription ( QString *theDescription* )

Set the model description

See Also

[description\(\)](#)

Definition at line 82 of file maddata.cpp.

```
83 {
84     mDescription=theDescription;
85 }
```

### 6.1.3.9 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid](#).

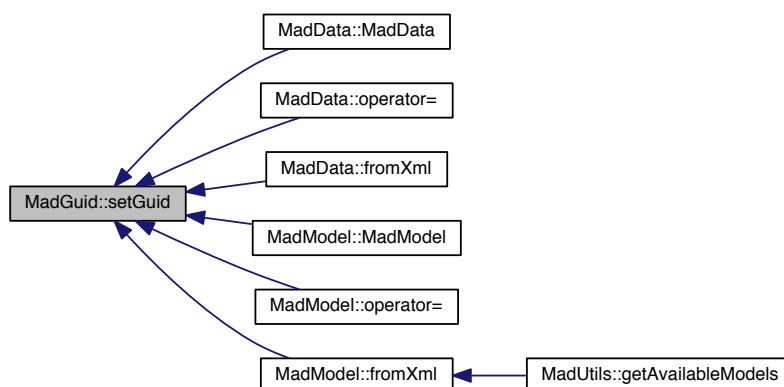
Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```
50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



### 6.1.3.10 void MadData::setImageFile ( QString *theImageFileName* )

Set the image file

## See Also

[imageFile\(\)](#)

Definition at line 87 of file maddata.cpp.

```
88 {  
89     mImageFile=theImageFileName;  
90 }
```

### 6.1.3.11 void MadData::setName ( QString *theName* )

Set the modelName

## See Also

[name\(\)](#)

Definition at line 77 of file maddata.cpp.

```
78 {  
79     mName=theName;  
80 }
```

### 6.1.3.12 QString MadData::toHtml ( )

Return a html text representation of this layer

Definition at line 133 of file maddata.cpp.

```
134 {  
135     QString myString;  
136     myString+="

### 

137     //myString+="138     myString+="139     myString+="140     myString+="141     return myString;  
142 }
```

Here is the call graph for this function:



### 6.1.3.13 QString MadData::toText ( )

Return a plain text representation of this layer

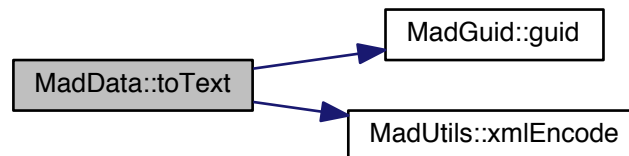
Definition at line 124 of file maddata.cpp.

```

125 {
126   QString myString;
127   myString+=QString("guid=> " + guid() + "\n");
128   myString+=QString("name=> " + MadUtils::xmlEncode(mName) + "\n");
129   myString+=QString("description=> " + MadUtils::xmlEncode(mDescription) + "\n");
130   return myString;
131 }

```

Here is the call graph for this function:



#### 6.1.3.14 QString MadData::toXml( ) [virtual]

Return an xml representation of this layer

##### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

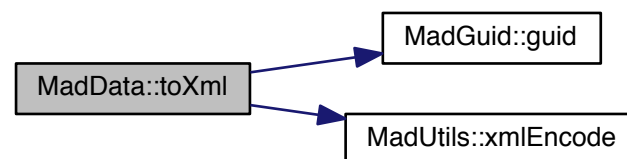
Definition at line 113 of file maddata.cpp.

```

114 {
115   QString myString;
116   myString+=QString("<dataset guid=\"\" + guid() + "\">\n");
117   myString+=QString("  <name>\" + MadUtils::xmlEncode(mName) + "</name>\n");
118   myString+=QString("  <description>\" + MadUtils::xmlEncode(mDescription) + \"\n");
119   myString+=QString("  <imageFile>\" + MadUtils::xmlEncode(mImageFile) + "</imageFile>\n");
120   myString+=QString("</dataset>\n");
121   return myString;
122 }

```

Here is the call graph for this function:



### 6.1.3.15 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

toXmlFile writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:



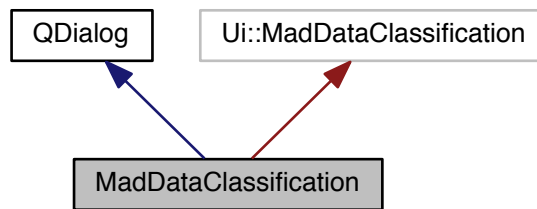
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[maddata.h](#)
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[maddata.cpp](#)

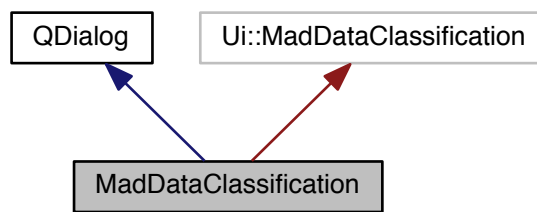
## 6.2 MadDataClassification Class Reference

```
#include <maddataclassification.h>
```

Inheritance diagram for MadDataClassification:



Collaboration diagram for MadDataClassification:



## Public Member Functions

- [MadDataClassification](#) (QWidget \*parent=0)

## Protected Member Functions

- void [changeEvent](#) (QEvent \*e)

### 6.2.1 Detailed Description

Definition at line 38 of file maddataclassification.h.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 MadDataClassification::MadDataClassification ( QWidget \* parent = 0 ) [explicit]

Definition at line 33 of file maddataclassification.cpp.

```

33                                     : QDialog(parent)
34 {
35     setupUi(this);

```



```

36  gbxCultivation->setChecked(false);
37  cbExamples->setEnabled(true);
38  lblExample->setVisible(true);
39  lblExample->setText("Select Example");
40  lblMedalCultivation->setVisible(false);
41  lblRankingCultivation->setVisible(false);
42  lblExample->setVisible(true);
43  cbExamples->setDisabled(false);
44
45  gbxPhenology->setChecked(false);
46  cbExamplesPhenology->setEnabled(true);
47  lblExamplePhenology->setVisible(true);
48  lblExamplePhenology->setText("Select Example");
49  lblMedalPhenology->setVisible(false);
50  lblRankingPhenology->setVisible(false);
51  lblExamplePhenology->setVisible(true);
52  cbExamplesPhenology->setDisabled(false);
53
54  // These must stay here at the end
55
56  // cultivation connections
57  connect ( sbVariety, SIGNAL ( valueChanged(int) ),
58           this, SLOT ( updateVarietyRatingLbl() ));
59  connect ( dsbVariety, SIGNAL ( valueChanged(double) ),
60           this, SLOT ( updateVarietyRatingLbl() ));
61  connect ( sbSowing, SIGNAL ( valueChanged(int) ),
62           this, SLOT ( updateSowingRatingLbl() ));
63  connect ( dsbSowing, SIGNAL ( valueChanged(double) ),
64           this, SLOT ( updateSowingRatingLbl() ));
65  connect ( sbHarvest, SIGNAL ( valueChanged(int) ),
66           this, SLOT ( updateHarvestRatingLbl() ));
67  connect ( dsbHarvest, SIGNAL ( valueChanged(double) ),
68           this, SLOT ( updateHarvestRatingLbl() ));
69  connect ( sbFertilisation, SIGNAL ( valueChanged(int) ),
70           this, SLOT ( updateFertilisationRatingLbl() ));
71  connect ( dsbFertilisation, SIGNAL ( valueChanged(double) ),
72           this, SLOT ( updateFertilisationRatingLbl() ));
73  connect ( sbIrrigation, SIGNAL ( valueChanged(int) ),
74           this, SLOT ( updateIrrigationRatingLbl() ));
75  connect ( dsbIrrigation, SIGNAL ( valueChanged(double) ),
76           this, SLOT ( updateIrrigationRatingLbl() ));
77  connect ( sbSeedDensity, SIGNAL ( valueChanged(int) ),
78           this, SLOT ( updateSeedDensityRatingLbl() ));
79  connect ( dsbSeedDensity, SIGNAL ( valueChanged(double) ),
80           this, SLOT ( updateSeedDensityRatingLbl() ));
81  connect ( sbYield, SIGNAL ( valueChanged(int) ),
82           this, SLOT ( updateYieldRatingLbl() ));
83  connect ( dsbYield, SIGNAL ( valueChanged(double) ),
84           this, SLOT ( updateYieldRatingLbl() ));
85  connect ( sbTillage, SIGNAL ( valueChanged(int) ),
86           this, SLOT ( updateTillageRatingLbl() ));
87  connect ( dsbTillage, SIGNAL ( valueChanged(double) ),
88           this, SLOT ( updateTillageRatingLbl() ));
89
90  // phenology connections
91  connect ( sbEmergencePhenology, SIGNAL ( valueChanged(int) ),
92           this, SLOT ( updatePhenologyEmergenceRatingLbl() ));
93  connect ( dsbEmergencePhenology, SIGNAL ( valueChanged(double) ),
94           this, SLOT ( updatePhenologyEmergenceRatingLbl() ));
95  connect ( sbStemElongationPhenology, SIGNAL ( valueChanged(int) ),
96           this, SLOT ( updatePhenologyStemElongationRatingLbl() ));
97  connect ( dsbStemElongationPhenology, SIGNAL ( valueChanged(double) ),
98           this, SLOT ( updatePhenologyStemElongationRatingLbl() ));
99  connect ( sbEarEmergencePhenology, SIGNAL ( valueChanged(int) ),
100          this, SLOT ( updatePhenologyEarEmergenceRatingLbl() ));
101  connect ( dsbEarEmergencePhenology, SIGNAL ( valueChanged(double) ),
102          this, SLOT ( updatePhenologyEarEmergenceRatingLbl() ));
103  connect ( sbFloweringPhenology, SIGNAL ( valueChanged(int) ),
104          this, SLOT ( updatePhenologyFloweringRatingLbl() ));
105  connect ( dsbFloweringPhenology, SIGNAL ( valueChanged(double) ),
106          this, SLOT ( updatePhenologyFloweringRatingLbl() ));
107  connect ( sbYellowRipenessPhenology, SIGNAL ( valueChanged(int) ),
108          this, SLOT ( updatePhenologyYellowRipenessRatingLbl() ));
109  connect ( dsbYellowRipenessPhenology, SIGNAL ( valueChanged(double) ),
110          this, SLOT ( updatePhenologyYellowRipenessRatingLbl() ));
111
112  // prev crop connections
113  connect ( sbCropPrevCrop, SIGNAL ( valueChanged(int) ),
114          this, SLOT ( updatePrevCropCropRatingLbl() ));
115  connect ( dsbCropPrevCrop, SIGNAL ( valueChanged(double) ),
116          this, SLOT ( updatePrevCropCropRatingLbl() ));
117  connect ( sbSowingDatePrevCrop, SIGNAL ( valueChanged(int) ),
118          this, SLOT ( updatePrevCropSowingDateRatingLbl() ));
119  connect ( dsbSowingDatePrevCrop, SIGNAL ( valueChanged(double) ),
120          this, SLOT ( updatePrevCropSowingDateRatingLbl() ));
121  connect ( sbHarvestDatePrevCrop, SIGNAL ( valueChanged(int) ),
122          this, SLOT ( updatePrevCropHarvestDateRatingLbl() ));

```

```

123 connect ( dsbHarvestDatePrevCrop, SIGNAL ( valueChanged(double) ),
124           this, SLOT ( updatePrevCropHarvestDateRatingLbl() ));
125 connect ( sbYieldPrevCrop, SIGNAL ( valueChanged(int) ),
126           this, SLOT ( updatePrevCropYieldRatingLbl() ));
127 connect ( dsbYieldPrevCrop, SIGNAL ( valueChanged(double) ),
128           this, SLOT ( updatePrevCropYieldRatingLbl() ));
129 connect ( sbResidueMgmtPrevCrop, SIGNAL ( valueChanged(int) ),
130           this, SLOT ( updatePrevCropResidueMgmtRatingLbl() ));
131 connect ( dsbResidueMgmtPrevCrop, SIGNAL ( valueChanged(double) ),
132           this, SLOT ( updatePrevCropResidueMgmtRatingLbl() ));
133 connect ( sbFertilisationPrevCrop, SIGNAL ( valueChanged(int) ),
134           this, SLOT ( updatePrevCropFertilisationRatingLbl() ));
135 connect ( dsbFertilisationPrevCrop, SIGNAL ( valueChanged(double) ),
136           this, SLOT ( updatePrevCropFertilisationRatingLbl() ));
137 connect ( sbIrrigationPrevCrop, SIGNAL ( valueChanged(int) ),
138           this, SLOT ( updatePrevCropIrrigationRatingLbl() ));
139 connect ( dsbIrrigationPrevCrop, SIGNAL ( valueChanged(double) ),
140           this, SLOT ( updatePrevCropIrrigationRatingLbl() ));
141
142 // initial values connections
143 connect ( sbSoilMoistureInitialValues, SIGNAL ( valueChanged(int) ),
144           this, SLOT ( updateInitialValuesSoilMoistureRatingLbl() ));
145 connect ( dsbSoilMoistureInitialValues, SIGNAL ( valueChanged(double) ),
146           this, SLOT ( updateInitialValuesSoilMoistureRatingLbl() ));
147 connect ( sbNMinInitialValues, SIGNAL ( valueChanged(int) ),
148           this, SLOT ( updateInitialValuesNMinRatingLbl() ));
149 connect ( dsbNMinInitialValues, SIGNAL ( valueChanged(double) ),
150           this, SLOT ( updateInitialValuesNMinRatingLbl() ));
151
152 // soil connections
153 connect ( sbCOrgSoil, SIGNAL ( valueChanged(int) ),
154           this, SLOT ( updateSoilCOrgRatingLbl() ));
155 connect ( dsbCOrgSoil, SIGNAL ( valueChanged(double) ),
156           this, SLOT ( updateSoilCOrgRatingLbl() ));
157 connect ( sbNOrgSoil, SIGNAL ( valueChanged(int) ),
158           this, SLOT ( updateSoilNOrgRatingLbl() ));
159 connect ( dsbNOrgSoil, SIGNAL ( valueChanged(double) ),
160           this, SLOT ( updateSoilNOrgRatingLbl() ));
161 connect ( sbTextureSoil, SIGNAL ( valueChanged(int) ),
162           this, SLOT ( updateSoilTextureRatingLbl() ));
163 connect ( dsbTextureSoil, SIGNAL ( valueChanged(double) ),
164           this, SLOT ( updateSoilTextureRatingLbl() ));
165 connect ( sbBulkDensitySoil, SIGNAL ( valueChanged(int) ),
166           this, SLOT ( updateSoilBulkDensityRatingLbl() ));
167 connect ( dsbBulkDensitySoil, SIGNAL ( valueChanged(double) ),
168           this, SLOT ( updateSoilBulkDensityRatingLbl() ));
169 connect ( sbFieldCapacitySoil, SIGNAL ( valueChanged(int) ),
170           this, SLOT ( updateSoilFieldCapacityRatingLbl() ));
171 connect ( dsbFieldCapacitySoil, SIGNAL ( valueChanged(double) ),
172           this, SLOT ( updateSoilFieldCapacityRatingLbl() ));
173 connect ( sbWiltingPointSoil, SIGNAL ( valueChanged(int) ),
174           this, SLOT ( updateSoilWiltingPointRatingLbl() ));
175 connect ( dsbWiltingPointSoil, SIGNAL ( valueChanged(double) ),
176           this, SLOT ( updateSoilWiltingPointRatingLbl() ));
177 connect ( sbPfCurveSoil, SIGNAL ( valueChanged(int) ),
178           this, SLOT ( updateSoilPfCurveRatingLbl() ));
179 connect ( dsbPfCurveSoil, SIGNAL ( valueChanged(double) ),
180           this, SLOT ( updateSoilPfCurveRatingLbl() ));
181 connect ( sbHydrCondCurveSoil, SIGNAL ( valueChanged(int) ),
182           this, SLOT ( updateSoilHydrCondCurveRatingLbl() ));
183 connect ( dsbHydrCondCurveSoil, SIGNAL ( valueChanged(double) ),
184           this, SLOT ( updateSoilHydrCondCurveRatingLbl() ));
185 connect ( sbPhSoil, SIGNAL ( valueChanged(int) ),
186           this, SLOT ( updateSoilPhRatingLbl() ));
187 connect ( dsbPhSoil, SIGNAL ( valueChanged(double) ),
188           this, SLOT ( updateSoilPhRatingLbl() ));
189
190 // site data connections
191 connect ( sbLatitudeSite, SIGNAL ( valueChanged(int) ),
192           this, SLOT ( updateSiteLatitudeRatingLbl() ));
193 connect ( dsbLatitudeSite, SIGNAL ( valueChanged(double) ),
194           this, SLOT ( updateSiteLatitudeRatingLbl() ));
195 connect ( sbLongitudeSite, SIGNAL ( valueChanged(int) ),
196           this, SLOT ( updateSiteLongitudeRatingLbl() ));
197 connect ( dsbLongitudeSite, SIGNAL ( valueChanged(double) ),
198           this, SLOT ( updateSiteLongitudeRatingLbl() ));
199 connect ( sbAltitudeSite, SIGNAL ( valueChanged(int) ),
200           this, SLOT ( updateSiteAltitudeRatingLbl() ));
201 connect ( dsbAltitudeSite, SIGNAL ( valueChanged(double) ),
202           this, SLOT ( updateSiteAltitudeRatingLbl() ));
203
204 // weather connections
205 connect ( sbPrecipitationWeather, SIGNAL ( valueChanged(int) ),
206           this, SLOT ( updateWeatherPrecipitationRatingLbl() ));
207 connect ( dsbPrecipitationWeather, SIGNAL ( valueChanged(double) ),
208           this, SLOT ( updateWeatherPrecipitationRatingLbl() ));
209 connect ( sbTAveWeather, SIGNAL ( valueChanged(int) ),

```

```

210         this, SLOT ( updateWeatherTaveRatingLbl() ) );
211 connect ( dsbTaveWeather, SIGNAL ( valueChanged(double) ),
212         this, SLOT ( updateWeatherTaveRatingLbl() ) );
213 connect ( sbTminWeather, SIGNAL ( valueChanged(int) ),
214         this, SLOT ( updateWeatherTminRatingLbl() ) );
215 connect ( dsbTminWeather, SIGNAL ( valueChanged(double) ),
216         this, SLOT ( updateWeatherTminRatingLbl() ) );
217 connect ( sbTmaxWeather, SIGNAL ( valueChanged(int) ),
218         this, SLOT ( updateWeatherTmaxRatingLbl() ) );
219 connect ( dsbTmaxWeather, SIGNAL ( valueChanged(double) ),
220         this, SLOT ( updateWeatherTmaxRatingLbl() ) );
221 connect ( sbRelHumidityWeather, SIGNAL ( valueChanged(int) ),
222         this, SLOT ( updateWeatherRelHumidityRatingLbl() ) );
223 connect ( dsbRelHumidityWeather, SIGNAL ( valueChanged(double) ),
224         this, SLOT ( updateWeatherRelHumidityRatingLbl() ) );
225 connect ( sbWindSpeedWeather, SIGNAL ( valueChanged(int) ),
226         this, SLOT ( updateWeatherWindSpeedRatingLbl() ) );
227 connect ( dsbWindSpeedWeather, SIGNAL ( valueChanged(double) ),
228         this, SLOT ( updateWeatherWindSpeedRatingLbl() ) );
229 connect ( sbGlobalRadiationWeather, SIGNAL ( valueChanged(int) ),
230         this, SLOT ( updateWeatherGlobalRadiationRatingLbl() ) );
231 connect ( dsbGlobalRadiationWeather, SIGNAL ( valueChanged(double) ),
232         this, SLOT ( updateWeatherGlobalRadiationRatingLbl() ) );
233 connect ( sbSunshineHoursWeather, SIGNAL ( valueChanged(int) ),
234         this, SLOT ( updateWeatherSunshineHoursRatingLbl() ) );
235 connect ( dsbSunshineHoursWeather, SIGNAL ( valueChanged(double) ),
236         this, SLOT ( updateWeatherSunshineHoursRatingLbl() ) );
237 connect ( sbLeafWetnessWeather, SIGNAL ( valueChanged(int) ),
238         this, SLOT ( updateWeatherLeafWetnessRatingLbl() ) );
239 connect ( dsbLeafWetnessWeather, SIGNAL ( valueChanged(double) ),
240         this, SLOT ( updateWeatherLeafWetnessRatingLbl() ) );
241 connect ( sbSoilTempWeather, SIGNAL ( valueChanged(int) ),
242         this, SLOT ( updateWeatherSoilTempRatingLbl() ) );
243 connect ( dsbSoilTempWeather, SIGNAL ( valueChanged(double) ),
244         this, SLOT ( updateWeatherSoilTempRatingLbl() ) );
245
246 // state vars connections
247 //crop
248 connect ( dsbSVCropAGrBiomassLayers, SIGNAL ( valueChanged(double) ),
249         this, SLOT ( updateSVCropAGrBiomassRatingLbl() ) );
250 connect ( sbSVCropAGrBiomassObservations, SIGNAL ( valueChanged(int) ),
251         this, SLOT ( updateSVCropAGrBiomassRatingLbl() ) );
252 connect ( dsbSVCropAGrBiomassWeightPts, SIGNAL ( valueChanged(double) ),
253         this, SLOT ( updateSVCropAGrBiomassRatingLbl() ) );
254 connect ( dsbSVCropAGrBiomassReplicates, SIGNAL ( valueChanged(double) ),
255         this, SLOT ( updateSVCropAGrBiomassRatingLbl() ) );
256
257 connect ( dsbSVCropWeightOrgansLayers, SIGNAL ( valueChanged(double) ),
258         this, SLOT ( updateSVCropWeightOrgansRatingLbl() ) );
259 connect ( sbSVCropWeightOrgansObservations, SIGNAL ( valueChanged(int) ),
260         this, SLOT ( updateSVCropWeightOrgansRatingLbl() ) );
261 connect ( dsbSVCropWeightOrgansWeightPts, SIGNAL ( valueChanged(double) ),
262         this, SLOT ( updateSVCropWeightOrgansRatingLbl() ) );
263 connect ( dsbSVCropWeightOrgansReplicates, SIGNAL ( valueChanged(double) ),
264         this, SLOT ( updateSVCropWeightOrgansRatingLbl() ) );
265
266 connect ( dsbSVCropRootBiomassLayers, SIGNAL ( valueChanged(double) ),
267         this, SLOT ( updateSVCropRootBiomassRatingLbl() ) );
268 connect ( sbSVCropRootBiomassObservations, SIGNAL ( valueChanged(int) ),
269         this, SLOT ( updateSVCropRootBiomassRatingLbl() ) );
270 connect ( dsbSVCropRootBiomassWeightPts, SIGNAL ( valueChanged(double) ),
271         this, SLOT ( updateSVCropRootBiomassRatingLbl() ) );
272 connect ( dsbSVCropRootBiomassReplicates, SIGNAL ( valueChanged(double) ),
273         this, SLOT ( updateSVCropRootBiomassRatingLbl() ) );
274
275 connect ( dsbSVCropNInAGrBiomassLayers, SIGNAL ( valueChanged(double) ),
276         this, SLOT ( updateSVCropNInAGrBiomassRatingLbl() ) );
277 connect ( sbSVCropNInAGrBiomassObservations, SIGNAL ( valueChanged(int) ),
278         this, SLOT ( updateSVCropNInAGrBiomassRatingLbl() ) );
279 connect ( dsbSVCropNInAGrBiomassWeightPts, SIGNAL ( valueChanged(double) ),
280         this, SLOT ( updateSVCropNInAGrBiomassRatingLbl() ) );
281 connect ( dsbSVCropNInAGrBiomassReplicates, SIGNAL ( valueChanged(double) ),
282         this, SLOT ( updateSVCropNInAGrBiomassRatingLbl() ) );
283
284 connect ( dsbSVCropNInOrgansLayers, SIGNAL ( valueChanged(double) ),
285         this, SLOT ( updateSVCropNInOrgansRatingLbl() ) );
286 connect ( sbSVCropNInOrgansObservations, SIGNAL ( valueChanged(int) ),
287         this, SLOT ( updateSVCropNInOrgansRatingLbl() ) );
288 connect ( dsbSVCropNInOrgansWeightPts, SIGNAL ( valueChanged(double) ),
289         this, SLOT ( updateSVCropNInOrgansRatingLbl() ) );
290 connect ( dsbSVCropNInOrgansReplicates, SIGNAL ( valueChanged(double) ),
291         this, SLOT ( updateSVCropNInOrgansRatingLbl() ) );
292
293 connect ( dsbSVCropLAIRLayers, SIGNAL ( valueChanged(double) ),
294         this, SLOT ( updateSVCropLAIRRatingLbl() ) );
295 connect ( sbSVCropLAIObservations, SIGNAL ( valueChanged(int) ),
296         this, SLOT ( updateSVCropLAIRRatingLbl() ) );

```

```

297 connect ( dsbSVCropLAIWeightPts, SIGNAL ( valueChanged(double) ),
298           this, SLOT ( updateSVCropLAIRatingLbl() ) );
299 connect ( dsbSVCropLAIReplicates, SIGNAL ( valueChanged(double) ),
300           this, SLOT ( updateSVCropLAIRatingLbl() ) );
301
302 // soil
303 connect ( dsbSVSoilSoilWaterSensorCalLayers, SIGNAL ( valueChanged(double) ),
304           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
305 connect ( sbSVSoilSoilWaterSensorCalObservations, SIGNAL ( valueChanged(int) ),
306           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
307 connect ( dsbSVSoilSoilWaterSensorCalWeightPts, SIGNAL ( valueChanged(double) ),
308           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
309 connect ( dsbSVSoilSoilWaterSensorCalReplicates, SIGNAL ( valueChanged(double) ),
310           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
311
312 connect ( dsbSVSoilPressureHeadsLayers, SIGNAL ( valueChanged(double) ),
313           this, SLOT ( updateSVSoilPressureHeadsRatingLbl() ) );
314 connect ( sbSVSoilPressureHeadsObservations, SIGNAL ( valueChanged(int) ),
315           this, SLOT ( updateSVSoilPressureHeadsRatingLbl() ) );
316 connect ( dsbSVSoilPressureHeadsWeightPts, SIGNAL ( valueChanged(double) ),
317           this, SLOT ( updateSVSoilPressureHeadsRatingLbl() ) );
318 connect ( dsbSVSoilPressureHeadsReplicates, SIGNAL ( valueChanged(double) ),
319           this, SLOT ( updateSVSoilPressureHeadsRatingLbl() ) );
320
321 connect ( dsbSVSoilNMinLayers, SIGNAL ( valueChanged(double) ),
322           this, SLOT ( updateSVSoilNMinRatingLbl() ) );
323 connect ( sbSVSoilNMinObservations, SIGNAL ( valueChanged(int) ),
324           this, SLOT ( updateSVSoilNMinRatingLbl() ) );
325 connect ( dsbSVSoilNMinWeightPts, SIGNAL ( valueChanged(double) ),
326           this, SLOT ( updateSVSoilNMinRatingLbl() ) );
327 connect ( dsbSVSoilNMinReplicates, SIGNAL ( valueChanged(double) ),
328           this, SLOT ( updateSVSoilNMinRatingLbl() ) );
329
330 connect ( dsbSVSoilSoilWaterSensorCalLayers, SIGNAL ( valueChanged(double) ),
331           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
332 connect ( sbSVSoilSoilWaterSensorCalObservations, SIGNAL ( valueChanged(int) ),
333           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
334 connect ( dsbSVSoilSoilWaterSensorCalWeightPts, SIGNAL ( valueChanged(double) ),
335           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
336 connect ( dsbSVSoilSoilWaterSensorCalReplicates, SIGNAL ( valueChanged(double) ),
337           this, SLOT ( updateSVSoilSoilWaterSensorCalRatingLbl() ) );
338
339 connect ( dsbSVSoilWaterFluxBottomRootLayers, SIGNAL ( valueChanged(double) ),
340           this, SLOT ( updateSVSoilWaterFluxBottomRootRatingLbl() ) );
341 connect ( sbSVSoilWaterFluxBottomRootObservations, SIGNAL ( valueChanged(int) ),
342           this, SLOT ( updateSVSoilWaterFluxBottomRootRatingLbl() ) );
343 connect ( dsbSVSoilWaterFluxBottomRootWeightPts, SIGNAL ( valueChanged(double) ),
344           this, SLOT ( updateSVSoilWaterFluxBottomRootRatingLbl() ) );
345 connect ( dsbSVSoilWaterFluxBottomRootReplicates, SIGNAL ( valueChanged(double) ),
346           this, SLOT ( updateSVSoilWaterFluxBottomRootRatingLbl() ) );
347
348 connect ( dsbSVSoilNFluxBottomRootLayers, SIGNAL ( valueChanged(double) ),
349           this, SLOT ( updateSVSoilNFluxBottomRootRatingLbl() ) );
350 connect ( sbSVSoilNFluxBottomRootObservations, SIGNAL ( valueChanged(int) ),
351           this, SLOT ( updateSVSoilNFluxBottomRootRatingLbl() ) );
352 connect ( dsbSVSoilNFluxBottomRootWeightPts, SIGNAL ( valueChanged(double) ),
353           this, SLOT ( updateSVSoilNFluxBottomRootRatingLbl() ) );
354 connect ( dsbSVSoilNFluxBottomRootReplicates, SIGNAL ( valueChanged(double) ),
355           this, SLOT ( updateSVSoilNFluxBottomRootRatingLbl() ) );
356
357 // surface fluxes
358 connect ( dsbSVSurfaceFluxesEtLayers, SIGNAL ( valueChanged(double) ),
359           this, SLOT ( updateSVSurfaceFluxesEtRatingLbl() ) );
360 connect ( sbSVSurfaceFluxesEtObservations, SIGNAL ( valueChanged(int) ),
361           this, SLOT ( updateSVSurfaceFluxesEtRatingLbl() ) );
362 connect ( dsbSVSurfaceFluxesEtWeightPts, SIGNAL ( valueChanged(double) ),
363           this, SLOT ( updateSVSurfaceFluxesEtRatingLbl() ) );
364 connect ( dsbSVSurfaceFluxesEtReplicates, SIGNAL ( valueChanged(double) ),
365           this, SLOT ( updateSVSurfaceFluxesEtRatingLbl() ) );
366
367 connect ( dsbSVSurfaceFluxesNh3LossLayers, SIGNAL ( valueChanged(double) ),
368           this, SLOT ( updateSVSurfaceFluxesNh3LossRatingLbl() ) );
369 connect ( sbSVSurfaceFluxesNh3LossObservations, SIGNAL ( valueChanged(int) ),
370           this, SLOT ( updateSVSurfaceFluxesNh3LossRatingLbl() ) );
371 connect ( dsbSVSurfaceFluxesNh3LossWeightPts, SIGNAL ( valueChanged(double) ),
372           this, SLOT ( updateSVSurfaceFluxesNh3LossRatingLbl() ) );
373 connect ( dsbSVSurfaceFluxesNh3LossReplicates, SIGNAL ( valueChanged(double) ),
374           this, SLOT ( updateSVSurfaceFluxesNh3LossRatingLbl() ) );
375
376 connect ( dsbSVSurfaceFluxesN2OLossLayers, SIGNAL ( valueChanged(double) ),
377           this, SLOT ( updateSVSurfaceFluxesN2OLossRatingLbl() ) );
378 connect ( sbSVSurfaceFluxesN2OLossObservations, SIGNAL ( valueChanged(int) ),
379           this, SLOT ( updateSVSurfaceFluxesN2OLossRatingLbl() ) );
380 connect ( dsbSVSurfaceFluxesN2OLossWeightPts, SIGNAL ( valueChanged(double) ),
381           this, SLOT ( updateSVSurfaceFluxesN2OLossRatingLbl() ) );
382 connect ( dsbSVSurfaceFluxesN2OLossReplicates, SIGNAL ( valueChanged(double) ),
383           this, SLOT ( updateSVSurfaceFluxesN2OLossRatingLbl() ) );

```

```

384
385 connect ( dsbSVSurfaceFluxesN2LossLayers, SIGNAL ( valueChanged(double) ),
386           this, SLOT ( updateSVSurfaceFluxesN2LossRatingLbl() ));
387 connect ( sbSVSurfaceFluxesN2LossObservations, SIGNAL ( valueChanged(int) ),
388           this, SLOT ( updateSVSurfaceFluxesN2LossRatingLbl() ));
389 connect ( dsbSVSurfaceFluxesN2LossWeightPts, SIGNAL ( valueChanged(double) ),
390           this, SLOT ( updateSVSurfaceFluxesN2LossRatingLbl() ));
391 connect ( dsbSVSurfaceFluxesN2LossReplicates, SIGNAL ( valueChanged(double) ),
392           this, SLOT ( updateSVSurfaceFluxesN2LossRatingLbl() ));
393
394 connect ( dsbSVSurfaceFluxesCh4LossLayers, SIGNAL ( valueChanged(double) ),
395           this, SLOT ( updateSVSurfaceFluxesCh4LossRatingLbl() ));
396 connect ( sbSVSurfaceFluxesCh4LossObservations, SIGNAL ( valueChanged(int) ),
397           this, SLOT ( updateSVSurfaceFluxesCh4LossRatingLbl() ));
398 connect ( dsbSVSurfaceFluxesCh4LossWeightPts, SIGNAL ( valueChanged(double) ),
399           this, SLOT ( updateSVSurfaceFluxesCh4LossRatingLbl() ));
400 connect ( dsbSVSurfaceFluxesCh4LossReplicates, SIGNAL ( valueChanged(double) ),
401           this, SLOT ( updateSVSurfaceFluxesCh4LossRatingLbl() ));
402
403 // observations
404 connect ( dsbSVObservationsLodgingLayers, SIGNAL ( valueChanged(double) ),
405           this, SLOT ( updateSVObservationsLodgingRatingLbl() ));
406 connect ( sbSVObservationsLodgingObservations, SIGNAL ( valueChanged(int) ),
407           this, SLOT ( updateSVObservationsLodgingRatingLbl() ));
408 connect ( dsbSVObservationsLodgingWeightPts, SIGNAL ( valueChanged(double) ),
409           this, SLOT ( updateSVObservationsLodgingRatingLbl() ));
410 connect ( dsbSVObservationsLodgingReplicates, SIGNAL ( valueChanged(double) ),
411           this, SLOT ( updateSVObservationsLodgingRatingLbl() ));
412
413 connect ( dsbSVObservationsPestsOrDiseasesLayers, SIGNAL ( valueChanged(double) ),
414           this, SLOT ( updateSVObservationsPestsOrDiseasesRatingLbl() ));
415 connect ( sbSVObservationsPestsOrDiseasesObservations, SIGNAL ( valueChanged(int) ),
416           this, SLOT ( updateSVObservationsPestsOrDiseasesRatingLbl() ));
417 connect ( dsbSVObservationsPestsOrDiseasesWeightPts, SIGNAL ( valueChanged(double) ),
418           this, SLOT ( updateSVObservationsPestsOrDiseasesRatingLbl() ));
419 connect ( dsbSVObservationsPestsOrDiseasesReplicates, SIGNAL ( valueChanged(double) ),
420           this, SLOT ( updateSVObservationsPestsOrDiseasesRatingLbl() ));
421
422 connect ( dsbSVObservationsDamagesLayers, SIGNAL ( valueChanged(double) ),
423           this, SLOT ( updateSVObservationsDamagesRatingLbl() ));
424 connect ( sbSVObservationsDamagesObservations, SIGNAL ( valueChanged(int) ),
425           this, SLOT ( updateSVObservationsDamagesRatingLbl() ));
426 connect ( dsbSVObservationsLodgingWeightPts, SIGNAL ( valueChanged(double) ),
427           this, SLOT ( updateSVObservationsDamagesRatingLbl() ));
428 connect ( dsbSVObservationsLodgingReplicates, SIGNAL ( valueChanged(double) ),
429           this, SLOT ( updateSVObservationsDamagesRatingLbl() ));
430 }

```

## 6.2.3 Member Function Documentation

### 6.2.3.1 void MadDataClassification::changeEvent ( QEvent \* e ) [protected]

Definition at line 431 of file maddataclassification.cpp.

```

432 {
433     QDialog::changeEvent(e);
434     switch (e->type()) {
435     case QEvent::LanguageChange:
436         retranslateUi(this);
437         break;
438     default:
439         break;
440     }
441 }

```

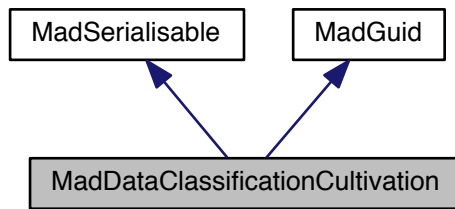
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.h
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/maddataclassification.cpp

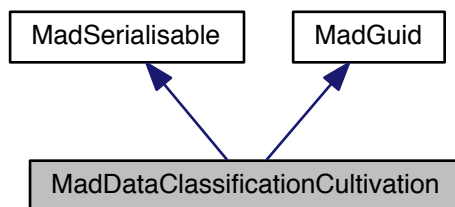
## 6.3 MadDataClassificationCultivation Class Reference

```
#include <maddataclassificationcultivation.h>
```

Inheritance diagram for MadDataClassificationCultivation:



Collaboration diagram for MadDataClassificationCultivation:



## Public Member Functions

- [MadDataClassificationCultivation](#) ()
- [MadDataClassificationCultivation](#) (const [MadDataClassificationCultivation](#) &theData)
- [MadDataClassificationCultivation](#) & operator= (const [MadDataClassificationCultivation](#) &theData)
- [MadSubCategory](#) variety () const
- [MadSubCategory](#) sowing () const
- [MadSubCategory](#) harvest () const
- [MadSubCategory](#) fertilisation () const
- [MadSubCategory](#) irrigation () const
- [MadSubCategory](#) seedDensity () const
- [MadSubCategory](#) yield () const
- [MadSubCategory](#) tillage () const
- [QString](#) toXml ()
- [QString](#) toText ()
- [QString](#) toHtml ()
- bool fromXml (const [QString](#) theXml)
- void setVariety ([MadSubCategory](#) theData)
- void setSowing ([MadSubCategory](#) theData)
- void setHarvest ([MadSubCategory](#) theData)
- void setFertilisation ([MadSubCategory](#) theData)
- void setIrrigation ([MadSubCategory](#) theData)



- void [setSeedDensity](#) (MadSubCategory theData)
- void [setYield](#) (MadSubCategory theData)
- void [setTillage](#) (MadSubCategory theData)
- virtual bool [toXmlFile](#) (const QString theFileName)  
*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*
- virtual bool [fromXmlFile](#) (const QString theFileName)  
*fromXmlFile Read this object from xml in a file*
- QString [guid](#) () const  
[MadGuid::guid.](#)
- void [setGuid](#) (QString theGuid="")  
[MadGuid::setGuid.](#)

### 6.3.1 Detailed Description

Definition at line 35 of file maddataclassificationcultivation.h.

### 6.3.2 Constructor & Destructor Documentation

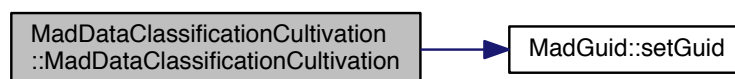
#### 6.3.2.1 MadDataClassificationCultivation::MadDataClassificationCultivation ( )

Definition at line 33 of file maddataclassificationcultivation.cpp.

```

33                                     :
34     MadSerialisable(), MadGuid()
35 {
36     setGuid();
37 }
```

Here is the call graph for this function:



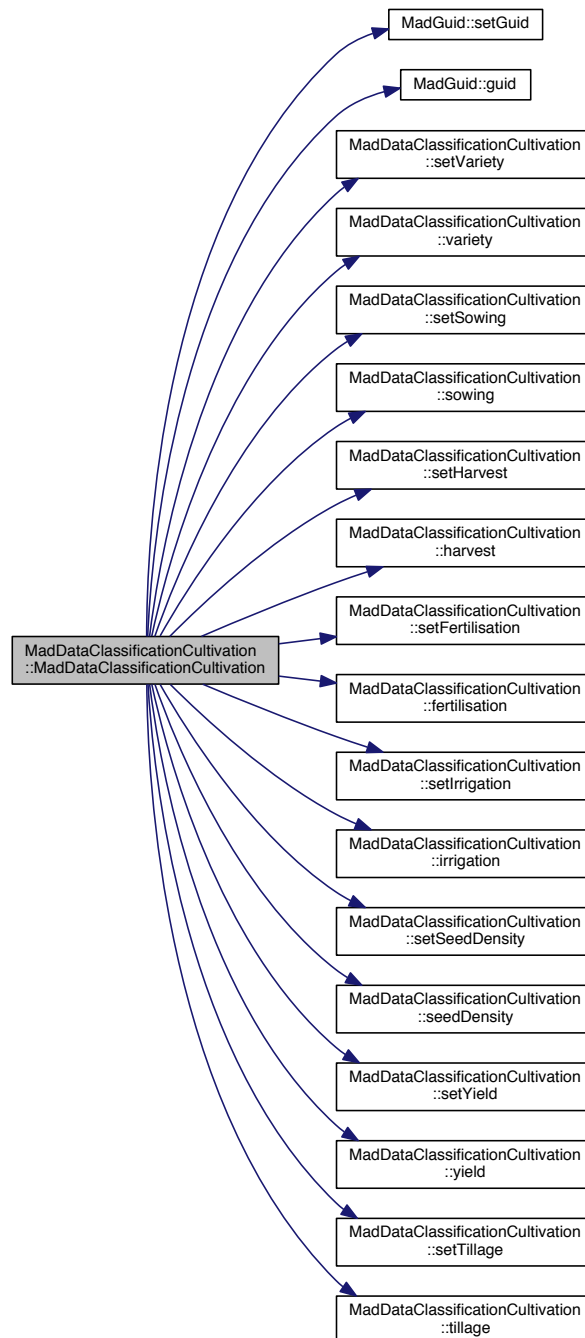
#### 6.3.2.2 MadDataClassificationCultivation::MadDataClassificationCultivation ( const MadDataClassificationCultivation & theData )

Definition at line 38 of file maddataclassificationcultivation.cpp.

```

39 {
40     setGuid(theData.guid());
41     setVariety(theData.variety());
42     setSowing(theData.sowing());
43     setHarvest(theData.harvest());
44     setFertilisation(theData.fertilisation());
45     setIrrigation(theData.irrigation());
46     setSeedDensity(theData.seedDensity());
47     setYield(theData.yield());
48     setTillage(theData.tillage());
49 }
```

Here is the call graph for this function:



### 6.3.3 Member Function Documentation

#### 6.3.3.1 MadSubCategory MadDataClassificationCultivation::fertilisation ( ) const

Definition at line 80 of file maddataclassificationcultivation.cpp.

```

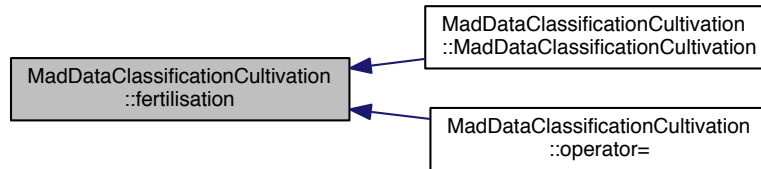
81 {
82     return mFertilisation;

```



83 }

Here is the caller graph for this function:



### 6.3.3.2 bool MadDataClassificationCultivation::fromXml ( const QString *theXml* ) [virtual]

true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

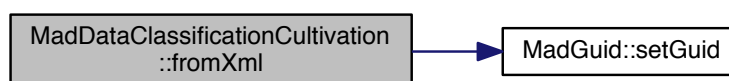
Implements [MadSerialisable](#).

Definition at line 136 of file `maddataclassificationcultivation.cpp`.

```

137 {
138     QDomDocument myDocument("mydocument");
139     myDocument.setContent(theXml);
140     QDomElement myTopElement = myDocument.firstChildElement("cultivation");
141     if (myTopElement.isNull())
142     {
143         //TODO - just make this a warning
144         qDebug("the top element couldn't be found!");
145         setGuid(myTopElement.attribute("guid"));
146         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
147         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
148         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
149         return true;
150     }
151     else
152         return false;
153 }
  
```

Here is the call graph for this function:



### 6.3.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



### 6.3.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid](#).

Destructor Retrieve the GUID

#### Returns

Definition at line 40 of file madguid.cpp.

```

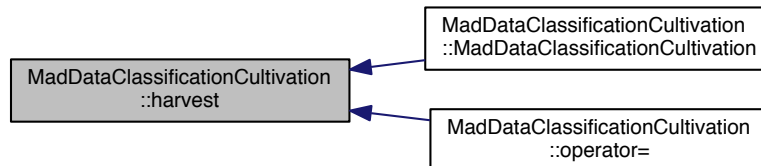
41 {
42     return mGuid;
43 }
```

### 6.3.3.5 MadSubCategory MadDataClassificationCultivation::harvest ( ) const

Definition at line 76 of file maddataclassificationcultivation.cpp.

```
77 {
78     return mHarvest;
79 }
```

Here is the caller graph for this function:

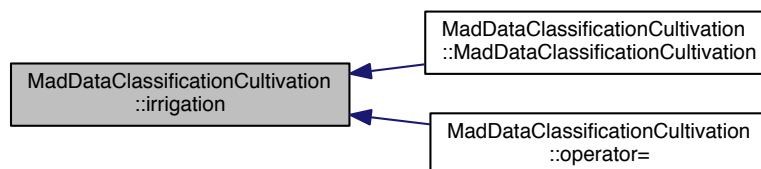


### 6.3.3.6 MadSubCategory MadDataClassificationCultivation::irrigation ( ) const

Definition at line 84 of file maddataclassificationcultivation.cpp.

```
85 {
86     return mIrrigation;
87 }
```

Here is the caller graph for this function:



### 6.3.3.7 MadDataClassificationCultivation & MadDataClassificationCultivation::operator= ( const MadDataClassificationCultivation & theData )

Definition at line 51 of file maddataclassificationcultivation.cpp.

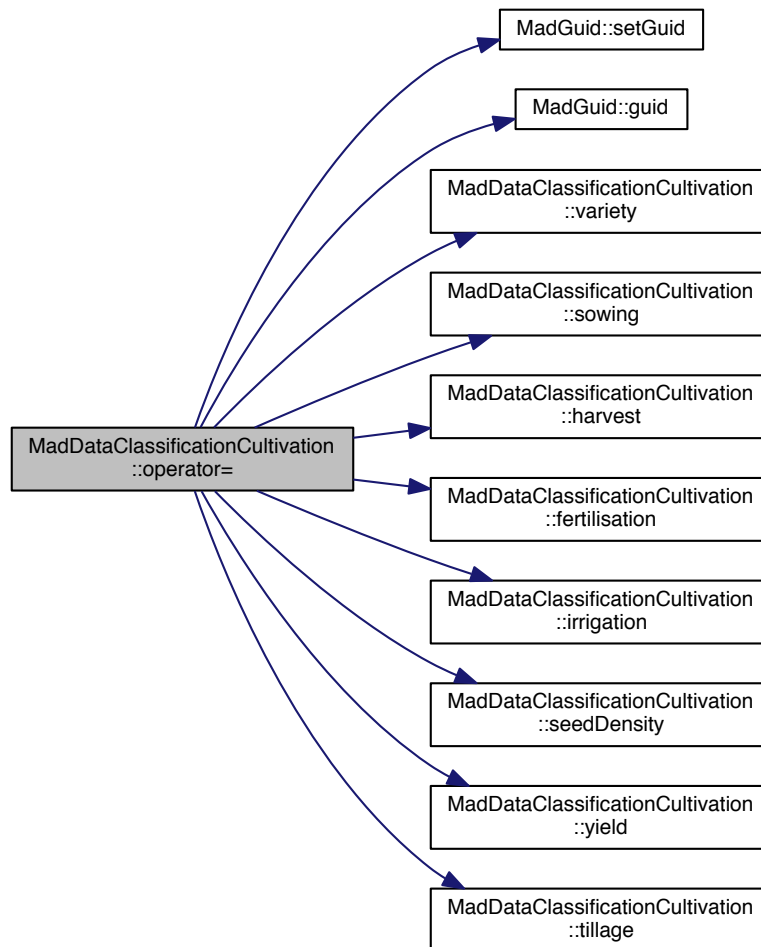
```
52 {
53     // gracefully handles self assignment
54     if (this == &theData) return *this;
55     setGuid(theData.guid());
56     mVariety=theData.variety();
57     mSowing=theData.sowing();
58     mHarvest=theData.harvest();
59     mFertilisation=theData.fertilisation();
60     mIrrigation=theData.irrigation();
  }
```

```

61     mSeedDensity=theData.seedDensity();
62     mYield=theData.yield();
63     mTillage=theData.tillage();
64     return *this;
65 }

```

Here is the call graph for this function:



### 6.3.3.8 MadSubCategory MadDataClassificationCultivation::seedDensity ( ) const

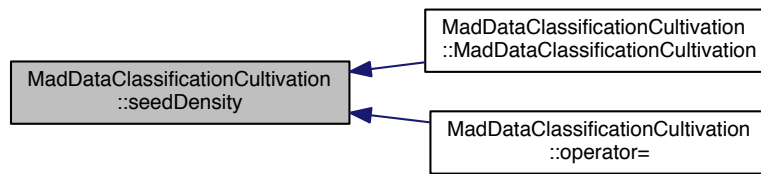
Definition at line 88 of file `maddataclassificationcultivation.cpp`.

```

89 {
90     return mSeedDensity;
91 }

```

Here is the caller graph for this function:



#### 6.3.3.9 void MadDataClassificationCultivation::setFertilisation ( MadSubCategory *theData* )

Definition at line 114 of file `maddataclassificationcultivation.cpp`.

```

115 {
116     mFertilisation = theData;
117 }
```

Here is the caller graph for this function:



#### 6.3.3.10 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid](#).

##### Parameters

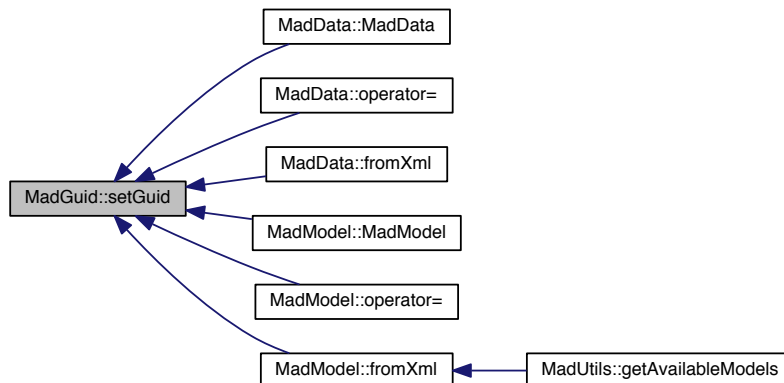
<i>theGuid</i>	
----------------	--

Definition at line 49 of file `madguid.cpp`.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



#### 6.3.3.11 void MadDataClassificationCultivation::setHarvest ( MadSubCategory *theData* )

Definition at line 110 of file `maddataclassificationcultivation.cpp`.

```

111 {
112     mHarvest = theData;
113 }
  
```

Here is the caller graph for this function:



#### 6.3.3.12 void MadDataClassificationCultivation::setIrrigation ( MadSubCategory *theData* )

Definition at line 118 of file `maddataclassificationcultivation.cpp`.

```

119 {
120     mIrrigation = theData;
121 }
  
```

Here is the caller graph for this function:



**6.3.3.13 void MadDataClassificationCultivation::setSeedDensity ( MadSubCategory *theData* )**

Definition at line 122 of file maddataclassificationcultivation.cpp.

```
123 {  
124     mSeedDensity = theData;  
125 }
```

Here is the caller graph for this function:

**6.3.3.14 void MadDataClassificationCultivation::setSowing ( MadSubCategory *theData* )**

Definition at line 106 of file maddataclassificationcultivation.cpp.

```
107 {  
108     mSowing = theData;  
109 }
```

Here is the caller graph for this function:

**6.3.3.15 void MadDataClassificationCultivation::setTillage ( MadSubCategory *theData* )**

Definition at line 130 of file maddataclassificationcultivation.cpp.

```
131 {  
132     mTillage = theData;  
133 }
```

Here is the caller graph for this function:



#### 6.3.3.16 void MadDataClassificationCultivation::setVariety ( MadSubCategory *theData* )

Definition at line 102 of file maddataclassificationcultivation.cpp.

```
103 {  
104     mVariety = theData;  
105 }
```

Here is the caller graph for this function:



#### 6.3.3.17 void MadDataClassificationCultivation::setYield ( MadSubCategory *theData* )

Definition at line 126 of file maddataclassificationcultivation.cpp.

```
127 {  
128     mYield = theData;  
129 }
```

Here is the caller graph for this function:



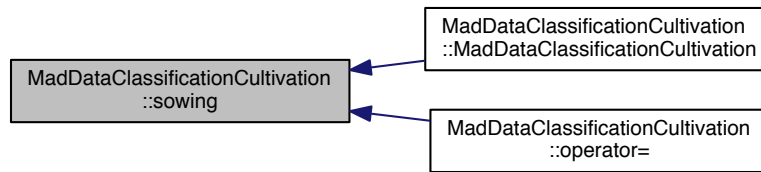
#### 6.3.3.18 MadSubCategory MadDataClassificationCultivation::sowing ( ) const

Definition at line 72 of file maddataclassificationcultivation.cpp.

```
73 {  
74     return mSowing;  
75 }
```



Here is the caller graph for this function:



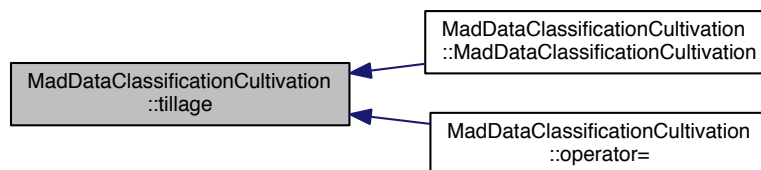
#### 6.3.3.19 MadSubCategory MadDataClassificationCultivation::tillage ( ) const

Definition at line 96 of file `maddataclassificationcultivation.cpp`.

```

97 {
98     return mTillage;
99 }
  
```

Here is the caller graph for this function:



#### 6.3.3.20 QString MadDataClassificationCultivation::toHtml ( )

Return a html text representation of this layer

Definition at line 206 of file `maddataclassificationcultivation.cpp`.

```

207 {
208     QString myString;
209     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
210     //myString+="<p>GUID: " + guid() + "</p>";
211     myString+="<table>";
212     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
213
214     //
215     // the following shows example of how to do a couple of things
216     //
217
218     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
219     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
220     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
221     //myString+="<tr><td><b>Fodder (kg/" + myUnits + ") : </b></td><td>" +
222     //    QString::number(mCropFodderProduction) + "</td></tr>";
222     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
223     //    "</td></tr>";
223     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
  
```

```

224 //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
225 myString+="</table>";
226 return myString;
227 }

```

### 6.3.3.21 QString MadDataClassificationCultivation::toText ( )

Return a plain text representation of this layer

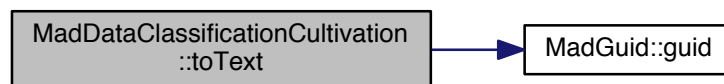
Definition at line 197 of file maddataclassificationcultivation.cpp.

```

198 {
199     QString myString;
200     myString+=QString("guid=>" + guid() + "\n");
201     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
202     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
203     return myString;
204 }

```

Here is the call graph for this function:



### 6.3.3.22 QString MadDataClassificationCultivation::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 155 of file maddataclassificationcultivation.cpp.

```

156 {
157     QString myString;
158     myString+=QString(" <cultivation guid=\"\" + guid() + "\">\n");
159
160     myString+=QString(" <variety>\n");
161     myString+=mVariety.toXml();
162     myString+=QString(" </variety>\n");
163
164     myString+=QString(" <sowing>\n");
165     myString+=mSowing.toXml();
166     myString+=QString(" </sowing>\n");
167
168     myString+=QString(" <harvest>\n");
169     myString+=mHarvest.toXml();
170     myString+=QString(" </harvest>\n");
171
172     myString+=QString(" <fertilisation>\n");
173     myString+=mFertilisation.toXml();
174     myString+=QString(" </fertilisation>\n");
175
176     myString+=QString(" <irrigation>\n");
177     myString+=mIrrigation.toXml();
178     myString+=QString(" </irrigation>\n");

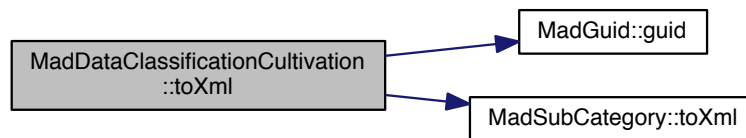
```

```

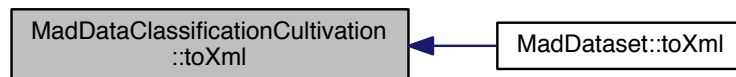
179
180 myString+=QString("    <seeddensity>\n");
181 myString+=mSeedDensity.toXml();
182 myString+=QString("    </seeddensity>\n");
183
184 myString+=QString("    <yield>\n");
185 myString+=mYield.toXml();
186 myString+=QString("    </yield>\n");
187
188 myString+=QString("    <tillage>\n");
189 myString+=mTillage.toXml();
190 myString+=QString("    </tillage>\n");
191
192 myString+=QString("  </cultivation>\n");
193
194 return myString;
195 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.3.3.23 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }

```

Here is the call graph for this function:

**6.3.3.24 MadSubCategory MadDataClassificationCultivation::variety ( ) const**

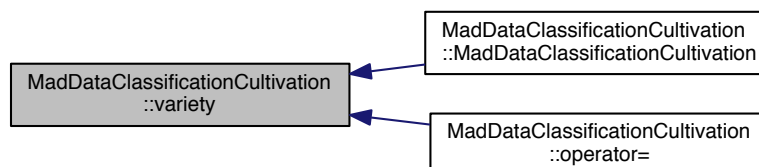
Definition at line 68 of file maddataclassificationcultivation.cpp.

```

69 {
70     return mVariety;
71 }

```

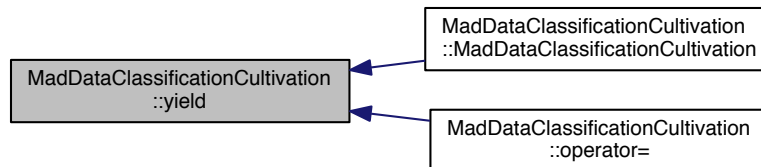
Here is the caller graph for this function:

**6.3.3.25 MadSubCategory MadDataClassificationCultivation::yield ( ) const**

Definition at line 92 of file maddataclassificationcultivation.cpp.

```
93 {  
94     return mYield;  
95 }
```

Here is the caller graph for this function:



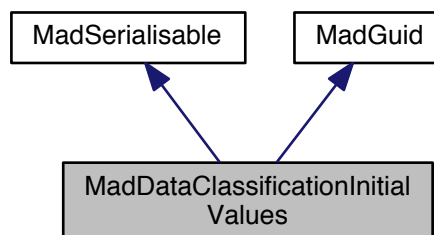
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.-h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.-cpp](#)

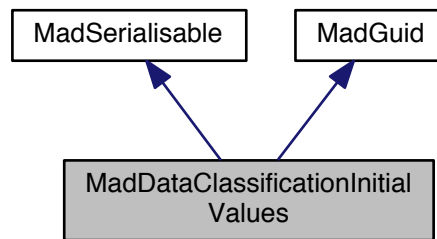
## 6.4 MadDataClassificationInitialValues Class Reference

```
#include <maddataclassificationinitialvalues.h>
```

Inheritance diagram for MadDataClassificationInitialValues:



Collaboration diagram for MadDataClassificationInitialValues:



## Public Member Functions

- [MadDataClassificationInitialValues](#) ()
- [MadDataClassificationInitialValues](#) (const [MadDataClassificationInitialValues](#) &theData)
- [MadDataClassificationInitialValues](#) & operator= (const [MadDataClassificationInitialValues](#) &theData)
- [MadSubCategory](#) soilMoisture () const
- [MadSubCategory](#) nitrogenMin () const
- QString [toXml](#) ()
- QString [toText](#) ()
- QString [toHtml](#) ()
- bool [fromXml](#) (const QString theXml)
- void [setSoilMoisture](#) ([MadSubCategory](#) theData)
- void [setNitrogenMin](#) ([MadSubCategory](#) theData)
- virtual bool [toXmlFile](#) (const QString theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- virtual bool [fromXmlFile](#) (const QString theFileName)

*fromXmlFile Read this object from xml in a file*

- QString [guid](#) () const  
*[MadGuid::guid](#).*
- void [setGuid](#) (QString theGuid="")  
*[MadGuid::setGuid](#).*

### 6.4.1 Detailed Description

Definition at line 36 of file maddataclassificationinitialvalues.h.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 MadDataClassificationInitialValues::MadDataClassificationInitialValues ( )

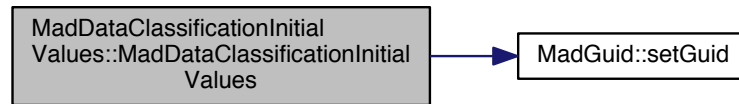
Definition at line 34 of file maddataclassificationinitialvalues.cpp.

```

34                                     :
35     MadSerializable(), MadGuid()
36 {
37     setGuid();
38 }

```

Here is the call graph for this function:



#### 6.4.2.2 MadDataClassificationInitialValues::MadDataClassificationInitialValues ( const MadDataClassificationInitialValues & theData )

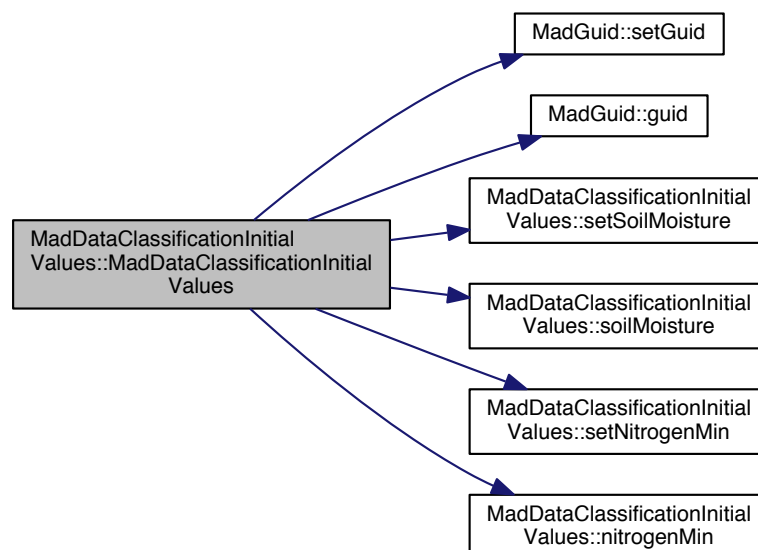
Definition at line 39 of file maddataclassificationinitialvalues.cpp.

```

40 {
41     setGuid(theData.guid());
42     setSoilMoisture(theData.soilMoisture());
43     setNitrogenMin(theData.nitrogenMin());
44 }

```

Here is the call graph for this function:



#### 6.4.3 Member Function Documentation

#### 6.4.3.1 bool MadDataClassificationInitialValues::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

##### See Also

[MadSerialisable](#)

##### Note

this class inherits the serialisable interface so it MUST implement this

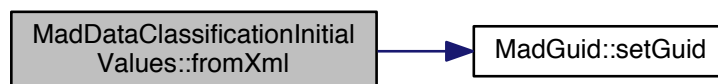
Implements [MadSerialisable](#).

Definition at line 77 of file maddataclassificationinitialvalues.cpp.

```

78 {
79     QDomDocument myDocument("mydocument");
80     myDocument.setContent(theXml);
81     QDomElement myTopElement = myDocument.firstChildElement("initialvalues");
82     if (myTopElement.isNull())
83     {
84         //TODO - just make this a warning
85         qDebug("the top element couldn't be found!");
86         setGuid(myTopElement.attribute("guid"));
87         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
88         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
89         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
90         return true;
91     }
92     else
93         return false;
94 }
```

Here is the call graph for this function:



#### 6.4.3.2 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

##### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

##### Parameters

<i>theFileName</i>	
--------------------	--



**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:

**6.4.3.3 QString MadGuid::guid ( ) const [inherited]**

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

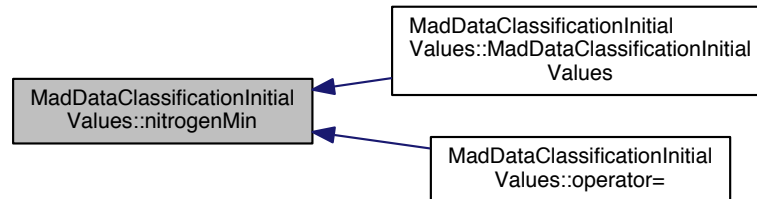
**6.4.3.4 MadSubCategory MadDataClassificationInitialValues::nitrogenMin ( ) const**

Definition at line 60 of file maddataclassificationinitialvalues.cpp.

```

61 {
62     return mNitrogenMin;
63 }
```

Here is the caller graph for this function:



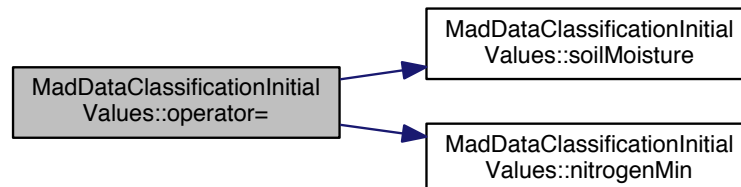
#### 6.4.3.5 MadDataClassificationInitialValues & MadDataClassificationInitialValues::operator= ( const MadDataClassificationInitialValues & theData )

Definition at line 46 of file maddataclassificationinitialvalues.cpp.

```

47 {
48     // gracefully handles self assignment
49     if (this == &theData) return *this;
50     mSoilMoisture = theData.soilMoisture();
51     mNitrogenMin = theData.nitrogenMin();
52     return *this;
53 }
  
```

Here is the call graph for this function:



#### 6.4.3.6 void MadGuid::setGuid ( QString theGuid = "" ) [inherited]

[MadGuid::setGuid](#).

##### Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

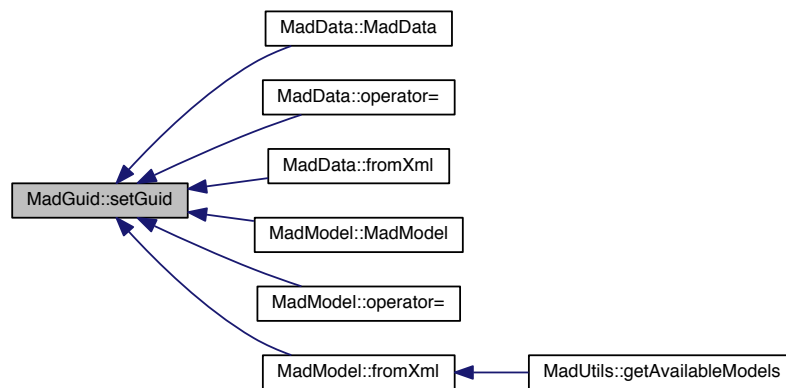
50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
  
```

```

54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:



#### 6.4.3.7 void MadDataClassificationInitialValues::setNitrogenMin ( MadSubCategory *theData* )

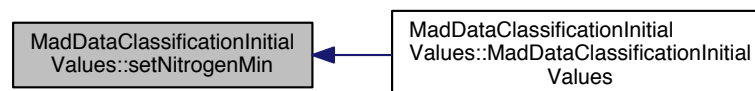
Definition at line 71 of file `maddataclassificationinitialvalues.cpp`.

```

72 {
73     mNitrogenMin = theData;
74 }

```

Here is the caller graph for this function:



#### 6.4.3.8 void MadDataClassificationInitialValues::setSoilMoisture ( MadSubCategory *theData* )

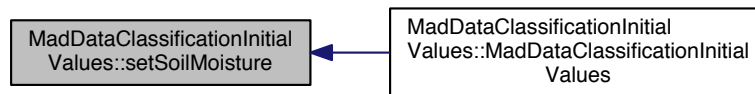
Definition at line 66 of file `maddataclassificationinitialvalues.cpp`.

```

67 {
68     mSoilMoisture = theData;
69 }

```

Here is the caller graph for this function:



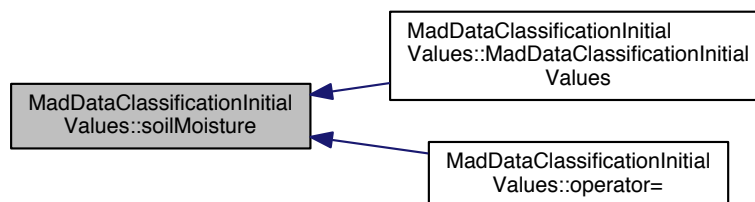
#### 6.4.3.9 MadSubCategory MadDataClassificationInitialValues::soilMoisture ( ) const

Definition at line 56 of file `maddataclassificationinitialvalues.cpp`.

```

57 {
58     return mSoilMoisture;
59 }
  
```

Here is the caller graph for this function:



#### 6.4.3.10 QString MadDataClassificationInitialValues::toHtml ( )

Return a html text representation of this layer

Definition at line 123 of file `maddataclassificationinitialvalues.cpp`.

```

124 {
125     QString myString;
126     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
127     //myString+="<p>GUID: " + guid() + "</p>";
128     myString+="<table>";
129     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
130
131     //
132     // the following shows example of how to do a couple of things
133     //
134
135     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
136     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
137     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
138     //myString+="<tr><td><b>Fodder (kg/" + myUnits + ")</b></td><td>" +
139     //    QString::number(mCropFodderProduction) + "</td></tr>";
139     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
140     //    "</td></tr>";
140     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
141     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
142     myString+="</table>";
143     return myString;
144 }
  
```

## 6.4.3.11 QString MadDataClassificationInitialValues::toText ( )

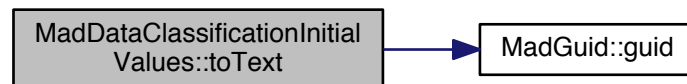
Return a plain text representation of this layer

Definition at line 114 of file maddataclassificationinitialvalues.cpp.

```

115 {
116     QString myString;
117     myString+=QString("guid=>" + guid() + "\n");
118     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
119     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
120     return myString;
121 }
```

Here is the call graph for this function:



## 6.4.3.12 QString MadDataClassificationInitialValues::toXml ( ) [virtual]

Return an xml representation of this layer

**Note**

this class inherits the serialisable interface so it MUST implement this

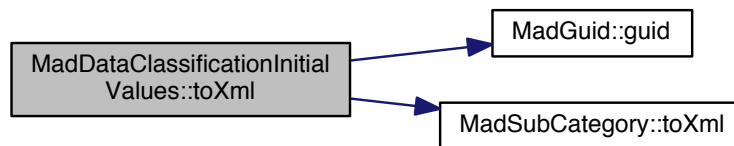
Implements [MadSerialisable](#).

Definition at line 96 of file maddataclassificationinitialvalues.cpp.

```

97 {
98     QString myString;
99     myString+=QString(" <initialvalues guid=\"\" + guid() + "\">\n");
100
101     myString+=QString("     <soilmoisture>\n");
102     myString+=mSoilMoisture.toXml();
103     myString+=QString("     </soilmoisture>\n");
104
105     myString+=QString("     <nmin>\n");
106     myString+=mNitrogenMin.toXml();
107     myString+=QString("     </nmin>\n");
108
109     myString+=QString(" </initialvalues>\n");
110     return myString;
111 }
112 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.4.3.13 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {

```

```
70     //@TODO Error handler!  
71     myResult=false;  
72 }  
73 return myResult ;  
74 }
```

Here is the call graph for this function:



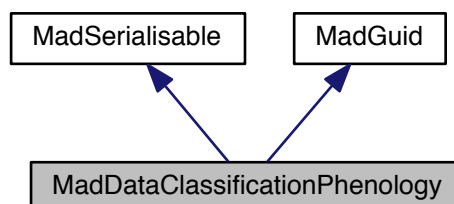
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.cpp](#)

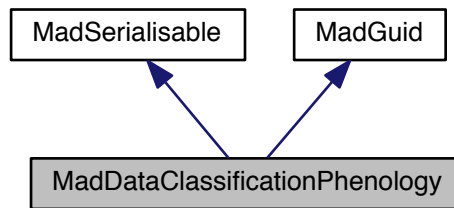
## 6.5 MadDataClassificationPhenology Class Reference

```
#include <maddataclassificationphenology.h>
```

Inheritance diagram for MadDataClassificationPhenology:



Collaboration diagram for MadDataClassificationPhenology:



## Public Member Functions

- [MadDataClassificationPhenology](#) ()
- [MadDataClassificationPhenology](#) (const [MadDataClassificationPhenology](#) &theData)
- [MadDataClassificationPhenology](#) & [operator=](#) (const [MadDataClassificationPhenology](#) &theData)
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- [bool](#) [fromXml](#) (const [QString](#) theXml)
- [MadSubCategory](#) [emergence](#) () const
- [MadSubCategory](#) [stemElongation](#) () const
- [MadSubCategory](#) [earEmergence](#) () const
- [MadSubCategory](#) [flowering](#) () const
- [MadSubCategory](#) [yellowRipeness](#) () const
- [void](#) [setEmergence](#) ([MadSubCategory](#) theData)
- [void](#) [setStemElongation](#) ([MadSubCategory](#) theData)
- [void](#) [setEarEmergence](#) ([MadSubCategory](#) theData)
- [void](#) [setFlowering](#) ([MadSubCategory](#) theData)
- [void](#) [setYellowRipeness](#) ([MadSubCategory](#) theData)
- [virtual bool](#) [toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- [virtual bool](#) [fromXmlFile](#) (const [QString](#) theFileName)

*fromXmlFile Read this object from xml in a file*

- [QString](#) [guid](#) () const  
*[MadGuid::guid](#).*
- [void](#) [setGuid](#) ([QString](#) theGuid="")  
*[MadGuid::setGuid](#).*

### 6.5.1 Detailed Description

Definition at line 36 of file maddataclassificationphenology.h.



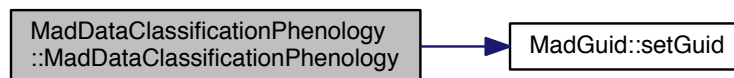
## 6.5.2 Constructor & Destructor Documentation

### 6.5.2.1 MadDataClassificationPhenology::MadDataClassificationPhenology ( )

Definition at line 33 of file maddataclassificationphenology.cpp.

```
33                                     : MadSerialisable(),
34   MadGuid()
35 {
36   setGuid();
37 }
```

Here is the call graph for this function:

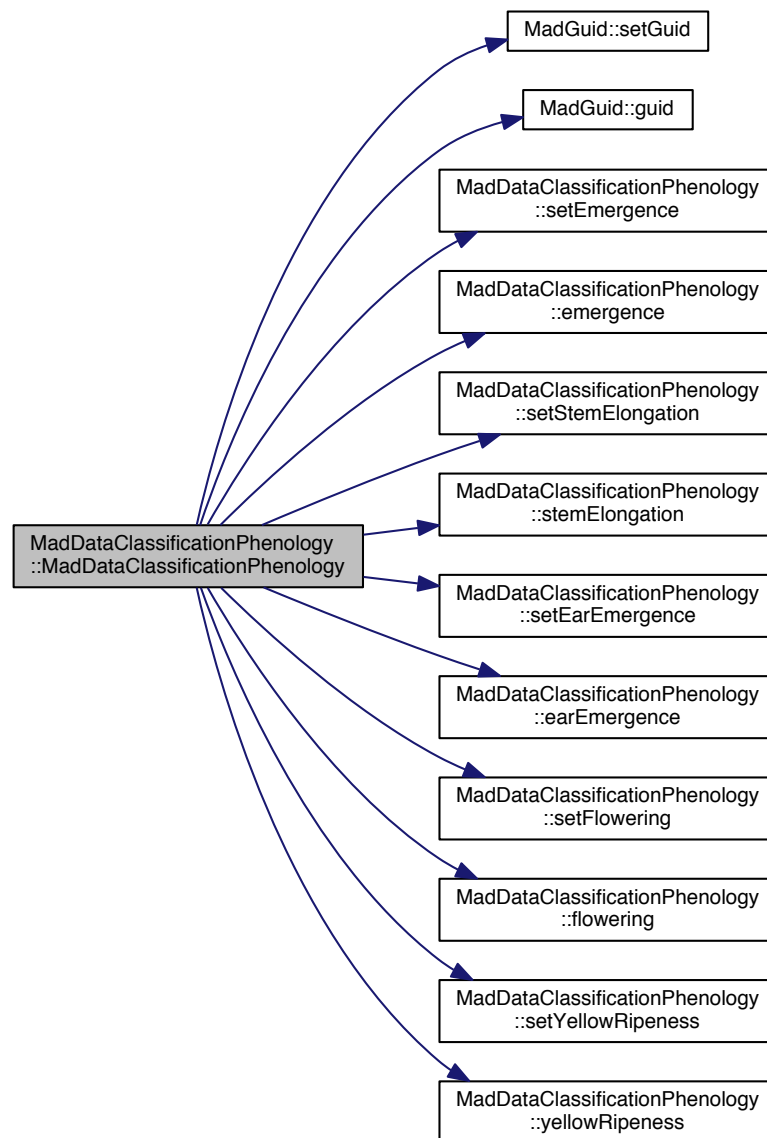


### 6.5.2.2 MadDataClassificationPhenology::MadDataClassificationPhenology ( const MadDataClassificationPhenology & theData )

Definition at line 38 of file maddataclassificationphenology.cpp.

```
39 {
40   setGuid(theData.guid());
41   setEmergence(theData.emergence());
42   setStemElongation(theData.stemElongation());
43   setEarEmergence(theData.earEmergence());
44   setFlowering(theData.flowering());
45   setYellowRipeness(theData.yellowRipeness());
46 }
```

Here is the call graph for this function:



### 6.5.3 Member Function Documentation

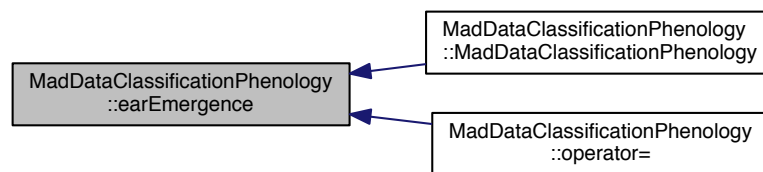
#### 6.5.3.1 MadSubCategory MadDataClassificationPhenology::earEmergence ( ) const

Definition at line 69 of file maddataclassificationphenology.cpp.

```

70 {
71     return mEarEmergence;
72 }
  
```

Here is the caller graph for this function:

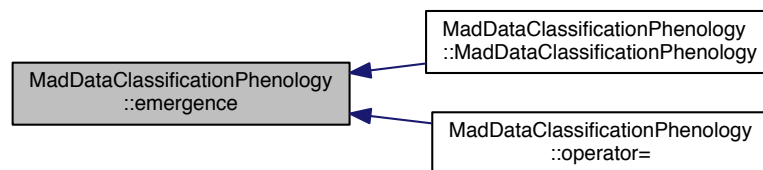


### 6.5.3.2 MadSubCategory MadDataClassificationPhenology::emergence ( ) const

Definition at line 61 of file `maddataclassificationphenology.cpp`.

```
62 {  
63     return mEmergence;  
64 }
```

Here is the caller graph for this function:

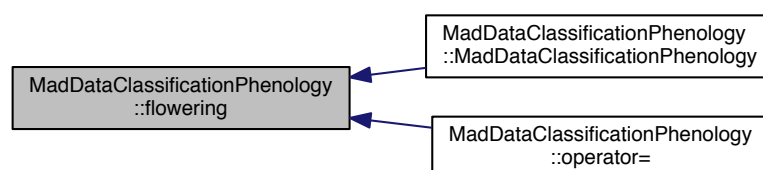


### 6.5.3.3 MadSubCategory MadDataClassificationPhenology::flowering ( ) const

Definition at line 73 of file `maddataclassificationphenology.cpp`.

```
74 {  
75     return mFlowering;  
76 }
```

Here is the caller graph for this function:



#### 6.5.3.4 bool MadDataClassificationPhenology::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

##### See Also

[MadSerialisable](#)

##### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

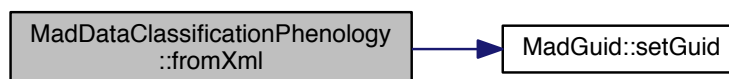
Definition at line 109 of file maddataclassificationphenology.cpp.

```

110 {
111     QDomDocument myDocument("mydocument");
112     myDocument.setContent(theXml);
113     QDomElement myTopElement = myDocument.firstChildElement("phenology");
114     if (myTopElement.isNull())
115     {
116         //TODO - just make this a warning
117         qDebug("the top element couldn't be found!");
118         setGuid(myTopElement.attribute("guid"));
119         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
120         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
121         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
122         return true;
123     }
124     else
125         return false;
126 }

```

Here is the call graph for this function:



#### 6.5.3.5 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

##### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

##### Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:

**6.5.3.6 QString MadGuid::guid ( ) const [inherited]**

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

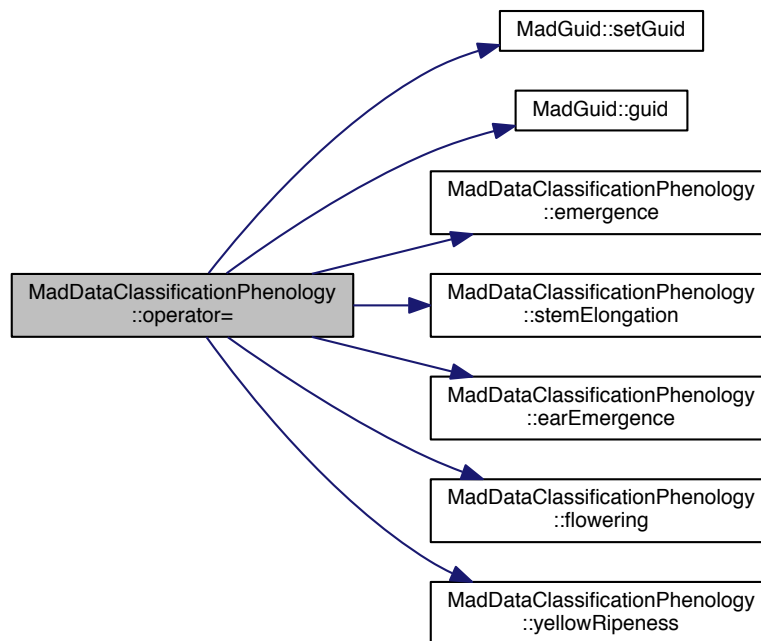
**6.5.3.7 MadDataClassificationPhenology & MadDataClassificationPhenology::operator= ( const MadDataClassificationPhenology & theData )**

Definition at line 48 of file maddataclassificationphenology.cpp.

```

49 {
50     // gracefully handles self assignment
51     if (this == &theData) return *this;
52     setGuid(theData.guid());
53     mEmergence=theData.emergence();
54     mStemElongation=theData.stemElongation();
55     mEarEmergence=theData.earEmergence();
56     mFlowering=theData.flowering();
57     mYellowRipeness=theData.yellowRipeness();
58     return *this;
59 }
```

Here is the call graph for this function:



#### 6.5.3.8 void MadDataClassificationPhenology::setEarEmergence ( MadSubCategory *theData* )

Definition at line 93 of file `maddataclassificationphenology.cpp`.

```

94 {
95     mEarEmergence = theData;
96 }
  
```

Here is the caller graph for this function:



#### 6.5.3.9 void MadDataClassificationPhenology::setEmergence ( MadSubCategory *theData* )

Definition at line 83 of file `maddataclassificationphenology.cpp`.

```

84 {
85     mEmergence = theData;
86 }
  
```

Here is the caller graph for this function:



#### 6.5.3.10 void MadDataClassificationPhenology::setFlowering ( MadSubCategory *theData* )

Definition at line 98 of file `maddataclassificationphenology.cpp`.

```

99 {
100     mFlowering = theData;
101 }
```

Here is the caller graph for this function:



#### 6.5.3.11 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

##### Parameters

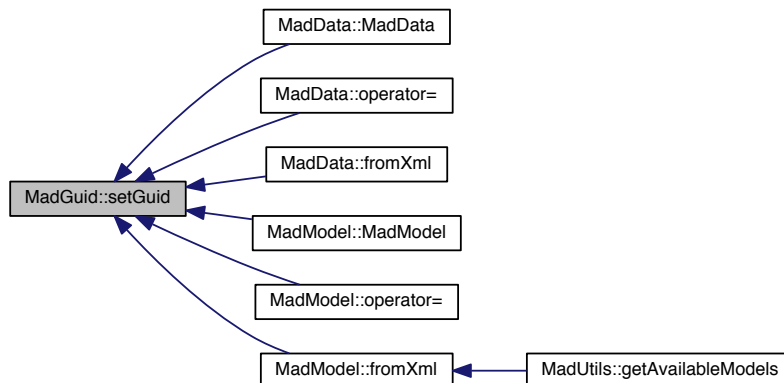
<i>theGuid</i>	
----------------	--

Definition at line 49 of file `madguid.cpp`.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



#### 6.5.3.12 void MadDataClassificationPhenology::setStemElongation ( MadSubCategory *theData* )

Definition at line 88 of file `maddataclassificationphenology.cpp`.

```

89 {
90     mStemElongation = theData;
91 }

```

Here is the caller graph for this function:



#### 6.5.3.13 void MadDataClassificationPhenology::setYellowRipeness ( MadSubCategory *theData* )

Definition at line 103 of file `maddataclassificationphenology.cpp`.

```

104 {
105     mYellowRipeness = theData;
106 }

```

Here is the caller graph for this function:



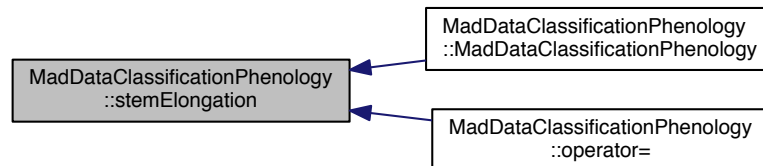


#### 6.5.3.14 MadSubCategory MadDataClassificationPhenology::stemElongation ( ) const

Definition at line 65 of file maddataclassificationphenology.cpp.

```
66 {
67     return mStemElongation;
68 }
```

Here is the caller graph for this function:



#### 6.5.3.15 QString MadDataClassificationPhenology::toHtml ( )

Return a html text representation of this layer

Definition at line 166 of file maddataclassificationphenology.cpp.

```
167 {
168     QString myString;
169     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
170     //myString+="<p>GUID: " + guid() + "</p>";
171     myString+="<table>";
172     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
173
174     //
175     // the following shows example of how to do a couple of things
176     //
177
178     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
179     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
180     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
181     //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
182     //    QString::number(mCropFodderProduction) + "</td></tr>";
183     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
184     //    "</td></tr>";
185     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
186     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
187     myString+="</table>";
188     return myString;
189 }
```

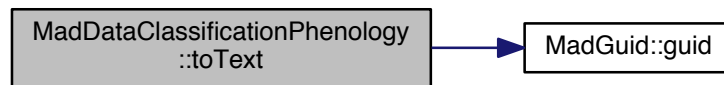
#### 6.5.3.16 QString MadDataClassificationPhenology::toText ( )

Return a plain text representation of this layer

Definition at line 157 of file maddataclassificationphenology.cpp.

```
158 {
159     QString myString;
160     myString+=QString("guid=>" + guid() + "\n");
161     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
162     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
163     return myString;
164 }
```

Here is the call graph for this function:



### 6.5.3.17 QString MadDataClassificationPhenology::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

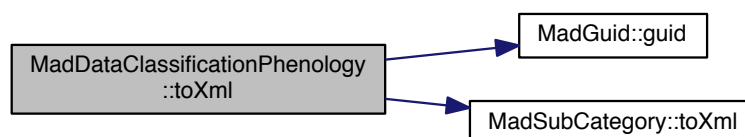
Implements [MadSerialisable](#).

Definition at line 128 of file maddataclassificationphenology.cpp.

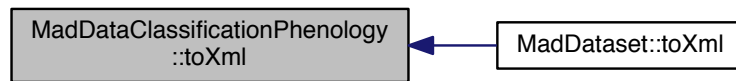
```

129 {
130     QString myString;
131     myString+=QString(" <phenology guid=\"\" + guid() + "\">\n");
132
133     myString+=QString(" <emergence>\n");
134     myString+=mEmergence.toXml();
135     myString+=QString(" </emergence>\n");
136
137     myString+=QString(" <stemelongation>\n");
138     myString+=mStemElongation.toXml();
139     myString+=QString(" </stemelongation>\n");
140
141     myString+=QString(" <earemergence>\n");
142     myString+=mEarEmergence.toXml();
143     myString+=QString(" </earemergence>\n");
144
145     myString+=QString(" <flowering>\n");
146     myString+=mFlowering.toXml();
147     myString+=QString(" </flowering>\n");
148
149     myString+=QString(" <yellowripeness>\n");
150     myString+=mYellowRipeness.toXml();
151     myString+=QString(" </yellowripeness>\n");
152
153     myString+=QString(" </phenology>\n");
154     return myString;
155 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.5.3.18 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:

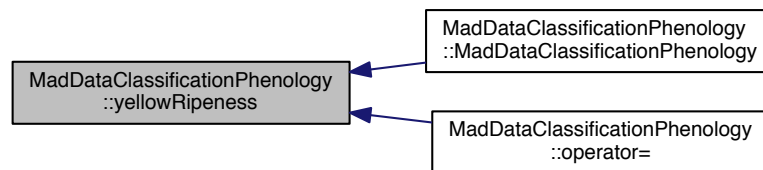


### 6.5.3.19 MadSubCategory MadDataClassificationPhenology::yellowRipeness ( ) const

Definition at line 77 of file maddataclassificationphenology.cpp.

```
78 {
79     return mYellowRipeness;
80 }
```

Here is the caller graph for this function:



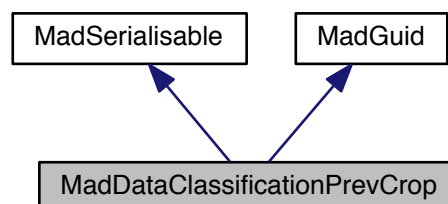
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.cpp](#)

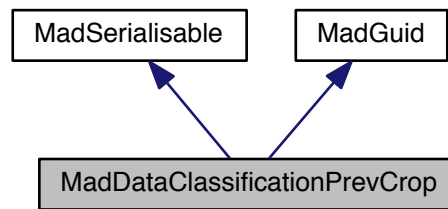
## 6.6 MadDataClassificationPrevCrop Class Reference

```
#include <maddataclassificationprevcrop.h>
```

Inheritance diagram for `MadDataClassificationPrevCrop`:



Collaboration diagram for MadDataClassificationPrevCrop:



## Public Member Functions

- [MadDataClassificationPrevCrop](#) ()
- [MadDataClassificationPrevCrop](#) (const [MadDataClassificationPrevCrop](#) &theData)
- [MadDataClassificationPrevCrop](#) & operator= (const [MadDataClassificationPrevCrop](#) &theData)
- [MadSubCategory crop](#) () const
- [MadSubCategory sowingDate](#) () const
- [MadSubCategory harvestDate](#) () const
- [MadSubCategory yield](#) () const
- [MadSubCategory residueMgmt](#) () const
- [MadSubCategory fertilisation](#) () const
- [MadSubCategory irrigation](#) () const
- [QString toXml](#) ()
- [QString toText](#) ()
- [QString toHtml](#) ()
- bool [fromXml](#) (const [QString](#) theXml)
- void [setCrop](#) ([MadSubCategory](#) theData)
- void [setSowingDate](#) ([MadSubCategory](#) theData)
- void [setHarvestDate](#) ([MadSubCategory](#) theData)
- void [setYield](#) ([MadSubCategory](#) theData)
- void [setResidueMgmt](#) ([MadSubCategory](#) theData)
- void [setFertilisation](#) ([MadSubCategory](#) theData)
- void [setIrrigation](#) ([MadSubCategory](#) theData)
- virtual bool [toXmlFile](#) (const [QString](#) theFileName)
  - toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*
- virtual bool [fromXmlFile](#) (const [QString](#) theFileName)
  - fromXmlFile Read this object from xml in a file*
- [QString guid](#) () const
  - [MadGuid::guid](#).*
- void [setGuid](#) ([QString](#) theGuid="")
  - [MadGuid::setGuid](#).*

### 6.6.1 Detailed Description

Definition at line 35 of file maddataclassificationprevcrop.h.

## 6.6.2 Constructor & Destructor Documentation

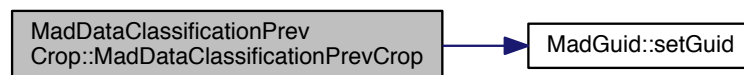
### 6.6.2.1 MadDataClassificationPrevCrop::MadDataClassificationPrevCrop ( )

Definition at line 33 of file maddataclassificationprevcrop.cpp.

```

33                                     : MadSerialisable(),
    MadGuid()
34 {
35     setGuid();
36 }
```

Here is the call graph for this function:



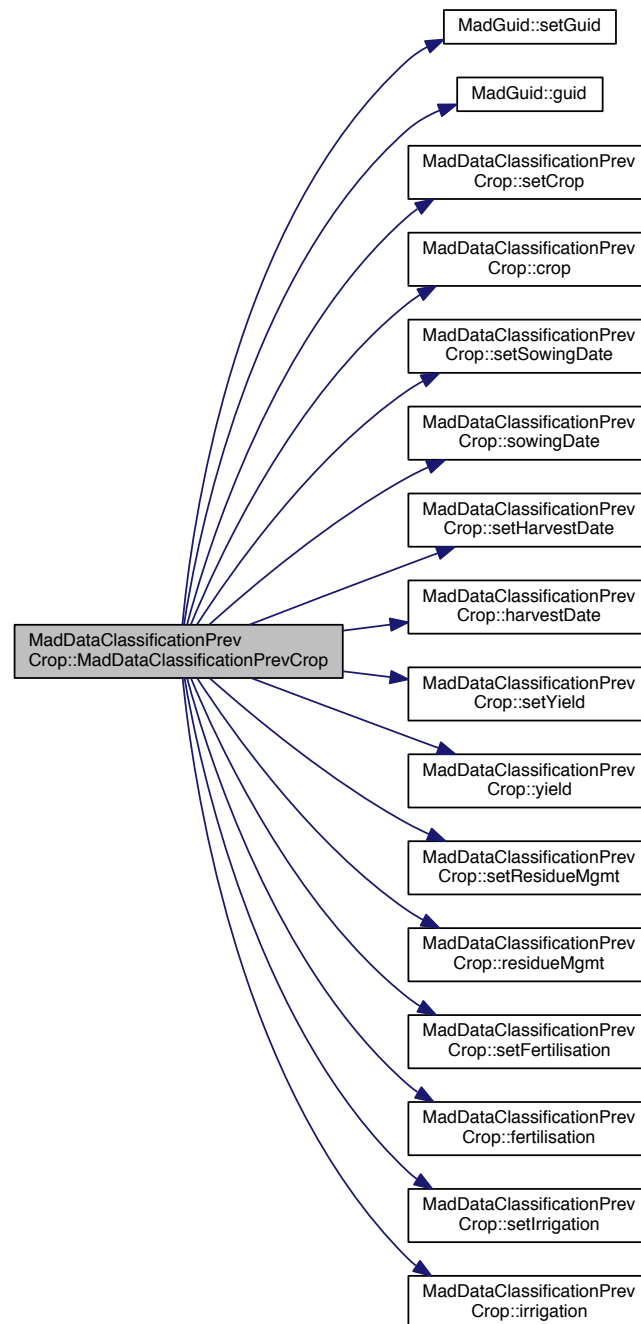
### 6.6.2.2 MadDataClassificationPrevCrop::MadDataClassificationPrevCrop ( const MadDataClassificationPrevCrop & theData )

Definition at line 38 of file maddataclassificationprevcrop.cpp.

```

39 {
40     setGuid(theData.guid());
41     setCrop(theData.crop());
42     setSowingDate(theData.sowingDate());
43     setHarvestDate(theData.harvestDate());
44     setYield(theData.yield());
45     setResidueMgmt(theData.residueMgmt());
46     setFertilisation(theData.fertilisation());
47     setIrrigation(theData.irrigation());
48 }
```

Here is the call graph for this function:



## 6.6.3 Member Function Documentation

### 6.6.3.1 MadSubCategory MadDataClassificationPrevCrop::crop ( ) const

Definition at line 67 of file maddataclassificationprevcrop.cpp.

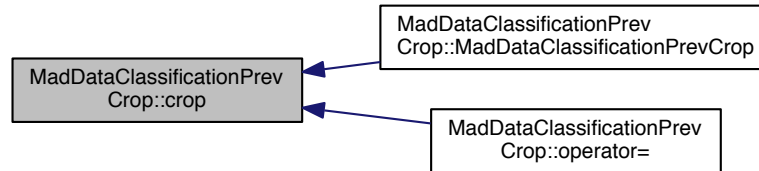
```

68 {
69     return mCrop;

```

```
70 }
```

Here is the caller graph for this function:

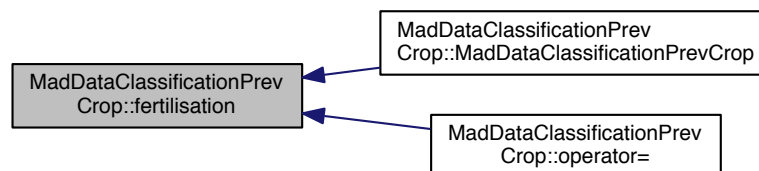


### 6.6.3.2 MadSubCategory MadDataClassificationPrevCrop::fertilisation ( ) const

Definition at line 87 of file `maddataclassificationprevcrop.cpp`.

```
88 {
89     return mFertilisation;
90 }
```

Here is the caller graph for this function:



### 6.6.3.3 bool MadDataClassificationPrevCrop::fromXml ( const QString theXml ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 132 of file `maddataclassificationprevcrop.cpp`.

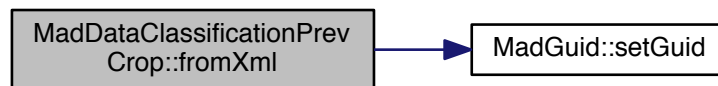


```

133 {
134     QDomDocument myDocument("mydocument");
135     myDocument.setContent(theXml);
136     QDomElement myTopElement = myDocument.firstChildElement("prevcrop");
137     if (myTopElement.isNull())
138     {
139         //TODO - just make this a warning
140         qDebug("the top element couldn't be found!");
141         setGuid(myTopElement.attribute("guid"));
142         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
143         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
144         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
145         return true;
146     }
147     else
148         return false;
149 }

```

Here is the call graph for this function:



#### 6.6.3.4 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

##### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

##### Parameters

<i>theFileName</i>	
--------------------	--

##### Returns

result as true for success, false for failure.

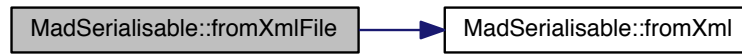
Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }

```

Here is the call graph for this function:



#### 6.6.3.5 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

Returns

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

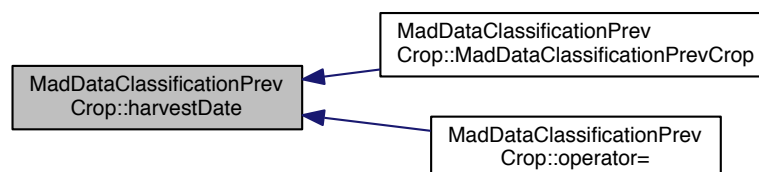
#### 6.6.3.6 MadSubCategory MadDataClassificationPrevCrop::harvestDate ( ) const

Definition at line 75 of file maddataclassificationprevcrop.cpp.

```

76 {
77     return mHarvestDate;
78 }
```

Here is the caller graph for this function:



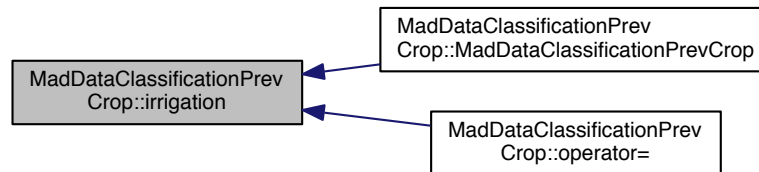
#### 6.6.3.7 MadSubCategory MadDataClassificationPrevCrop::irrigation ( ) const

Definition at line 91 of file maddataclassificationprevcrop.cpp.

```

92 {
93     return mIrrigation;
94 }
```

Here is the caller graph for this function:



#### 6.6.3.8 MadDataClassificationPrevCrop & MadDataClassificationPrevCrop::operator= ( const MadDataClassificationPrevCrop & *theData* )

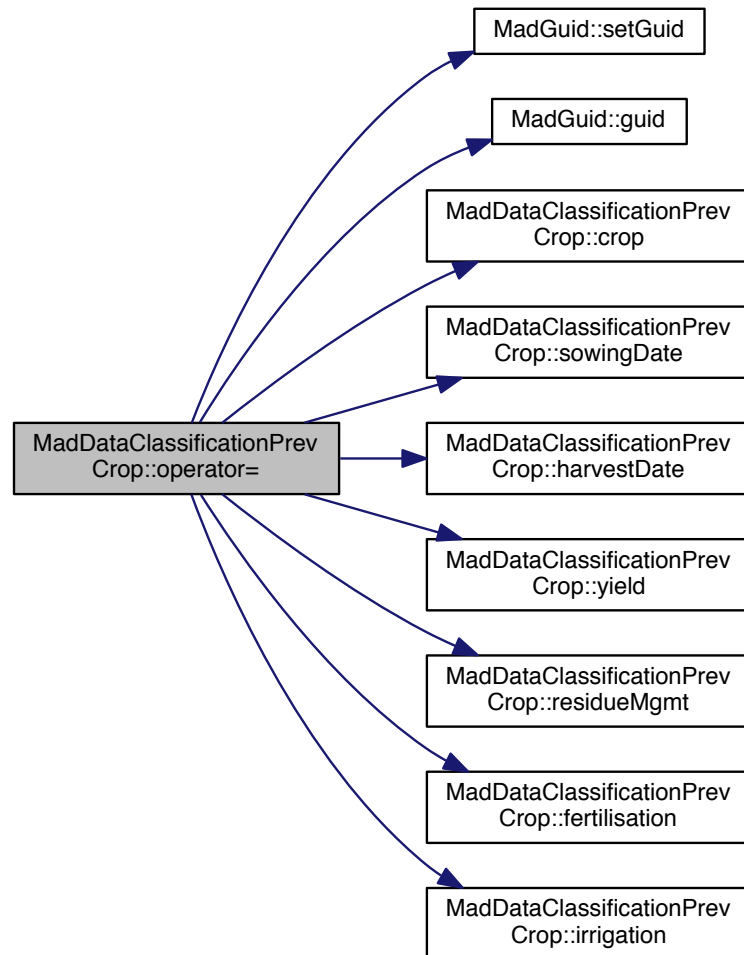
Definition at line 50 of file `maddataclassificationprevcrop.cpp`.

```

51 {
52     // gracefully handles self assignment
53     if (this == &theData) return *this;
54     setGuid(theData.guid());
55     mCrop=theData.crop();
56     mSowingDate=theData.sowingDate();
57     mHarvestDate=theData.harvestDate();
58     mYield=theData.yield();
59     mResidueMgmt=theData.residueMgmt();
60     mFertilisation=theData.fertilisation();
61     mIrrigation=theData.irrigation();
62     return *this;
63 }

```

Here is the call graph for this function:



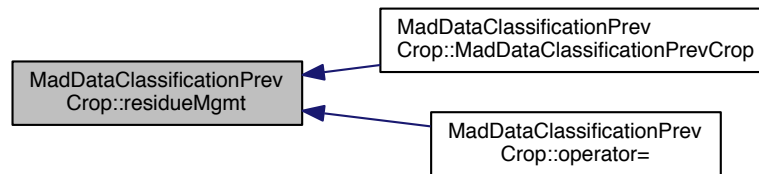
#### 6.6.3.9 MadSubCategory MadDataClassificationPrevCrop::residueMgmt ( ) const

Definition at line 83 of file `maddataclassificationprevcrop.cpp`.

```

84 {
85     return mResidueMgmt;
86 }
  
```

Here is the caller graph for this function:



#### 6.6.3.10 void MadDataClassificationPrevCrop::setCrop ( MadSubCategory *theData* )

Definition at line 97 of file `maddataclassificationprevcrop.cpp`.

```
98 {  
99     mCrop = theData;  
100 }
```

Here is the caller graph for this function:



#### 6.6.3.11 void MadDataClassificationPrevCrop::setFertilisation ( MadSubCategory *theData* )

Definition at line 122 of file `maddataclassificationprevcrop.cpp`.

```
123 {  
124     mFertilisation = theData;  
125 }
```

Here is the caller graph for this function:



### 6.6.3.12 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

#### Parameters

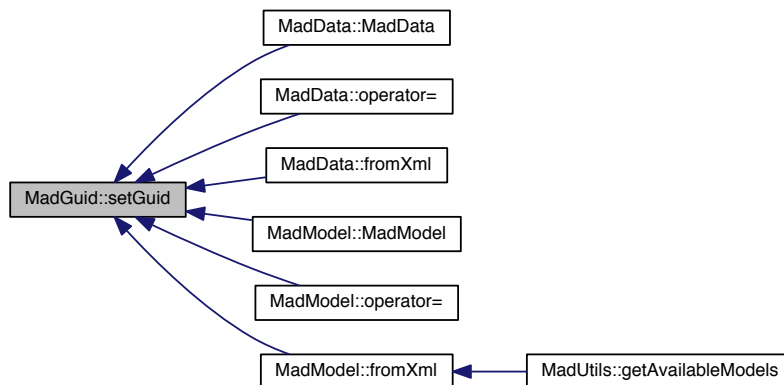
<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



### 6.6.3.13 void MadDataClassificationPrevCrop::setHarvestDate ( MadSubCategory *theData* )

Definition at line 107 of file maddataclassificationprevcrop.cpp.

```

108 {
109     mHarvestDate = theData;
110 }
```

Here is the caller graph for this function:



**6.6.3.14 void MadDataClassificationPrevCrop::setIrrigation ( MadSubCategory *theData* )**

Definition at line 127 of file maddataclassificationprevcrop.cpp.

```
128 {  
129     mIrrigation = theData;  
130 }
```

Here is the caller graph for this function:

**6.6.3.15 void MadDataClassificationPrevCrop::setResidueMgmt ( MadSubCategory *theData* )**

Definition at line 117 of file maddataclassificationprevcrop.cpp.

```
118 {  
119     mResidueMgmt = theData;  
120 }
```

Here is the caller graph for this function:

**6.6.3.16 void MadDataClassificationPrevCrop::setSowingDate ( MadSubCategory *theData* )**

Definition at line 102 of file maddataclassificationprevcrop.cpp.

```
103 {  
104     mSowingDate = theData;  
105 }
```

Here is the caller graph for this function:



### 6.6.3.17 void MadDataClassificationPrevCrop::setYield ( MadSubCategory *theData* )

Definition at line 112 of file maddataclassificationprevcrop.cpp.

```
113 {
114     mYield = theData;
115 }
```

Here is the caller graph for this function:

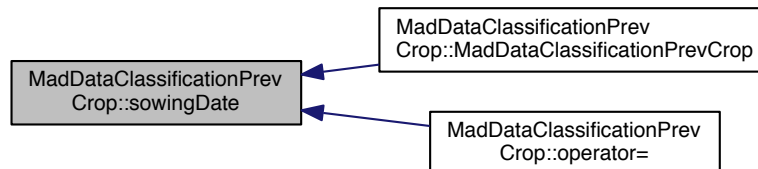


### 6.6.3.18 MadSubCategory MadDataClassificationPrevCrop::sowingDate ( ) const

Definition at line 71 of file maddataclassificationprevcrop.cpp.

```
72 {
73     return mSowingDate;
74 }
```

Here is the caller graph for this function:



### 6.6.3.19 QString MadDataClassificationPrevCrop::toHtml ( )

Return a html text representation of this layer

Definition at line 198 of file maddataclassificationprevcrop.cpp.

```
199 {
200     QString myString;
201     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
202     //myString+="<p>GUID: " + guid() + "</p>";
203     myString+="<table>";
204     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
205
206     //
207     // the following shows example of how to do a couple of things
208     //
209
210     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
```



```

211 //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
212 //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
213 //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
    QString::number(mCropFodderProduction) + "</td></tr>";
214 //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
    "</td></tr>";
215 //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
216 //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
217 myString+="</table>";
218 return myString;
219 }

```

### 6.6.3.20 QString MadDataClassificationPrevCrop::toText ( )

Return a plain text representation of this layer

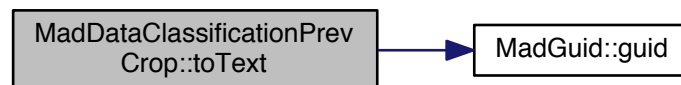
Definition at line 189 of file maddataclassificationprevcrop.cpp.

```

190 {
191     QString myString;
192     myString+=QString("guid=>" + guid() + "\n");
193     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
194     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
195     return myString;
196 }

```

Here is the call graph for this function:



### 6.6.3.21 QString MadDataClassificationPrevCrop::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 151 of file maddataclassificationprevcrop.cpp.

```

152 {
153     QString myString;
154     myString+=QString(" <prevcrop guid=\"\" + guid() + "\">\n");
155
156     myString+=QString(" <crop>\n");
157     myString+=mCrop.toXml();
158     myString+=QString(" </crop>\n");
159
160     myString+=QString(" <sowingdate>\n");
161     myString+=mSowingDate.toXml();
162     myString+=QString(" </sowingdate>\n");
163
164     myString+=QString(" <harvestdate>\n");
165     myString+=mHarvestDate.toXml();
166     myString+=QString(" </harvestdate>\n");

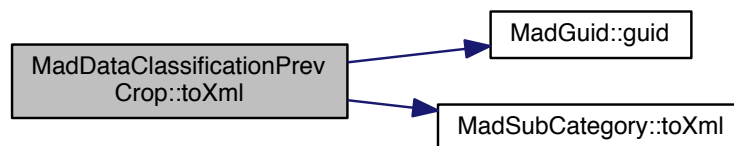
```

```

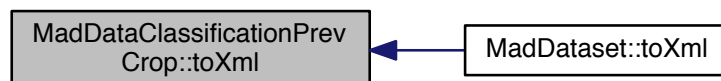
167
168 myString+=QString("    <yield>\n");
169 myString+=mYield.toXml();
170 myString+=QString("    </yield>\n");
171
172 myString+=QString("    <residuegmt>\n");
173 myString+=mResidueMgmt.toXml();
174 myString+=QString("    </residuegmt>\n");
175
176 myString+=QString("    <fertilisation>\n");
177 myString+=mFertilisation.toXml();
178 myString+=QString("    </fertilisation>\n");
179
180 myString+=QString("    <irrigation>\n");
181 myString+=mIrrigation.toXml();
182 myString+=QString("    </irrigation>\n");
183
184 myString+=QString("  </prevcrop>\n");
185 return myString;
186
187 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.6.3.22 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

See Also

[toXml\(\)](#)

Parameters

<i>theFileName</i>	
--------------------	--

## Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //@TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:



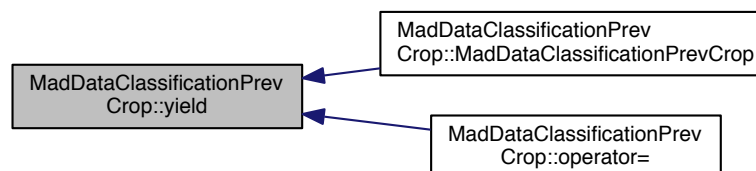
### 6.6.3.23 MadSubCategory MadDataClassificationPrevCrop::yield ( ) const

Definition at line 79 of file maddataclassificationprevcrop.cpp.

```

80 {
81     return mYield;
82 }
```

Here is the caller graph for this function:



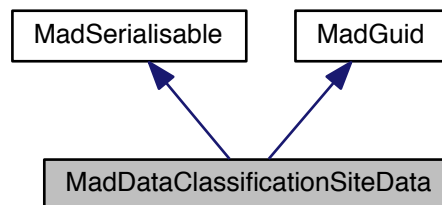
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationprevcrop.cpp](#)

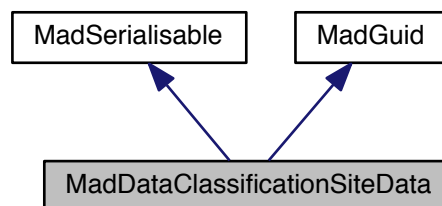
## 6.7 MadDataClassificationSiteData Class Reference

```
#include <maddataclassificationssitedata.h>
```

Inheritance diagram for MadDataClassificationSiteData:



Collaboration diagram for MadDataClassificationSiteData:



### Public Member Functions

- [MadDataClassificationSiteData](#) ()
- [MadDataClassificationSiteData](#) (const [MadDataClassificationSiteData](#) &theData)
- [MadDataClassificationSiteData](#) & [operator=](#) (const [MadDataClassificationSiteData](#) &theData)
- [MadSubCategory latitude](#) () const
- [MadSubCategory longitude](#) () const
- [MadSubCategory altitude](#) () const
- [QString toXml](#) ()
- [QString toText](#) ()
- [QString toHtml](#) ()
- [bool fromXml](#) (const [QString](#) theXml)
- [void setLatitude](#) ([MadSubCategory](#) theData)
- [void setLongitude](#) ([MadSubCategory](#) theData)
- [void setAltitude](#) ([MadSubCategory](#) theData)
- [virtual bool toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile* writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

- virtual bool [fromXmlFile](#) (const QString theFileName)

*fromXmlFile* Read this object from xml in a file

- QString [guid](#) () const

*MadGuid::guid.*

- void [setGuid](#) (QString theGuid="")

*MadGuid::setGuid.*

### 6.7.1 Detailed Description

Definition at line 36 of file maddataclassificationsitedata.h.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 MadDataClassificationSiteData::MadDataClassificationSiteData ( )

Definition at line 33 of file maddataclassificationsitedata.cpp.

```

33                                     : MadSerialisable(),
    MadGuid()
34 {
35     setGuid();
36 }
```

Here is the call graph for this function:



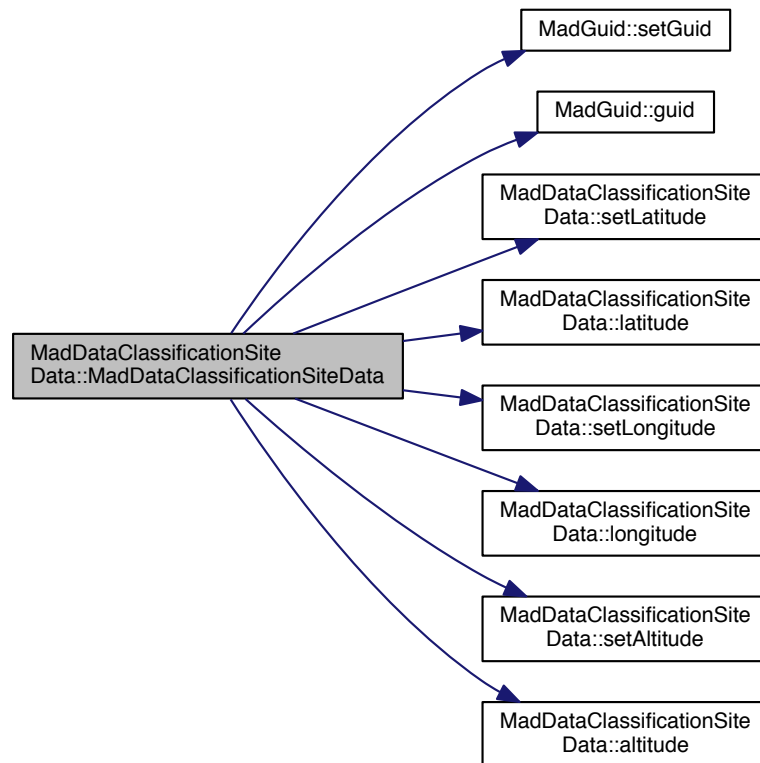
#### 6.7.2.2 MadDataClassificationSiteData::MadDataClassificationSiteData ( const MadDataClassificationSiteData & theData )

Definition at line 38 of file maddataclassificationsitedata.cpp.

```

39 {
40     setGuid(theData.guid());
41     setLatitude(theData.latitude());
42     setLongitude(theData.longitude());
43     setAltitude(theData.altitude());
44 }
```

Here is the call graph for this function:



### 6.7.3 Member Function Documentation

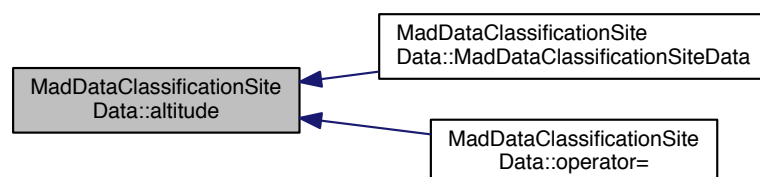
#### 6.7.3.1 MadSubCategory MadDataClassificationSiteData::altitude ( ) const

Definition at line 65 of file `maddataclassificationsitedata.cpp`.

```

66 {
67     return mAltitude;
68 }
  
```

Here is the caller graph for this function:



### 6.7.3.2 bool MadDataClassificationSiteData::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

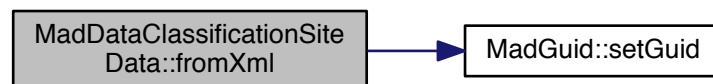
Implements [MadSerialisable](#).

Definition at line 86 of file maddataclassificationssitedata.cpp.

```

87 {
88     QDomDocument myDocument("mydocument");
89     myDocument.setContent(theXml);
90     QDomElement myTopElement = myDocument.firstChildElement("site");
91     if (myTopElement.isNull())
92     {
93         //TODO - just make this a warning
94         qDebug("the top element couldn't be found!");
95         setGuid(myTopElement.attribute("guid"));
96         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
97         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
98         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
99         return true;
100     }
101     else
102         return false;
103 }
```

Here is the call graph for this function:



### 6.7.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:

**6.7.3.4 QString MadGuid::guid ( ) const [inherited]**

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

**6.7.3.5 MadSubCategory MadDataClassificationSiteData::latitude ( ) const**

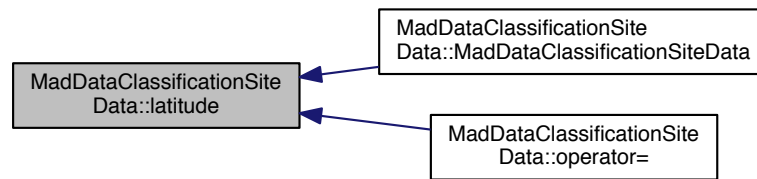
Definition at line 57 of file maddataclassificationssitedata.cpp.

```

58 {
59     return mLatitude;
60 }
```



Here is the caller graph for this function:



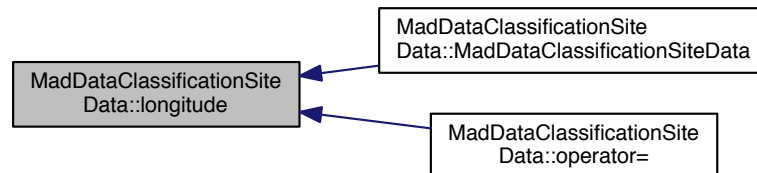
#### 6.7.3.6 MadSubCategory MadDataClassificationSiteData::longitude ( ) const

Definition at line 61 of file `maddataclassificationsitedata.cpp`.

```

62 {
63     return mLongitude;
64 }
  
```

Here is the caller graph for this function:



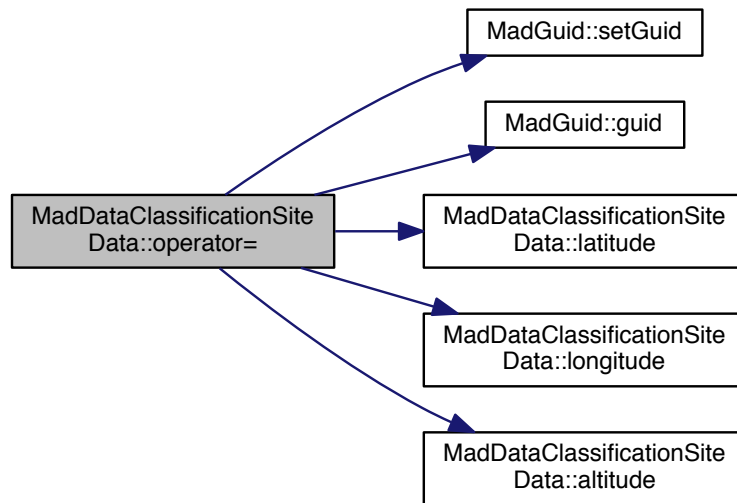
#### 6.7.3.7 MadDataClassificationSiteData & MadDataClassificationSiteData::operator= ( const MadDataClassificationSiteData & theData )

Definition at line 46 of file `maddataclassificationsitedata.cpp`.

```

47 {
48     // gracefully handles self assignment
49     if (this == &theData) return *this;
50     setGuid(theData.guid());
51     mLatitude=theData.latitude();
52     mLongitude=theData.longitude();
53     mAltitude=theData.altitude();
54     return *this;
55 }
  
```

Here is the call graph for this function:



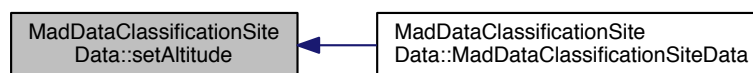
#### 6.7.3.8 void MadDataClassificationSiteData::setAltitude ( MadSubCategory *theData* )

Definition at line 81 of file `maddataclassificationssitedata.cpp`.

```

82 {
83     mAltitude = theData;
84 }
  
```

Here is the caller graph for this function:



#### 6.7.3.9 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid](#).

##### Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file `madguid.cpp`.

```

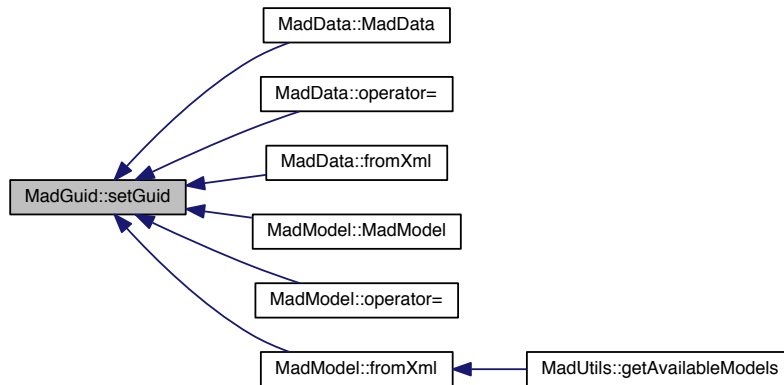
50 {
  
```

```

51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:



#### 6.7.3.10 void MadDataClassificationSiteData::setLatitude ( MadSubCategory *theData* )

Definition at line 71 of file `maddataclassificationsitedata.cpp`.

```

72 {
73     mLatitude = theData;
74 }

```

Here is the caller graph for this function:



#### 6.7.3.11 void MadDataClassificationSiteData::setLongitude ( MadSubCategory *theData* )

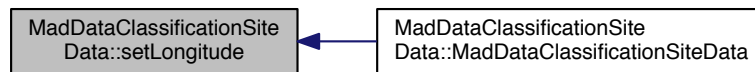
Definition at line 76 of file `maddataclassificationsitedata.cpp`.

```

77 {
78     mLongitude = theData;
79 }

```

Here is the caller graph for this function:



#### 6.7.3.12 QString MadDataClassificationSiteData::toHtml ( )

Return a html text representation of this layer

Definition at line 136 of file maddataclassificationsitedata.cpp.

```

137 {
138     QString myString;
139     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
140     //myString+="<p>GUID: " + guid() + "</p>";
141     myString+="<table>";
142     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
143
144     //
145     // the following shows example of how to do a couple of things
146     //
147
148     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
149     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
150     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
151     //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
152     //    QString::number(mCropFodderProduction) + "</td></tr>";
153     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
154     //    "</td></tr>";
155     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
156     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
157     myString+="</table>";
158     return myString;
159 }
```

#### 6.7.3.13 QString MadDataClassificationSiteData::toText ( )

Return a plain text representation of this layer

Definition at line 127 of file maddataclassificationsitedata.cpp.

```

128 {
129     QString myString;
130     myString+=QString("guid=> " + guid() + "\n");
131     //myString+=QString("name=> " + MadUtils::xmlEncode(mName) + "\n");
132     //myString+=QString("description=> " + MadUtils::xmlEncode(mDescription) + "\n");
133     return myString;
134 }
```

Here is the call graph for this function:



#### 6.7.3.14 QString MadDataClassificationSiteData::toXml( ) [virtual]

Return an xml representation of this layer

##### Note

this class inherits the serialisable interface so it MUST implement this

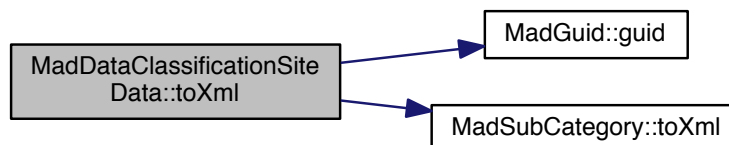
Implements [MadSerialisable](#).

Definition at line 105 of file `maddataclassificationsitedata.cpp`.

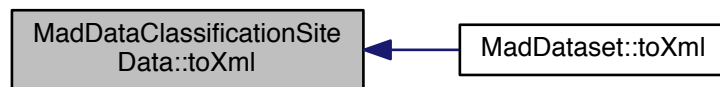
```

106 {
107     QString myString;
108     myString+=QString(" <site guid=\"" + guid() + "\">\n");
109
110     myString+=QString("         <latitude>\n");
111     myString+=mLatitude.toXml();
112     myString+=QString("         </latitude>\n");
113
114     myString+=QString("         <longitude>\n");
115     myString+=mLongitude.toXml();
116     myString+=QString("         </longitude>\n");
117
118     myString+=QString("         <altitude>\n");
119     myString+=mAltitude.toXml();
120     myString+=QString("         </altitude>\n");
121
122     myString+=QString(" </site>\n");
123     return myString;
124 }
125 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.7.3.15 `bool MadSerialisable::toXmlFile ( const QString theFileName )` [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:



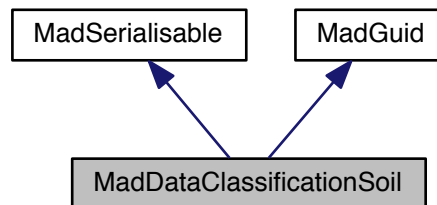
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoildata.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoildata.cpp](#)

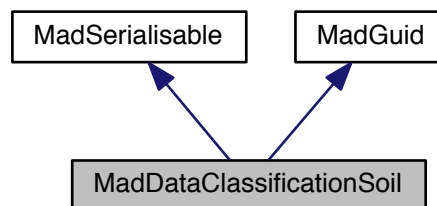
## 6.8 MadDataClassificationSoil Class Reference

```
#include <maddataclassificationsoil.h>
```

Inheritance diagram for MadDataClassificationSoil:



Collaboration diagram for MadDataClassificationSoil:



### Public Member Functions

- [MadDataClassificationSoil \(\)](#)
- [MadDataClassificationSoil \(const \[MadDataClassificationSoil\]\(#\) &theData\)](#)
- [MadDataClassificationSoil & operator= \(const \[MadDataClassificationSoil\]\(#\) &theData\)](#)
- [MadSubCategory carbonOrganic \(\) const](#)
- [MadSubCategory nitrogenOrganic \(\) const](#)
- [MadSubCategory texture \(\) const](#)
- [MadSubCategory bulkDensity \(\) const](#)
- [MadSubCategory fieldCapacityMeas \(\) const](#)

- [MadSubCategory wiltingPointMeas](#) () const
- [MadSubCategory pfCurve](#) () const
- [MadSubCategory hydrCondCurve](#) () const
- [MadSubCategory pH](#) () const
- [QString toXml](#) ()
- [QString toText](#) ()
- [QString toHtml](#) ()
- [bool fromXml](#) (const [QString](#) theXml)
- [void setCarbonOrganic](#) ([MadSubCategory](#) theData)
- [void setNitrogenOrganic](#) ([MadSubCategory](#) theData)
- [void setTexture](#) ([MadSubCategory](#) theData)
- [void setBulkDensity](#) ([MadSubCategory](#) theData)
- [void setFieldCapacityMeas](#) ([MadSubCategory](#) theData)
- [void setWiltingPointMeas](#) ([MadSubCategory](#) theData)
- [void setPfCurve](#) ([MadSubCategory](#) theData)
- [void setHydrCondCurve](#) ([MadSubCategory](#) theData)
- [void setPh](#) ([MadSubCategory](#) theData)
- [virtual bool toXmlFile](#) (const [QString](#) theFileName)  
*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*
- [virtual bool fromXmlFile](#) (const [QString](#) theFileName)  
*fromXmlFile Read this object from xml in a file*
- [QString guid](#) () const  
[MadGuid::guid.](#)
- [void setGuid](#) ([QString](#) theGuid="")  
[MadGuid::setGuid.](#)

### 6.8.1 Detailed Description

Definition at line 34 of file maddataclassificationsoil.h.

### 6.8.2 Constructor & Destructor Documentation

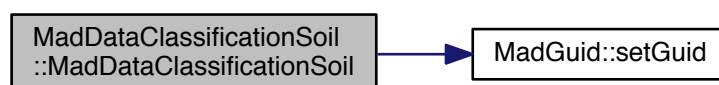
#### 6.8.2.1 MadDataClassificationSoil::MadDataClassificationSoil ( )

Definition at line 33 of file maddataclassificationsoil.cpp.

```

33                                     : MadSerialisable() ,
    MadGuid()
34 {
35     setGuid();
36 }
```

Here is the call graph for this function:



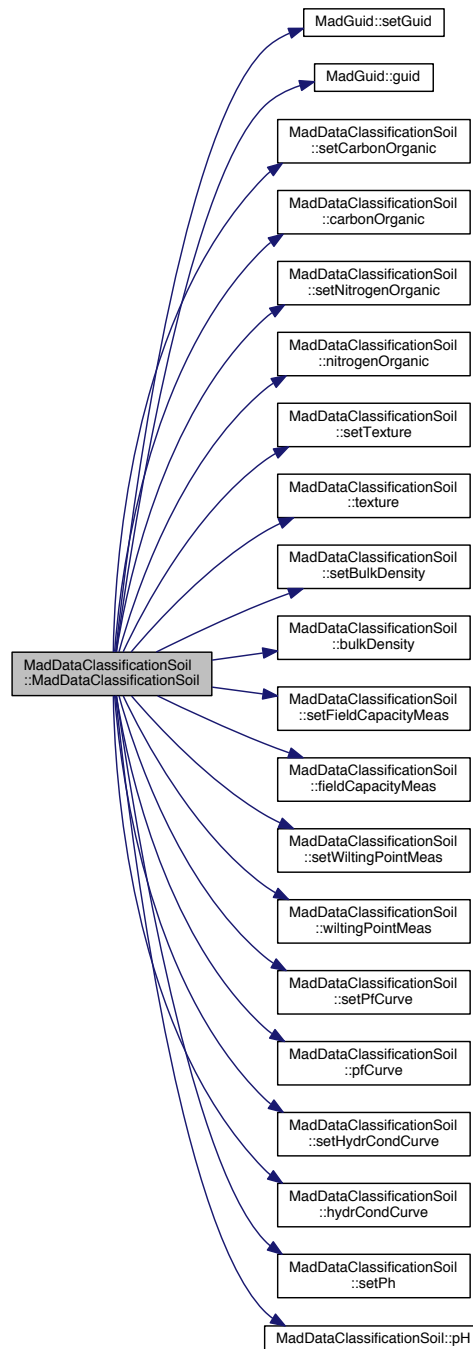


### 6.8.2.2 MadDataClassificationSoil::MadDataClassificationSoil ( const MadDataClassificationSoil & *theData* )

Definition at line 38 of file maddataclassificationsoil.cpp.

```
39 {  
40     setGuid(theData.guid());  
41     setCarbonOrganic(theData.carbonOrganic());  
42     setNitrogenOrganic(theData.nitrogenOrganic());  
43     setTexture(theData.texture());  
44     setBulkDensity(theData.bulkDensity());  
45     setFieldCapacityMeas(theData.fieldCapacityMeas());  
46     setWiltingPointMeas(theData.wiltingPointMeas());  
47     setPfCurve(theData.pfCurve());  
48     setHydrCondCurve(theData.hydrCondCurve());  
49     setPh(theData.pH());  
50 }
```

Here is the call graph for this function:



### 6.8.3 Member Function Documentation

#### 6.8.3.1 MadSubCategory MadDataClassificationSoil::bulkDensity ( ) const

Definition at line 82 of file maddataclassificationsoil.cpp.

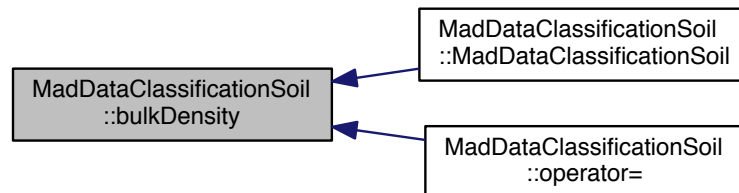
```

83 {
84     return mBulkDensity;

```

```
85 }
```

Here is the caller graph for this function:

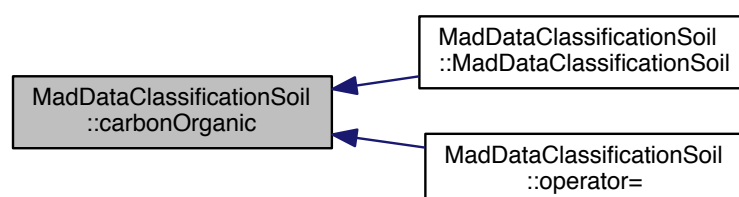


### 6.8.3.2 MadSubCategory MadDataClassificationSoil::carbonOrganic ( ) const

Definition at line 70 of file `maddataclassificationsoil.cpp`.

```
71 {  
72     return mCarbonOrganic;  
73 }
```

Here is the caller graph for this function:

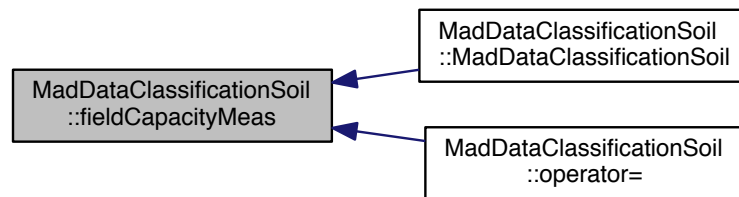


### 6.8.3.3 MadSubCategory MadDataClassificationSoil::fieldCapacityMeas ( ) const

Definition at line 86 of file `maddataclassificationsoil.cpp`.

```
87 {  
88     return mFieldCapacityMeas;  
89 }
```

Here is the caller graph for this function:



#### 6.8.3.4 `bool MadDataClassificationSoil::fromXml ( const QString theXml ) [virtual]`

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

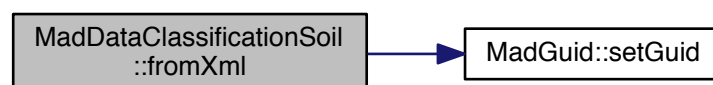
Implements [MadSerialisable](#).

Definition at line 153 of file `maddataclassificationsoil.cpp`.

```

154 {
155     QDomDocument myDocument("mydocument");
156     myDocument.setContent(theXml);
157     QDomElement myTopElement = myDocument.firstChildElement("soil");
158     if (myTopElement.isNull())
159     {
160         //TODO - just make this a warning
161         qDebug("the top element couldn't be found!");
162         setGuid(myTopElement.attribute("guid"));
163         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
164         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
165         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
166         return true;
167     }
168     else
169         return false;
170 }
  
```

Here is the call graph for this function:



**6.8.3.5** `bool MadSerialisable::fromXmlFile ( const QString theFileName )` `[virtual],[inherited]`

fromXmlFile Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



**6.8.3.6** `QString MadGuid::guid ( ) const` `[inherited]`

[MadGuid::guid.](#)

Destructor Retrieve the GUID

#### Returns

Definition at line 40 of file madguid.cpp.

```

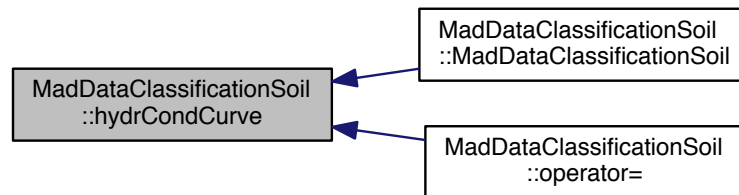
41 {
42     return mGuid;
43 }
```

### 6.8.3.7 MadSubCategory MadDataClassificationSoil::hydrCondCurve ( ) const

Definition at line 98 of file maddataclassificationsoil.cpp.

```
99 {
100     return mHydrCondCurve;
101 }
```

Here is the caller graph for this function:

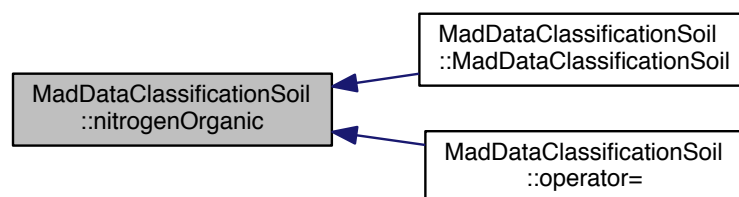


### 6.8.3.8 MadSubCategory MadDataClassificationSoil::nitrogenOrganic ( ) const

Definition at line 74 of file maddataclassificationsoil.cpp.

```
75 {
76     return mNitrogenOrganic;
77 }
```

Here is the caller graph for this function:



### 6.8.3.9 MadDataClassificationSoil & MadDataClassificationSoil::operator= ( const MadDataClassificationSoil & theData )

Definition at line 52 of file maddataclassificationsoil.cpp.

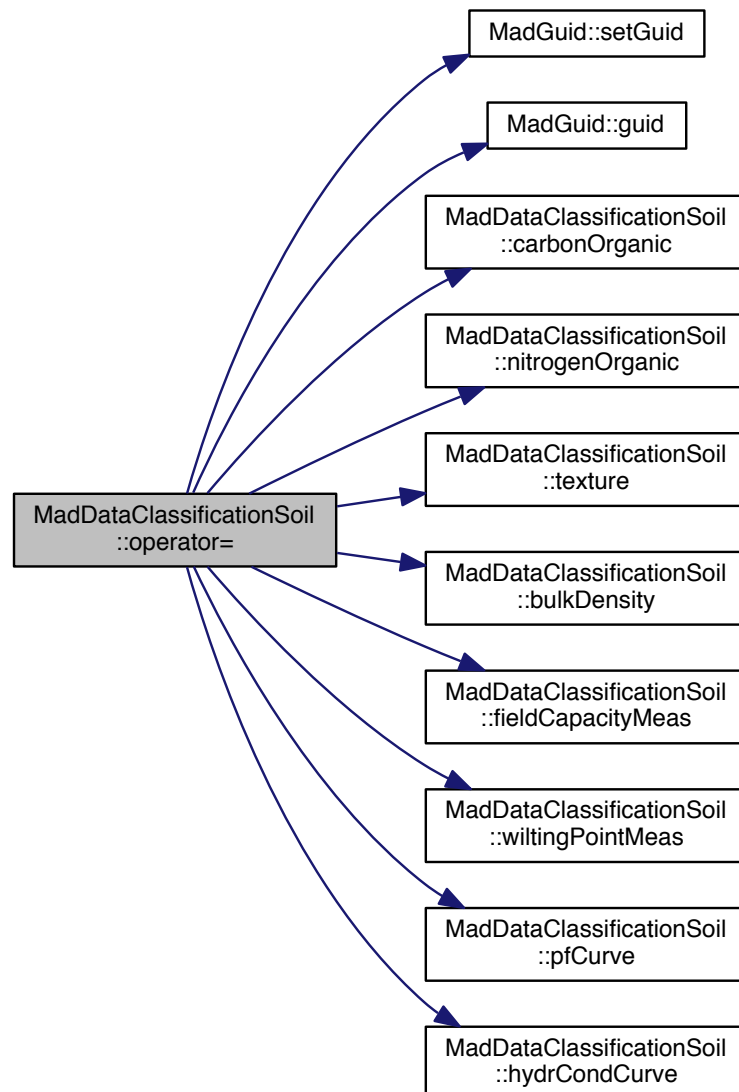
```
53 {
54     // gracefully handles self assignment
55     if (this == &theData) return *this;
56     setGuid(theData.guid());
57 }
```

```

57  mCarbonOrganic=theData.carbonOrganic();
58  mNitrogenOrganic=theData.nitrogenOrganic();
59  mTexture=theData.texture();
60  mBulkDensity=theData.bulkDensity();
61  mFieldCapacityMeas=theData.fieldCapacityMeas();
62  mWiltingPointMeas=theData.wiltingPointMeas();
63  mPfCurve=theData.pfCurve();
64  mHydrCondCurve=theData.hydrCondCurve();
65  mPH=theData.mPH;
66  return *this;
67 }

```

Here is the call graph for this function:



#### 6.8.3.10 MadSubCategory MadDataClassificationSoil::pfCurve ( ) const

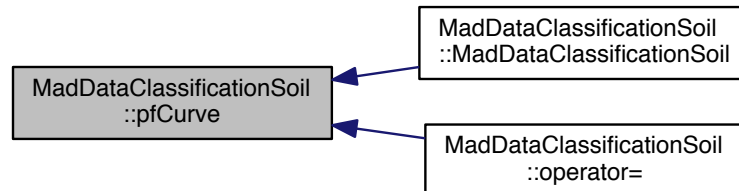
Definition at line 94 of file `maddataclassificationsoil.cpp`.

```

95 {
96     return mPfCurve;
97 }

```

Here is the caller graph for this function:



#### 6.8.3.11 MadSubCategory MadDataClassificationSoil::pH ( ) const

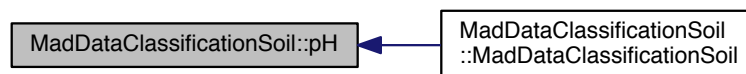
Definition at line 102 of file `maddataclassificationsoil.cpp`.

```

103 {
104     return mPH;
105 }

```

Here is the caller graph for this function:



#### 6.8.3.12 void MadDataClassificationSoil::setBulkDensity ( MadSubCategory theData )

Definition at line 122 of file `maddataclassificationsoil.cpp`.

```

123 {
124     mBulkDensity = theData;
125 }

```



Here is the caller graph for this function:



#### 6.8.3.13 void MadDataClassificationSoil::setCarbonOrganic ( MadSubCategory *theData* )

Definition at line 107 of file maddataclassificationsoil.cpp.

```
108 {  
109     mCarbonOrganic = theData;  
110 }
```

Here is the caller graph for this function:



#### 6.8.3.14 void MadDataClassificationSoil::setFieldCapacityMeas ( MadSubCategory *theData* )

Definition at line 127 of file maddataclassificationsoil.cpp.

```
128 {  
129     mFieldCapacityMeas = theData;  
130 }
```

Here is the caller graph for this function:



### 6.8.3.15 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

#### Parameters

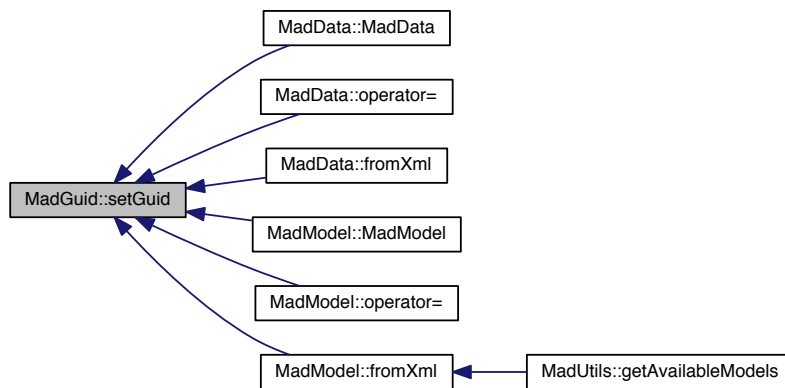
<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



### 6.8.3.16 void MadDataClassificationSoil::setHydrCondCurve ( MadSubCategory *theData* )

Definition at line 142 of file maddataclassificationsoil.cpp.

```

143 {
144     mHydrCondCurve = theData;
145 }
```

Here is the caller graph for this function:



**6.8.3.17 void MadDataClassificationSoil::setNitrogenOrganic ( MadSubCategory *theData* )**

Definition at line 112 of file maddataclassificationsoil.cpp.

```
113 {  
114     mNitrogenOrganic = theData;  
115 }
```

Here is the caller graph for this function:

**6.8.3.18 void MadDataClassificationSoil::setPfCurve ( MadSubCategory *theData* )**

Definition at line 137 of file maddataclassificationsoil.cpp.

```
138 {  
139     mPfCurve = theData;  
140 }
```

Here is the caller graph for this function:

**6.8.3.19 void MadDataClassificationSoil::setPh ( MadSubCategory *theData* )**

Definition at line 147 of file maddataclassificationsoil.cpp.

```
148 {  
149     mPH = theData;  
150 }
```

Here is the caller graph for this function:



#### 6.8.3.20 void MadDataClassificationSoil::setTexture ( MadSubCategory *theData* )

Definition at line 117 of file maddataclassificationsoil.cpp.

```
118 {  
119     mTexture = theData;  
120 }
```

Here is the caller graph for this function:



#### 6.8.3.21 void MadDataClassificationSoil::setWiltingPointMeas ( MadSubCategory *theData* )

Definition at line 132 of file maddataclassificationsoil.cpp.

```
133 {  
134     mWiltingPointMeas = theData;  
135 }
```

Here is the caller graph for this function:

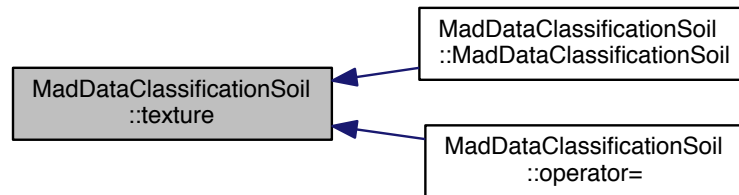


### 6.8.3.22 MadSubCategory MadDataClassificationSoil::texture ( ) const

Definition at line 78 of file maddataclassificationsoil.cpp.

```
79 {
80     return mTexture;
81 }
```

Here is the caller graph for this function:



### 6.8.3.23 QString MadDataClassificationSoil::toHtml ( )

Return a html text representation of this layer

Definition at line 227 of file maddataclassificationsoil.cpp.

```
228 {
229     QString myString;
230     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
231     //myString+="<p>GUID: " + guid() + "</p>";
232     myString+="<table>";
233     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
234
235     //
236     // the following shows example of how to do a couple of things
237     //
238
239     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
240     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
241     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
242     //myString+="<tr><td><b>Fodder (kg/" + myUnits + ")</b></td><td>" +
243     //    QString::number(mCropFodderProduction) + "</td></tr>";
244     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
245     //    "</td></tr>";
246     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
247     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
248     myString+="</table>";
249     return myString;
250 }
```

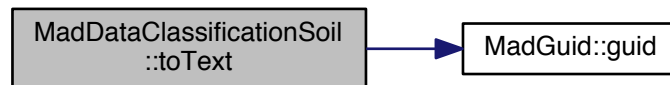
### 6.8.3.24 QString MadDataClassificationSoil::toText ( )

Return a plain text representation of this layer

Definition at line 218 of file maddataclassificationsoil.cpp.

```
219 {
220     QString myString;
221     myString+=QString("guid=>" + guid() + "\n");
222     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
223     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
224     return myString;
225 }
```

Here is the call graph for this function:



#### 6.8.3.25 QString MadDataClassificationSoil::toXml ( ) [virtual]

Return an xml representation of this layer

##### Note

this class inherits the serialisable interface so it MUST implement this

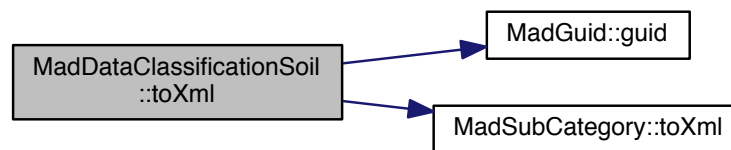
Implements [MadSerialisable](#).

Definition at line 172 of file maddataclassificationsoil.cpp.

```

173 {
174     QString myString;
175     myString+=QString("  <soil guid=\"" + guid() + "\">\n");
176
177     myString+=QString("    <corg>\n");
178     myString+=mCarbonOrganic.toXml();
179     myString+=QString("    </corg>\n");
180
181     myString+=QString("    <norg>\n");
182     myString+=mNitrogenOrganic.toXml();
183     myString+=QString("    </norg>\n");
184
185     myString+=QString("    <texture>\n");
186     myString+=mTexture.toXml();
187     myString+=QString("    </texture>\n");
188
189     myString+=QString("    <bulkdensity>\n");
190     myString+=mBulkDensity.toXml();
191     myString+=QString("    </bulkdensity>\n");
192
193     myString+=QString("    <fieldcapacity>\n");
194     myString+=mFieldCapacityMeas.toXml();
195     myString+=QString("    </fieldcapacity>\n");
196
197     myString+=QString("    <wiltingpoint>\n");
198     myString+=mWiltingPointMeas.toXml();
199     myString+=QString("    </wiltingpoint>\n");
200
201     myString+=QString("    <pfcurve>\n");
202     myString+=mPfCurve.toXml();
203     myString+=QString("    </pfcurve>\n");
204
205     myString+=QString("    <hydrcondcurve>\n");
206     myString+=mHydrCondCurve.toXml();
207     myString+=QString("    </hydrcondcurve>\n");
208
209     myString+=QString("    <ph>\n");
210     myString+=mPH.toXml();
211     myString+=QString("    </ph>\n");
212
213     myString+=QString("  </soil>\n");
214     return myString;
215 }
216
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.8.3.26 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
  
```

```

69  {
70      //@TODO Error handler!
71      myResult=false;
72  }
73  return myResult ;
74  }

```

Here is the call graph for this function:



### 6.8.3.27 MadSubCategory MadDataClassificationSoil::wiltingPointMeas ( ) const

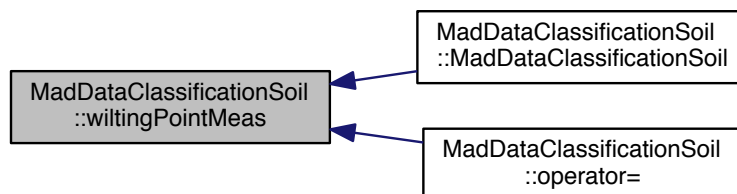
Definition at line 90 of file `maddataclassificationsoil.cpp`.

```

91 {
92     return mWiltingPointMeas;
93 }

```

Here is the caller graph for this function:



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

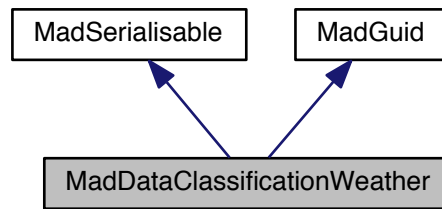
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationsoil.cpp](#)

## 6.9 MadDataClassificationWeather Class Reference

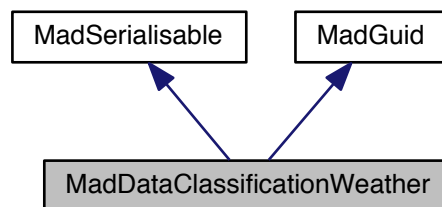
```
#include <maddataclassificationweather.h>
```



Inheritance diagram for MadDataClassificationWeather:



Collaboration diagram for MadDataClassificationWeather:



## Public Member Functions

- [MadDataClassificationWeather](#) ()
- [MadDataClassificationWeather](#) (const [MadDataClassificationWeather](#) &theData)
- [MadDataClassificationWeather](#) & operator= (const [MadDataClassificationWeather](#) &theData)
- bool [minData](#) () const
- [MadSubCategory precipitation](#) () const
- [MadSubCategory tAve](#) () const
- [MadSubCategory tMin](#) () const
- [MadSubCategory tMax](#) () const
- [MadSubCategory relativeHumidity](#) () const
- [MadSubCategory windSpeed](#) () const
- [MadSubCategory globalRadiation](#) () const
- [MadSubCategory sunshineHours](#) () const
- [MadSubCategory leafWetness](#) () const
- [MadSubCategory soilTemp](#) () const
- QString [toXml](#) ()
- QString [toText](#) ()
- QString [toHtml](#) ()
- bool [fromXml](#) (const QString theXml)
- void [setMinData](#) (bool theBool)
- void [setPrecipitation](#) ([MadSubCategory](#) theData)

- void [setTAve](#) ([MadSubCategory](#) theData)
- void [setTMin](#) ([MadSubCategory](#) theData)
- void [setTMax](#) ([MadSubCategory](#) theData)
- void [setRelativeHumidity](#) ([MadSubCategory](#) theData)
- void [setWindSpeed](#) ([MadSubCategory](#) theData)
- void [setGlobalRadiation](#) ([MadSubCategory](#) theData)
- void [setSunshineHours](#) ([MadSubCategory](#) theData)
- void [setLeafWetness](#) ([MadSubCategory](#) theData)
- void [setSoilTemp](#) ([MadSubCategory](#) theData)
- virtual bool [toXmlFile](#) (const QString theFileName)

*toXmlFile* writes *object* to *xml* and return result (virtual *qstring*) We provide a basic default implementation where given a file name, we will write the serialised *xml* to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

- virtual bool [fromXmlFile](#) (const QString theFileName)  
*fromXmlFile* Read this object from *xml* in a file
- QString [guid](#) () const  
[MadGuid::guid](#).
- void [setGuid](#) (QString theGuid="")  
[MadGuid::setGuid](#).

### 6.9.1 Detailed Description

Definition at line 35 of file `maddataclassificationweather.h`.

### 6.9.2 Constructor & Destructor Documentation

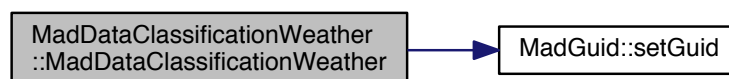
#### 6.9.2.1 MadDataClassificationWeather::MadDataClassificationWeather ( )

Definition at line 33 of file `maddataclassificationweather.cpp`.

```

33                                     : MadSerialisable(),
34     MadGuid()
35 {
36     setGuid();
37 }
```

Here is the call graph for this function:



#### 6.9.2.2 MadDataClassificationWeather::MadDataClassificationWeather ( const MadDataClassificationWeather & theData )

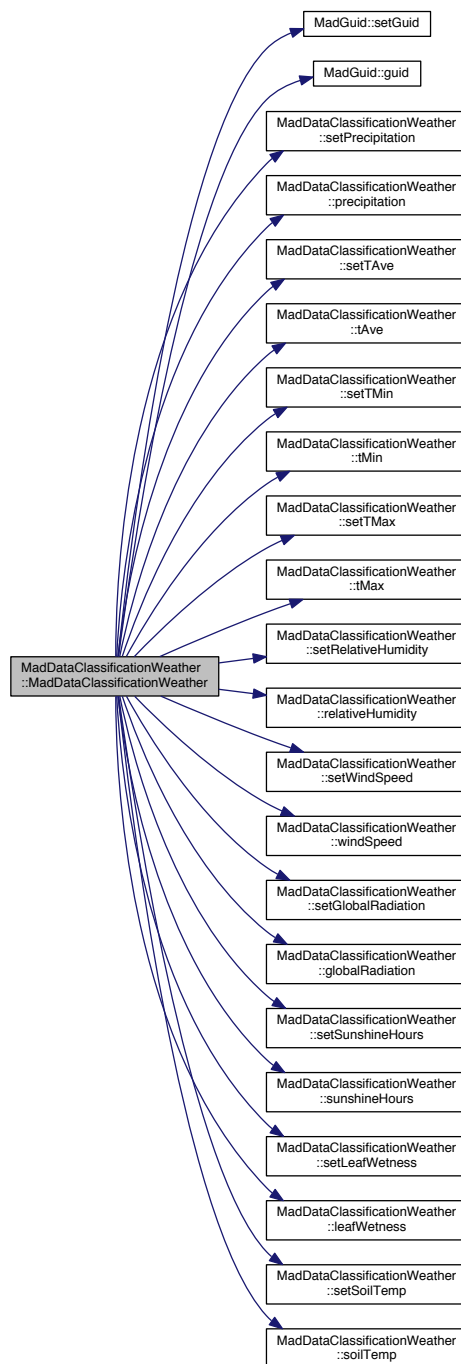
Definition at line 38 of file `maddataclassificationweather.cpp`.

```

39 {
40     setGuid(theData.guid());
41     setPrecipitation(theData.precipitation());
42     setTAve(theData.tAve());
43     setTMin(theData.tMin());
44     setTMax(theData.tMax());
45     setRelativeHumidity(theData.relativeHumidity());
46     setWindSpeed(theData.windSpeed());
47     setGlobalRadiation(theData.globalRadiation());
48     setSunshineHours(theData.sunshineHours());
49     setLeafWetness(theData.leafWetness());
50     setSoilTemp(theData.soilTemp());
51 }

```

Here is the call graph for this function:



### 6.9.3 Member Function Documentation

#### 6.9.3.1 bool MadDataClassificationWeather::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

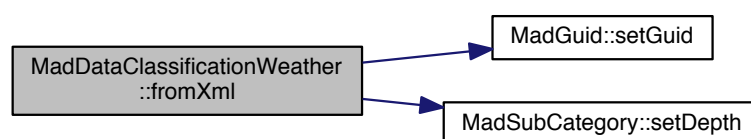
Definition at line 173 of file maddataclassificationweather.cpp.

```

174 {
175     QDomDocument myDocument("mydocument");
176     myDocument.setContent(theXml);
177     QDomElement myTopElement = myDocument.firstChildElement("weather");
178     if (myTopElement.isNull())
179     {
180         // TODO - just make this a warning
181         qDebug("the top element couldn't be found!");
182         setGuid(myTopElement.attribute("guid"));
183     }
184     //MadDataClassificationWeather myWeather;
185     //QString myPrecipitationXml =
186     QString(QDomDocumentFragment().firstChildElement("precipitation").text());
187     //myWeather.setPrecipitation(MadSubCategory::fromXml(myPrecipitationXml));
188     // the line below works and does the same as the line below it.
189     // (QString(myTopElement.firstChildElement("mindata").text())=="0" ? mMinData=false : mMinData=true;
190     mMinData = QString(myTopElement.firstChildElement("mindata").text()).toInt();
191     MadSubCategory myPrecipitationDetails;
192     myPrecipitationDetails.setDepth( QString(myTopElement.firstChildElement("precipitation").
193     nextSiblingElement("details").nextSiblingElement("depth").text()).toFloat());
194     //qDebug()
195     /* the following doesn't work
196     mPrecipitation = MadUtils::xmlDecode(myTopElement.firstChildElement("precipitation").text()).toInt();
197     mTAve = MadUtils::xmlDecode(myTopElement.firstChildElement("tave").text()).toInt();
198     mTMin = MadUtils::xmlDecode(myTopElement.firstChildElement("tmin").text()).toFloat();
199     mTMax = MadUtils::xmlDecode(myTopElement.firstChildElement("tmax").text()).toInt();
200     mRelativeHumidity =
201     MadUtils::xmlDecode(myTopElement.firstChildElement("relativehumidity").text()).toInt();
202     mWindSpeed = MadUtils::xmlDecode(myTopElement.firstChildElement("windspeed").text()).toInt();
203     mGlobalRadiation =
204     MadUtils::xmlDecode(myTopElement.firstChildElement("globalradiation").text()).toInt();
205     mSunshineHours = MadUtils::xmlDecode(myTopElement.firstChildElement("sunshinehours").text()).toInt();
206     mLeafWetness = MadUtils::xmlDecode(myTopElement.firstChildElement("leafwetness").text()).toInt();
207     mSoilTemp = MadUtils::xmlDecode(myTopElement.firstChildElement("soiltemp").text()).toInt();
208     */
209     return true;
210 }
211 else
212     return false;
213 }

```

Here is the call graph for this function:



### 6.9.3.2 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



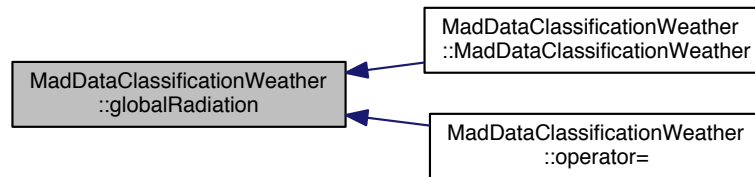
### 6.9.3.3 MadSubCategory MadDataClassificationWeather::globalRadiation ( ) const

Definition at line 101 of file maddataclassificationweather.cpp.

```

102 {
103     return mGlobalRadiation;
104 }
```

Here is the caller graph for this function:



#### 6.9.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

##### Returns

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

```

41 {
42     return mGuid;
43 }
```

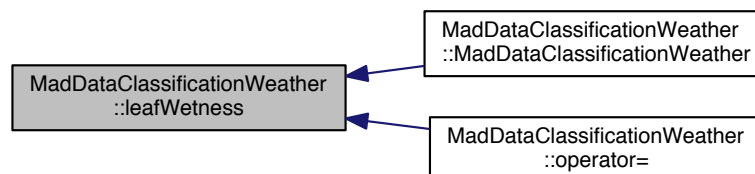
#### 6.9.3.5 MadSubCategory MadDataClassificationWeather::leafWetness ( ) const

Definition at line 109 of file `maddataclassificationweather.cpp`.

```

110 {
111     return mLeafWetness;
112 }
```

Here is the caller graph for this function:



### 6.9.3.6 bool MadDataClassificationWeather::minData ( ) const

Definition at line 73 of file maddataclassificationweather.cpp.

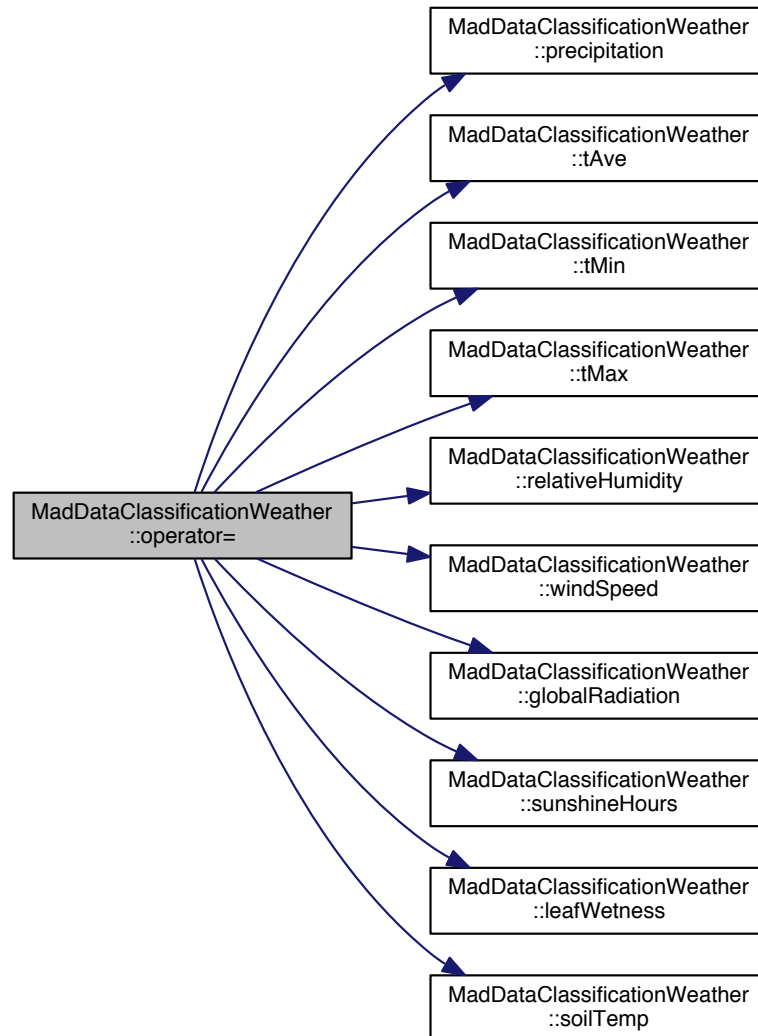
```
74 {  
75     return mMinData;  
76 }
```

### 6.9.3.7 MadDataClassificationWeather & MadDataClassificationWeather::operator= ( const MadDataClassificationWeather & *theData* )

Definition at line 53 of file maddataclassificationweather.cpp.

```
54 {  
55     // gracefully handles self assignment  
56     if (this == &theData) return *this;  
57     //setGuid(theData.guid());  
58     mPrecipitation=theData.precipitation();  
59     mTAve=theData.tAve();  
60     mTMin=theData.tMin();  
61     mTMax=theData.tMax();  
62     mRelativeHumidity=theData.relativeHumidity();  
63     mWindSpeed=theData.windSpeed();  
64     mGlobalRadiation=theData.globalRadiation();  
65     mSunshineHours=theData.sunshineHours();  
66     mLeafWetness=theData.leafWetness();  
67     mSoilTemp=theData.soilTemp();  
68  
69     return *this;  
70 }
```

Here is the call graph for this function:



#### 6.9.3.8 MadSubCategory MadDataClassificationWeather::precipitation ( ) const

Definition at line 77 of file `maddataclassificationweather.cpp`.

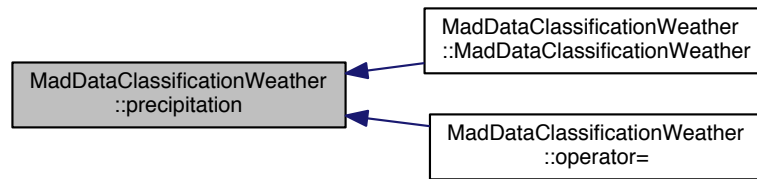
```

78 {
79     return mPrecipitation;
80 }

```



Here is the caller graph for this function:

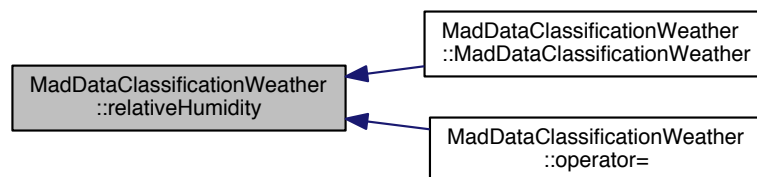


#### 6.9.3.9 MadSubCategory MadDataClassificationWeather::relativeHumidity ( ) const

Definition at line 93 of file `maddataclassificationweather.cpp`.

```
94 {  
95     return mRelativeHumidity;  
96 }
```

Here is the caller graph for this function:



#### 6.9.3.10 void MadDataClassificationWeather::setGlobalRadiation ( MadSubCategory theData )

Definition at line 153 of file `maddataclassificationweather.cpp`.

```
154 {  
155     mGlobalRadiation = theData;  
156 }
```

Here is the caller graph for this function:



### 6.9.3.11 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

#### Parameters

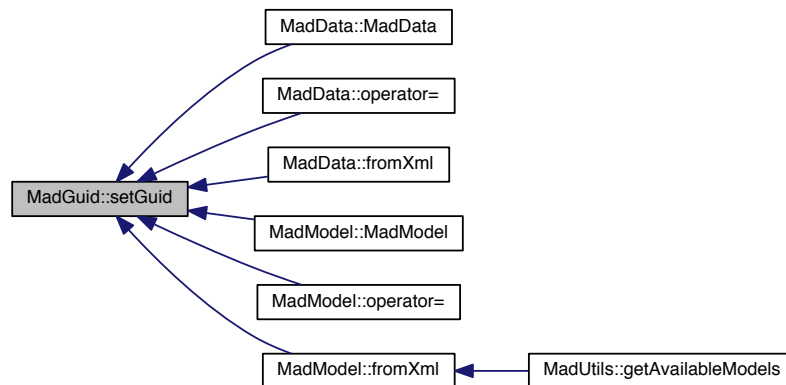
<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



### 6.9.3.12 void MadDataClassificationWeather::setLeafWetness ( MadSubCategory *theData* )

Definition at line 163 of file maddataclassificationweather.cpp.

```

164 {
165     mLeafWetness = theData;
166 }
```

Here is the caller graph for this function:



**6.9.3.13 void MadDataClassificationWeather::setMinData ( bool *theBool* )**

Definition at line 119 of file maddataclassificationweather.cpp.

```
120 {  
121     mMinData = theBool;  
122 }
```

**6.9.3.14 void MadDataClassificationWeather::setPrecipitation ( MadSubCategory *theData* )**

Definition at line 123 of file maddataclassificationweather.cpp.

```
124 {  
125     mPrecipitation = theData;  
126 }
```

Here is the caller graph for this function:

**6.9.3.15 void MadDataClassificationWeather::setRelativeHumidity ( MadSubCategory *theData* )**

Definition at line 143 of file maddataclassificationweather.cpp.

```
144 {  
145     mRelativeHumidity = theData;  
146 }
```

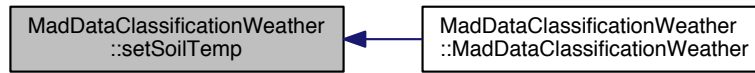
Here is the caller graph for this function:

**6.9.3.16 void MadDataClassificationWeather::setSoilTemp ( MadSubCategory *theData* )**

Definition at line 168 of file maddataclassificationweather.cpp.

```
169 {  
170     mSoilTemp = theData;  
171 }
```

Here is the caller graph for this function:



#### 6.9.3.17 void MadDataClassificationWeather::setSunshineHours ( MadSubCategory *theData* )

Definition at line 158 of file maddataclassificationweather.cpp.

```
159 {  
160     mSunshineHours = theData;  
161 }
```

Here is the caller graph for this function:



#### 6.9.3.18 void MadDataClassificationWeather::setTAve ( MadSubCategory *theData* )

Definition at line 128 of file maddataclassificationweather.cpp.

```
129 {  
130     mTAve = theData;  
131 }
```

Here is the caller graph for this function:



#### 6.9.3.19 void MadDataClassificationWeather::setTMax ( MadSubCategory *theData* )

Definition at line 138 of file maddataclassificationweather.cpp.

```
139 {  
140     mTMax = theData;  
141 }
```

Here is the caller graph for this function:



#### 6.9.3.20 void MadDataClassificationWeather::setTMin ( MadSubCategory *theData* )

Definition at line 133 of file maddataclassificationweather.cpp.

```
134 {  
135     mTMin = theData;  
136 }
```

Here is the caller graph for this function:



#### 6.9.3.21 void MadDataClassificationWeather::setWindSpeed ( MadSubCategory *theData* )

Definition at line 148 of file maddataclassificationweather.cpp.

```
149 {  
150     mWindSpeed = theData;  
151 }
```

Here is the caller graph for this function:

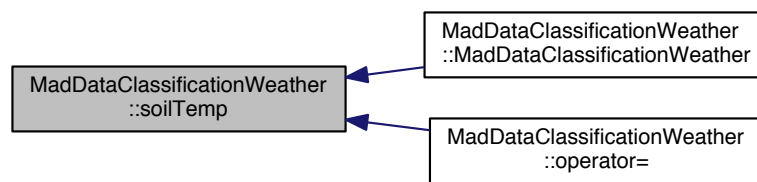


#### 6.9.3.22 MadSubCategory MadDataClassificationWeather::soilTemp ( ) const

Definition at line 113 of file maddataclassificationweather.cpp.

```
114 {  
115     return mSoilTemp;  
116 }
```

Here is the caller graph for this function:

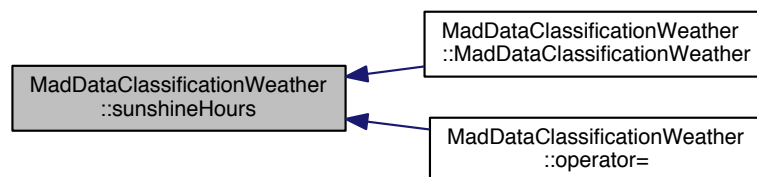


#### 6.9.3.23 MadSubCategory MadDataClassificationWeather::sunshineHours ( ) const

Definition at line 105 of file maddataclassificationweather.cpp.

```
106 {  
107     return mSunshineHours;  
108 }
```

Here is the caller graph for this function:

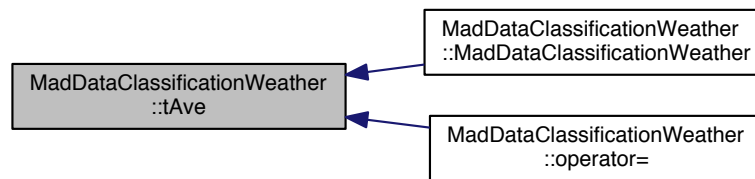


#### 6.9.3.24 MadSubCategory MadDataClassificationWeather::tAve ( ) const

Definition at line 81 of file maddataclassificationweather.cpp.

```
82 {  
83     return mTAve;  
84 }
```

Here is the caller graph for this function:

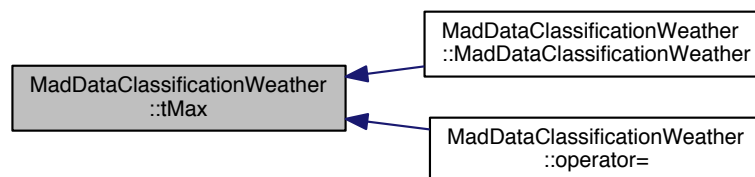


#### 6.9.3.25 MadSubCategory MadDataClassificationWeather::tMax ( ) const

Definition at line 89 of file `maddataclassificationweather.cpp`.

```
90 {  
91     return mTMax;  
92 }
```

Here is the caller graph for this function:

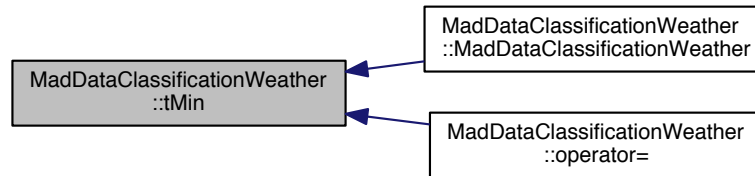


#### 6.9.3.26 MadSubCategory MadDataClassificationWeather::tMin ( ) const

Definition at line 85 of file `maddataclassificationweather.cpp`.

```
86 {  
87     return mTMin;  
88 }
```

Here is the caller graph for this function:



### 6.9.3.27 QString MadDataClassificationWeather::toHtml ( )

Return a html text representation of this layer

Definition at line 272 of file maddataclassificationweather.cpp.

```

273 {
274     QString myString;
275     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
276     //myString+="<p>GUID: " + guid() + "</p>";
277     myString+="<table>";
278     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
279
280     //
281     // the following shows example of how to do a couple of things
282     //
283
284     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
285     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
286     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
287     //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
288         //QString::number(mCropFodderProduction) + "</td></tr>";
289     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
290         //QString::number(mCropFodderEnergyType) + "</td></tr>";
291     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
292     myString+="</table>";
293     return myString;
294 }
  
```

### 6.9.3.28 QString MadDataClassificationWeather::toText ( )

Return a plain text representation of this layer

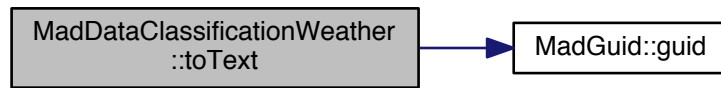
Definition at line 263 of file maddataclassificationweather.cpp.

```

264 {
265     QString myString;
266     myString+=QString("guid=>" + guid() + "\n");
267     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
268     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
269     return myString;
270 }
  
```



Here is the call graph for this function:



### 6.9.3.29 QString MadDataClassificationWeather::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

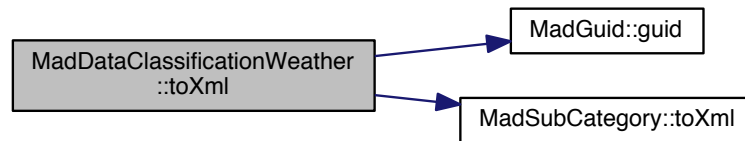
Implements [MadSerialisable](#).

Definition at line 213 of file maddataclassificationweather.cpp.

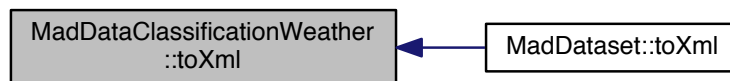
```

214 {
215     QString myString;
216     myString+=QString(" <weather guid=\"" + guid() + "\">\n");
217
218     myString+=QString("     <precipitation>\n");
219     myString+=mPrecipitation.toXml();
220     myString+=QString("     </precipitation>\n");
221
222     myString+=QString("     <tave>\n");
223     myString+=mTAve.toXml();
224     myString+=QString("     </tave>\n");
225
226     myString+=QString("     <tmin>\n");
227     myString+=mTMin.toXml();
228     myString+=QString("     </tmin>\n");
229
230     myString+=QString("     <tmax>\n");
231     myString+=mTMax.toXml();
232     myString+=QString("     </tmax>\n");
233
234     myString+=QString("     <relativehumidity>\n");
235     myString+=mRelativeHumidity.toXml();
236     myString+=QString("     </relativehumidity>\n");
237
238     myString+=QString("     <windspeed>\n");
239     myString+=mWindSpeed.toXml();
240     myString+=QString("     </windspeed>\n");
241
242     myString+=QString("     <globalradiation>\n");
243     myString+=mGlobalRadiation.toXml();
244     myString+=QString("     </globalradiation>\n");
245
246     myString+=QString("     <sunshinehours>\n");
247     myString+=mSunshineHours.toXml();
248     myString+=QString("     </sunshinehours>\n");
249
250     myString+=QString("     <leafwetness>\n");
251     myString+=mLeafWetness.toXml();
252     myString+=QString("     </leafwetness>\n");
253
254     myString+=QString("     <soiltemp>\n");
255     myString+=mSoilTemp.toXml();
256     myString+=QString("     </soiltemp>\n");
257
258     myString+=QString(" </weather>\n");
259     return myString;
260 }
261 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.9.3.30 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //@TODO Error handler!
  
```

```

71     myResult=false;
72 }
73 return myResult ;
74 }

```

Here is the call graph for this function:



#### 6.9.3.31 MadSubCategory MadDataClassificationWeather::windSpeed ( ) const

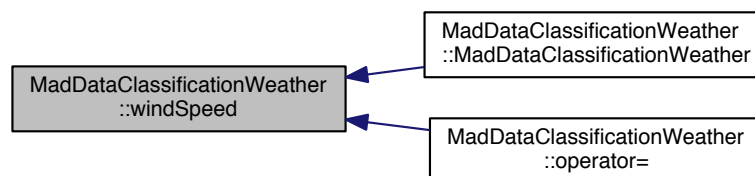
Definition at line 97 of file `maddataclassificationweather.cpp`.

```

98 {
99     return mWindSpeed;
100 }

```

Here is the caller graph for this function:



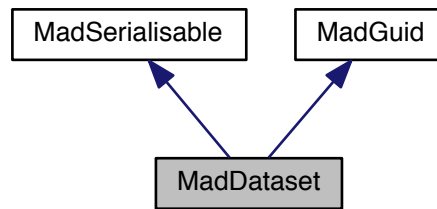
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.cpp](#)

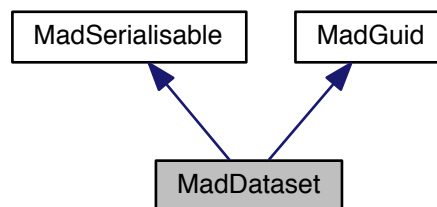
## 6.10 MadDataset Class Reference

```
#include <maddataset.h>
```

Inheritance diagram for MadDataset:



Collaboration diagram for MadDataset:



## Public Member Functions

- [MadDataset](#) ()
- [MadDataset](#) (const [MadDataset](#) &theData)
- [MadDataset](#) & operator= (const [MadDataset](#) &theDataset)
- [QString](#) [name](#) () const
- [QString](#) [description](#) () const
- [MadDataClassificationCultivation](#) [cultivation](#) () const
- [MadDataClassificationInitialValues](#) [initialValues](#) () const
- [MadDataClassificationPhenology](#) [phenology](#) () const
- [MadDataClassificationPrevCrop](#) [prevCrop](#) () const
- [MadDataClassificationSiteData](#) [siteData](#) () const
- [MadDataClassificationSoil](#) [soil](#) () const
- [MadDataClassificationWeather](#) [weather](#) () const
- [MadStateVars](#) [stateVars](#) () const
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- [bool](#) [fromXml](#) (const [QString](#) theXml)
- [void](#) [setName](#) ([QString](#) theName)
- [void](#) [setDescription](#) ([QString](#) theDescription)
- [void](#) [setCultivation](#) ([MadDataClassificationCultivation](#) theCultivationData)

- void [setInitialValues](#) ([MadDataClassificationInitialValues](#) theInitialValues)
- void [setPhenology](#) ([MadDataClassificationPhenology](#) thePhenologyData)
- void [setPrevCrop](#) ([MadDataClassificationPrevCrop](#) thePrevCropData)
- void [setSiteData](#) ([MadDataClassificationSiteData](#) theSiteData)
- void [setSoil](#) ([MadDataClassificationSoil](#) theSoilData)
- void [setWeather](#) ([MadDataClassificationWeather](#) theWeatherData)
- void [setStateVars](#) ([MadStateVars](#) theStateVarsData)
- virtual bool [toXmlFile](#) (const QString theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- virtual bool [fromXmlFile](#) (const QString theFileName)

*fromXmlFile Read this object from xml in a file*

- QString [guid](#) () const  
*MadGuid::guid.*
- void [setGuid](#) (QString theGuid="")  
*MadGuid::setGuid.*

### 6.10.1 Detailed Description

Definition at line 44 of file maddataset.h.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 MadDataset::MadDataset ( )

Definition at line 34 of file maddataset.cpp.

```

34                                     : MadSerialisable(), MadGuid()
35 {
36     setGuid();
37     mName="No Name Set";
38     mDescription="Not Set";
39     // we can put in other defaults here, such as
40     // mTheme="Valid for all themes"; <-- this doesn't exist though haha
41 }
```

Here is the call graph for this function:



#### 6.10.2.2 MadDataset::MadDataset ( const MadDataset & theData )

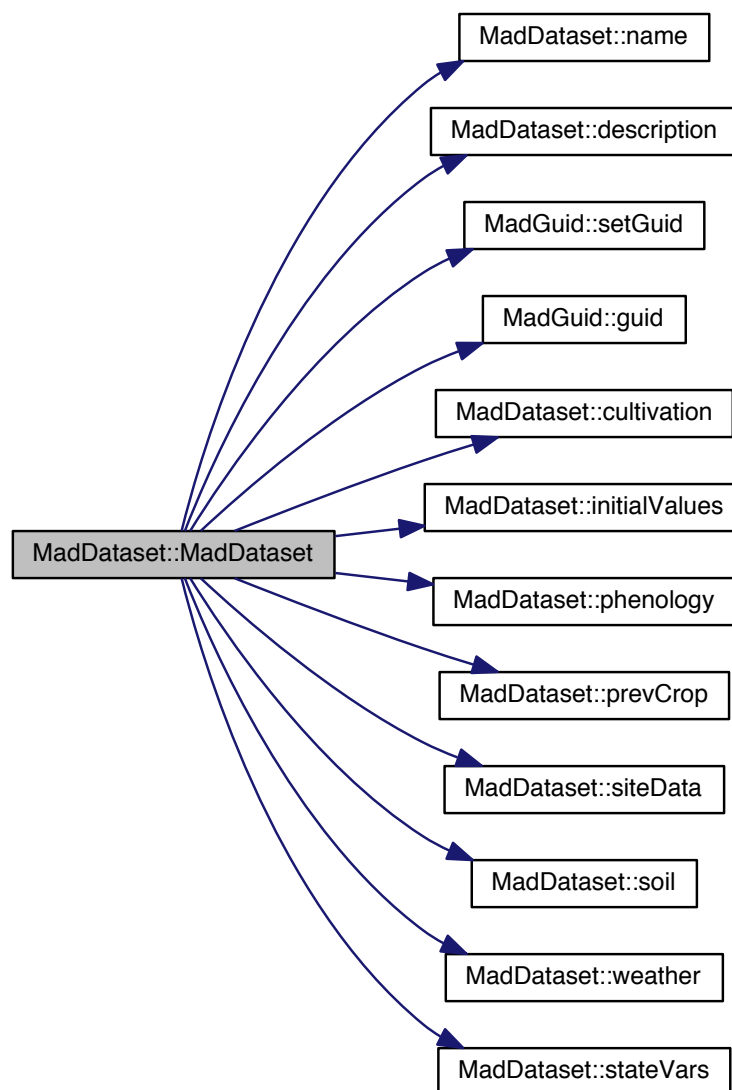
Definition at line 43 of file maddataset.cpp.

```

44 {
45     mName=theData.name();
46     mDescription=theData.description();
47     setGuid(theData.guid());
48     mCultivation=theData.cultivation();
49     mInitialValues=theData.initialValues();
50     mPhenology=theData.phenology();
51     mPrevCrop=theData.prevCrop();
52     mSiteData=theData.siteData();
53     mSoil=theData.soil();
54     mWeather=theData.weather();
55     mStateVars=theData.stateVars();
56 }

```

Here is the call graph for this function:



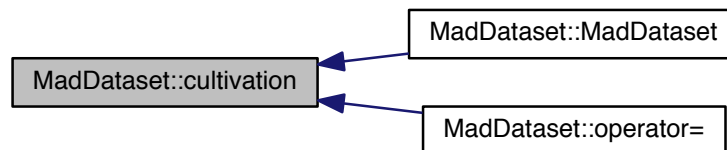
### 6.10.3 Member Function Documentation

### 6.10.3.1 MadDataClassificationCultivation MadDataset::cultivation ( ) const

Definition at line 86 of file maddataset.cpp.

```
87 {  
88     return mCultivation;  
89 }
```

Here is the caller graph for this function:

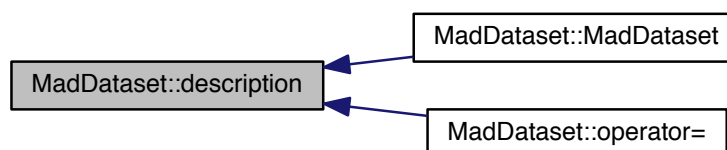


### 6.10.3.2 QString MadDataset::description ( ) const

Definition at line 81 of file maddataset.cpp.

```
82 {  
83     return mDescription;  
84 }
```

Here is the caller graph for this function:



### 6.10.3.3 bool MadDataset::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

**Note**

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 171 of file maddataset.cpp.

```

172 {
173     QDomDocument myDocument("mydocument");
174     myDocument.setContent(theXml);
175     QDomElement myTopElement = myDocument.firstChildElement("model");
176     if (myTopElement.isNull())
177     {
178         //TODO - just make this a warning
179         qDebug("the top element couldn't be found!");
180         setGuid(myTopElement.attribute("guid"));
181         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
182         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
183         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
184         return true;
185     }
186     else
187         return false;
188 }

```

Here is the call graph for this function:



#### 6.10.3.4 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

**See Also**

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

**Parameters**

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else

```

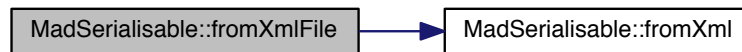


```

86  {
87      // @TODO Error handler!
88      myResult=false;
89  }
90  return myResult ;
91  }

```

Here is the call graph for this function:



#### 6.10.3.5 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

Returns

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }

```

#### 6.10.3.6 MadDataClassificationInitialValues MadDataset::initialValues ( ) const

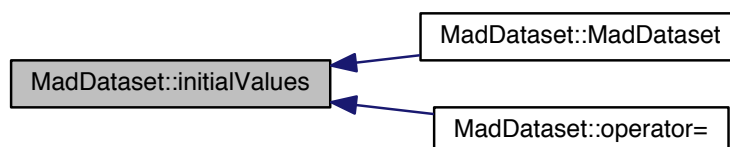
Definition at line 90 of file maddataset.cpp.

```

91 {
92     return mInitialValues;
93 }

```

Here is the caller graph for this function:

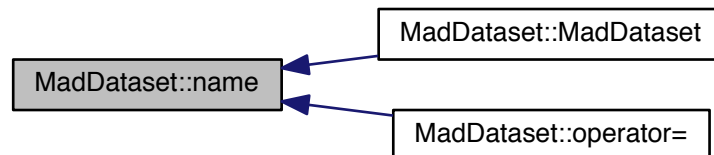


### 6.10.3.7 QString MadDataset::name ( ) const

Definition at line 77 of file maddataset.cpp.

```
78 {
79     return mName;
80 }
```

Here is the caller graph for this function:

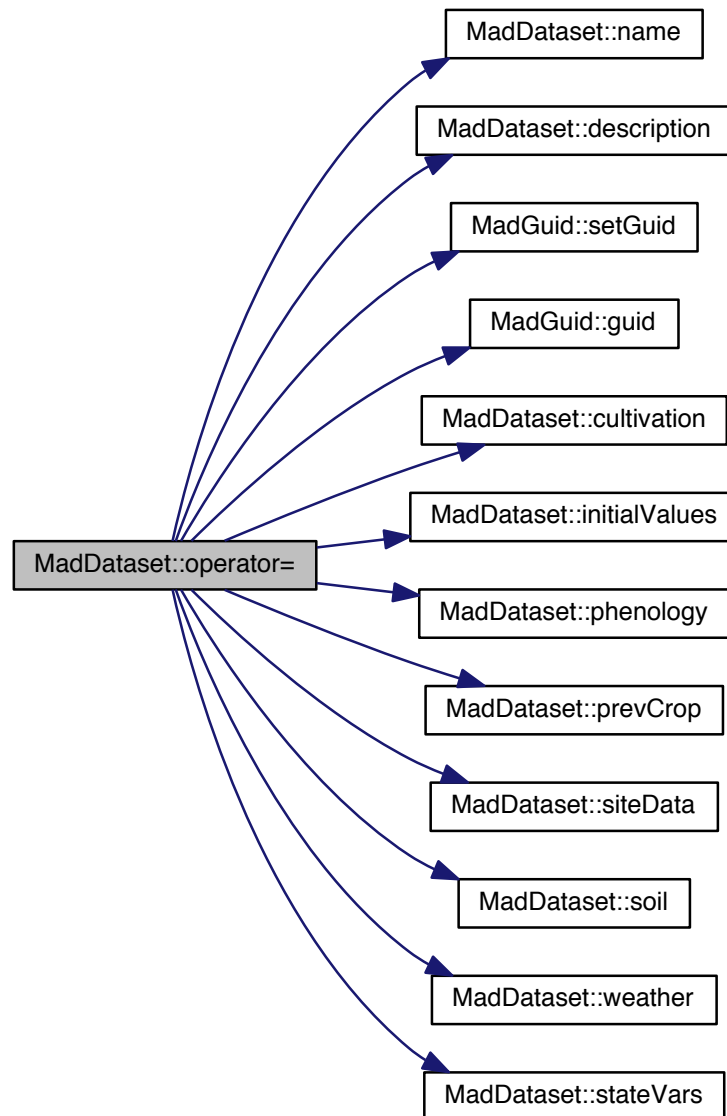


### 6.10.3.8 MadDataset & MadDataset::operator= ( const MadDataset & *theDataset* )

Definition at line 58 of file maddataset.cpp.

```
59 {
60     // gracefully handles self assignment
61     if (this == &theData) return *this;
62     mName=theData.name();
63     mDescription=theData.description();
64     setGuid(theData.guid());
65     mCultivation=theData.cultivation();
66     mInitialValues=theData.initialValues();
67     mPhenology=theData.phenology();
68     mPrevCrop=theData.prevCrop();
69     mSiteData=theData.siteData();
70     mSoil=theData.soil();
71     mWeather=theData.weather();
72     mStateVars=theData.stateVars();
73     return *this;
74 }
```

Here is the call graph for this function:

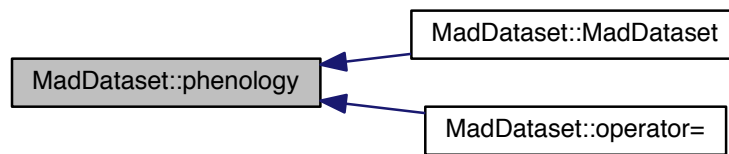


#### 6.10.3.9 MadDataClassificationPhenology MadDataset::phenology ( ) const

Definition at line 94 of file `maddataset.cpp`.

```
95 {  
96     return mPhenology;  
97 }
```

Here is the caller graph for this function:



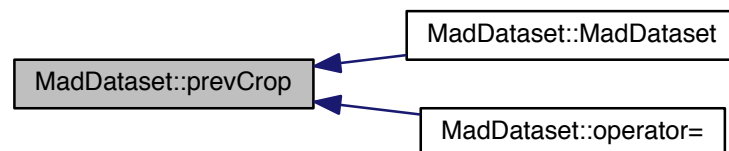
#### 6.10.3.10 MadDataClassificationPrevCrop MadDataset::prevCrop ( ) const

Definition at line 98 of file maddataset.cpp.

```

99 {
100     return mPrevCrop;
101 }
  
```

Here is the caller graph for this function:



#### 6.10.3.11 void MadDataset::setCultivation ( MadDataClassificationCultivation theCultivationData )

Definition at line 131 of file maddataset.cpp.

```

132 {
133     mCultivation = theCultivationData;
134 }
  
```

#### 6.10.3.12 void MadDataset::setDescription ( QString theDescription )

Definition at line 126 of file maddataset.cpp.

```

127 {
128     mDescription = theDescription;
129 }
  
```

6.10.3.13 void MadGuid::setGuid ( QString *theGuid* = "" ) [inherited]

[MadGuid::setGuid](#).

#### Parameters

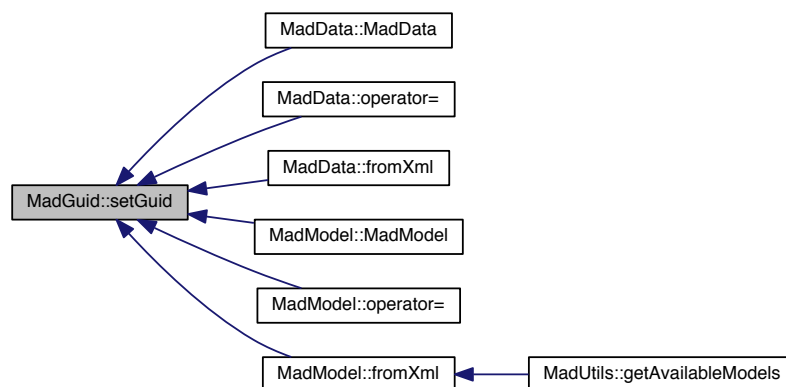
<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



6.10.3.14 void MadDataset::setInitialValues ( MadDataClassificationInitialValues *theInitialValues* )

Definition at line 136 of file maddataset.cpp.

```

137 {
138     mInitialValues = theInitialValues;
139 }
```

6.10.3.15 void MadDataset::setName ( QString *theName* )

Definition at line 121 of file maddataset.cpp.

```

122 {
123     mName = theName;
124 }
```

**6.10.3.16 void MadDataset::setPhenology ( MadDataClassificationPhenology *thePhenologyData* )**

Definition at line 141 of file maddataset.cpp.

```
142 {  
143     mPhenology = thePhenologyData;  
144 }
```

**6.10.3.17 void MadDataset::setPrevCrop ( MadDataClassificationPrevCrop *thePrevCropData* )**

Definition at line 146 of file maddataset.cpp.

```
147 {  
148     mPrevCrop = thePrevCropData;  
149 }
```

**6.10.3.18 void MadDataset::setSiteData ( MadDataClassificationSiteData *theSiteData* )**

Definition at line 151 of file maddataset.cpp.

```
152 {  
153     mSiteData = theSiteData;  
154 }
```

**6.10.3.19 void MadDataset::setSoil ( MadDataClassificationSoil *theSoilData* )**

Definition at line 156 of file maddataset.cpp.

```
157 {  
158     mSoil = theSoilData;  
159 }
```

**6.10.3.20 void MadDataset::setStateVars ( MadStateVars *theStateVarsData* )**

Definition at line 166 of file maddataset.cpp.

```
167 {  
168     mStateVars = theStateVarsData;  
169 }
```

**6.10.3.21 void MadDataset::setWeather ( MadDataClassificationWeather *theWeatherData* )**

Definition at line 161 of file maddataset.cpp.

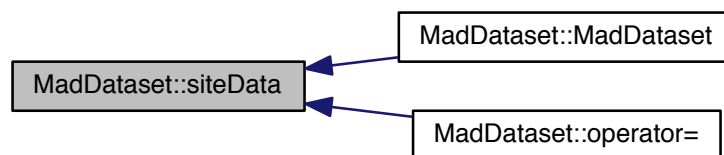
```
162 {  
163     mWeather = theWeatherData;  
164 }
```

#### 6.10.3.22 MadDataClassificationSiteData MadDataset::siteData ( ) const

Definition at line 102 of file maddataset.cpp.

```
103 {  
104     return mSiteData;  
105 }
```

Here is the caller graph for this function:

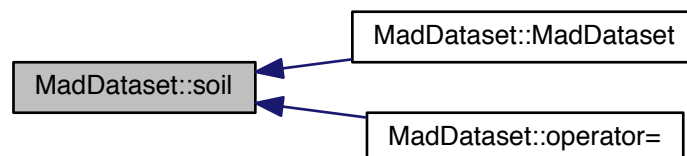


#### 6.10.3.23 MadDataClassificationSoil MadDataset::soil ( ) const

Definition at line 106 of file maddataset.cpp.

```
107 {  
108     return mSoil;  
109 }
```

Here is the caller graph for this function:

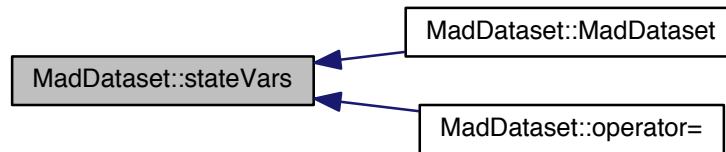


#### 6.10.3.24 MadStateVars MadDataset::stateVars ( ) const

Definition at line 114 of file maddataset.cpp.

```
115 {  
116     return mStateVars;  
117 }
```

Here is the caller graph for this function:



#### 6.10.3.25 QString MadDataset::toHtml ( )

Return a html text representation of this layer

Definition at line 217 of file maddataset.cpp.

```

218 {
219     QString myString;
220     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
221     //myString+="<p>GUID: " + guid() + "</p>";
222     myString+="<table>";
223     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
224
225     //
226     // the following shows example of how to do a couple of things
227     //
228
229     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
230     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
231     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
232     //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
233     //    QString::number(mCropFodderProduction) + "</td></tr>";
234     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
235     //    "</td></tr>";
236     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
237     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
238     myString+="</table>";
239     return myString;
240 }
```

#### 6.10.3.26 QString MadDataset::toText ( )

Return a plain text representation of this layer

Definition at line 208 of file maddataset.cpp.

```

209 {
210     QString myString;
211     myString+=QString("guid=>" + guid() + "\n");
212     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
213     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
214     return myString;
215 }
```



Here is the call graph for this function:



#### 6.10.3.27 QString MadDataset::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

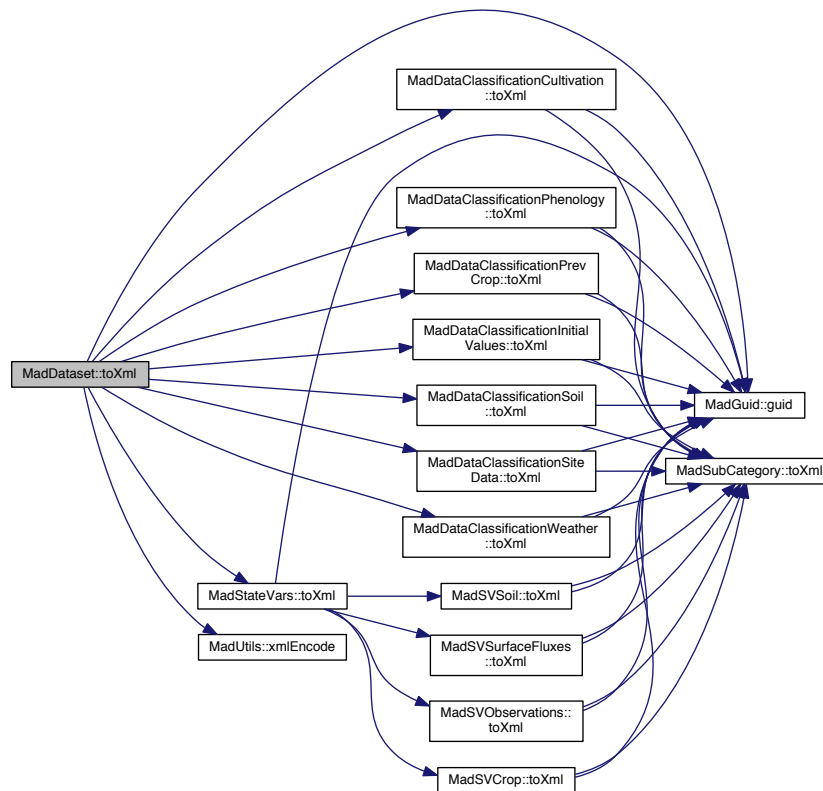
Implements [MadSerialisable](#).

Definition at line 190 of file maddataset.cpp.

```

191 {
192     QString myString;
193     myString+=QString("<dataset guid=\"" + guid() + "\">\n");
194     myString+=QString("  <name>" + MadUtils::xmlEncode(mName) + "</name>\n");
195     myString+=QString("  <description>" + MadUtils::xmlEncode(mDescription) + "\n");
196     myString+=mCultivation.toXml();
197     myString+=mPhenology.toXml();
198     myString+=mPrevCrop.toXml();
199     myString+=mInitialValues.toXml();
200     myString+=mSoil.toXml();
201     myString+=mSiteData.toXml();
202     myString+=mWeather.toXml();
203     myString+=mStateVars.toXml();
204     myString+=QString("</dataset>\n");
205     return myString;
206 }
  
```

Here is the call graph for this function:



#### 6.10.3.28 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

toXmlFile writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );

```

```

64     myQTextStream << this->toXml();
65     myFile.close();
66     myResult=true;
67 }
68 else
69 {
70     //@TODO Error handler!
71     myResult=false;
72 }
73 return myResult ;
74 }

```

Here is the call graph for this function:



#### 6.10.3.29 MadDataClassificationWeather MadDataset::weather ( ) const

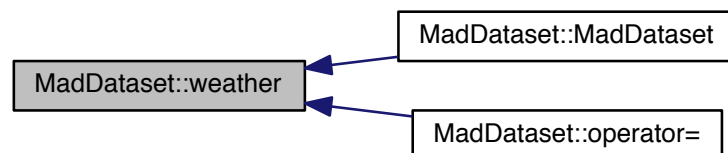
Definition at line 110 of file maddataset.cpp.

```

111 {
112     return mWeather;
113 }

```

Here is the caller graph for this function:



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

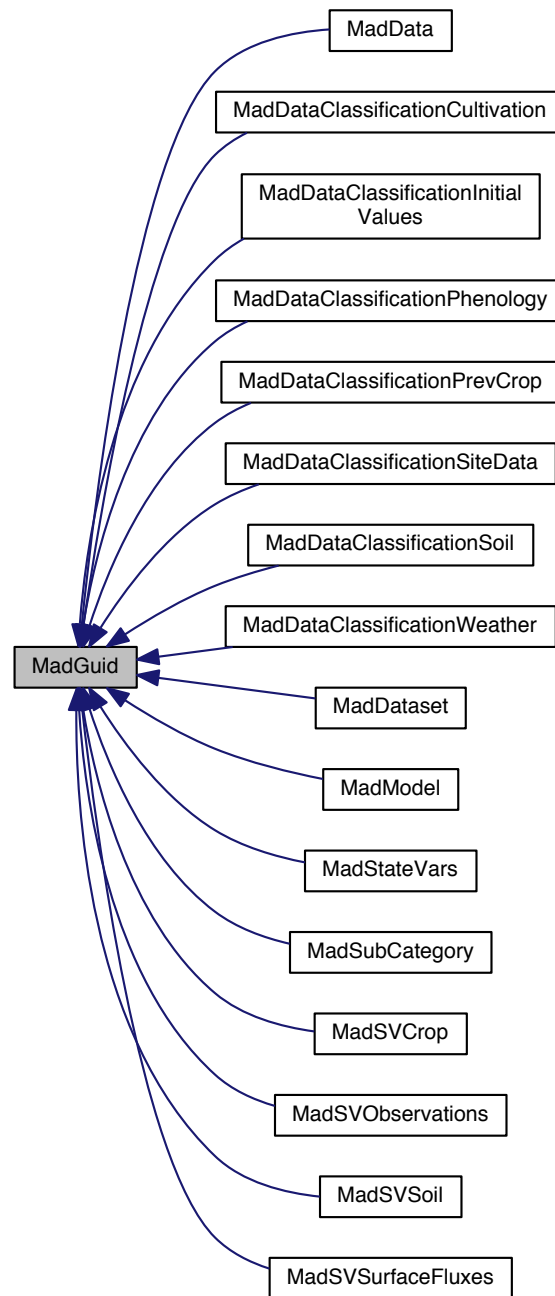
- `/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.h`
- `/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.cpp`

## 6.11 MadGuid Class Reference

The `MadGuid` class An abstract base class that has a Globally Unique Identifier (GUID) to represent a unique instance.

```
#include <madguid.h>
```

Inheritance diagram for MadGuid:



## Public Member Functions

- [MadGuid](#) ()
- [QString guid](#) () const  
*MadGuid::guid.*
- void [setGuid](#) (QString theGuid="")  
*MadGuid::setGuid.*

### 6.11.1 Detailed Description

The [MadGuid](#) class An abstract base class that has a Globally Unique Identifier (GUID) to represent a unique instance.

Definition at line 32 of file madguid.h.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 MadGuid::MadGuid ( )

Constructor

Definition at line 28 of file madguid.cpp.

```
29 {
30 }
```

### 6.11.3 Member Function Documentation

#### 6.11.3.1 QString MadGuid::guid ( ) const

[MadGuid::guid.](#)

Destructor Retrieve the GUID

Returns

Definition at line 40 of file madguid.cpp.

```
41 {
42     return mGuid;
43 }
```

#### 6.11.3.2 void MadGuid::setGuid ( QString *theGuid* = " " )

[MadGuid::setGuid.](#)

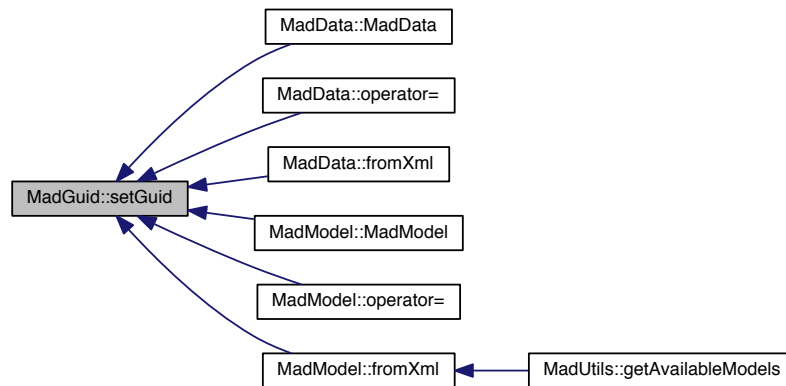
Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```
50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
```

Here is the caller graph for this function:



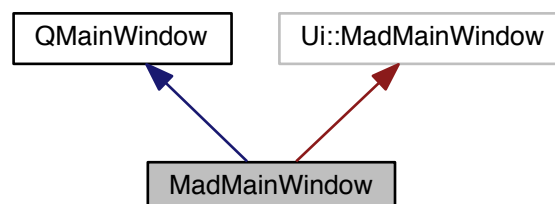
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.cpp](#)

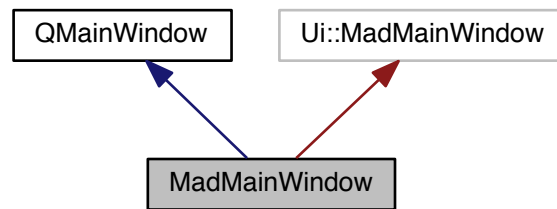
## 6.12 MadMainWindow Class Reference

```
#include <madmainwindow.h>
```

Inheritance diagram for MadMainWindow:



Collaboration diagram for MadMainWindow:



## Public Member Functions

- [MadMainWindow](#) (QWidget \*parent=0)
- QString [modelText](#) () const
- void [setModelText](#) (QString theModelText)

## Protected Member Functions

- void [changeEvent](#) (QEvent \*e)  
*changeEvent for translations in the future*

### 6.12.1 Detailed Description

This is the main GUI class

#### Author

Jason Jorgenson

Definition at line 44 of file madmainwindow.h.

### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 MadMainWindow::MadMainWindow ( QWidget \* parent = 0 ) [explicit]

This is the main form GUI of MAD (Macsur ADapter) It sets up the required slot connections and initialises the GUI

#### Parameters

<i>parent</i>	
---------------	--

Definition at line 35 of file madmainwindow.cpp.

```

36                                     :
37     QMainWindow(parent)
38 {
39     setupUi(this);
40     // the key to making the revision autoupdate is to use a feature in svn
41     // that will update keywords on commits. to make this work, you need to:
42     // svn propset svn:keywords "Revision" madmainwindow.cpp
  
```

```

43 // and then it works! note that you need to 'touch' madmainform.cpp every
44 // commit for this to work. This could be simply adding/removing a LF in
45 // this file (madmainform.cpp)
46 lblVersion->setText(QString("Version: %1").arg(VERSION) + " "
47                  + QString("$Revision: 141 $").replace("$", ""));
48 }

```

### 6.12.3 Member Function Documentation

#### 6.12.3.1 void MadMainWindow::changeEvent ( QEvent \* e ) [protected]

changeEvent for translations in the future

##### Parameters

e	
---	--

Definition at line 59 of file madmainwindow.cpp.

```

60 {
61     QMainWindow::changeEvent(e);
62     switch (e->type()) {
63     case QEvent::LanguageChange:
64         retranslateUi(this);
65         break;
66     default:
67         break;
68     }
69 }

```

#### 6.12.3.2 QString MadMainWindow::modelText ( ) const

Definition at line 49 of file madmainwindow.cpp.

```

50 {
51     return mModelText;
52 }

```

#### 6.12.3.3 void MadMainWindow::setModelText ( QString theModelText )

Definition at line 54 of file madmainwindow.cpp.

```

55 {
56     mModelText=theModelText;
57 }

```

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

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.h
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.cpp

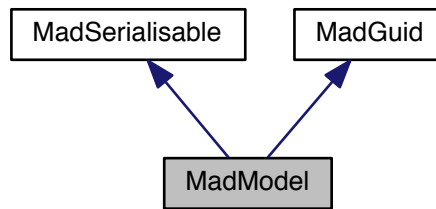
## 6.13 MadModel Class Reference

The [MadModel](#) class, to represent a ModelTheme.

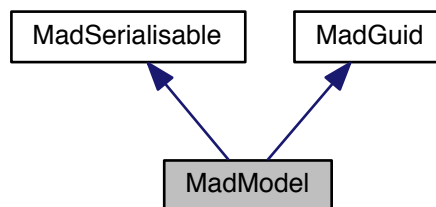
```
#include <madmodel.h>
```



Inheritance diagram for MadModel:



Collaboration diagram for MadModel:



## Public Member Functions

- [MadModel](#) ()
- [MadModel](#) (const [MadModel](#) &theModel)
- [MadModel](#) & [operator=](#) (const [MadModel](#) &theModel)
- [QString](#) [name](#) () const
- [QString](#) [description](#) () const
- [QString](#) [imageFile](#) () const
- void [setName](#) ([QString](#) theName)
- void [setDescription](#) ([QString](#) theDescription)
- void [setImageFile](#) ([QString](#) theImageFileName)
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- bool [fromXml](#) (const [QString](#) theXml)
- virtual bool [toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- virtual bool [fromXmlFile](#) (const [QString](#) theFileName)
- *fromXmlFile Read this object from xml in a file*
- [QString](#) [guid](#) () const

*MadGuid::guid.*

- void `setGuid` (QString theGuid="")

*MadGuid::setGuid.*

### 6.13.1 Detailed Description

The `MadModel` class, to represent a ModelTheme.

Definition at line 56 of file madmodel.h.

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 MadModel::MadModel ( )

Constructor .

Definition at line 33 of file madmodel.cpp.

```

33             : MadSerialisable(), MadGuid()
34 {
35     setGuid();
36     mName="No Name Set";
37     mDescription="Not Set";
38 }
```

Here is the call graph for this function:



#### 6.13.2.2 MadModel::MadModel ( const MadModel & theModel )

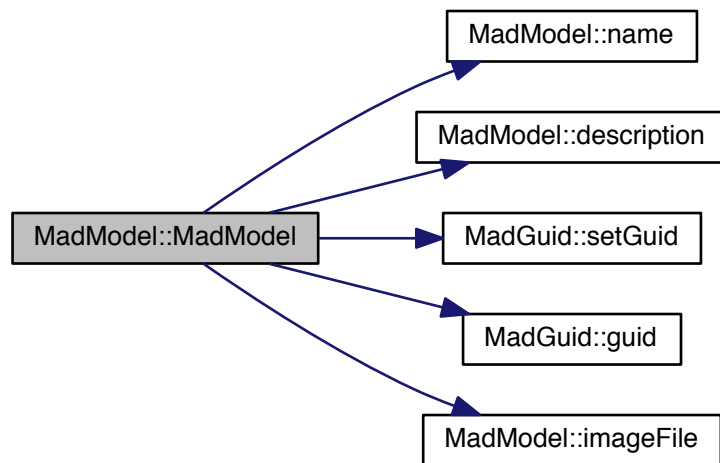
Destructor . copy constructor

Definition at line 46 of file madmodel.cpp.

```

47 {
48     mName=theModel.name();
49     mDescription=theModel.description();
50     setGuid(theModel.guid());
51     mImageFile=theModel.imageFile();
52 }
```

Here is the call graph for this function:



### 6.13.3 Member Function Documentation

#### 6.13.3.1 QString MadModel::description ( ) const

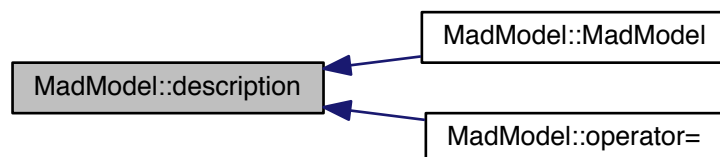
The description of this model

Definition at line 70 of file madmodel.cpp.

```

71 {
72     return mDescription;
73 }
  
```

Here is the caller graph for this function:



#### 6.13.3.2 bool MadModel::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

## See Also

[MadSerialisable](#)

## Note

this class inherits the serialisable interface so it MUST implement this

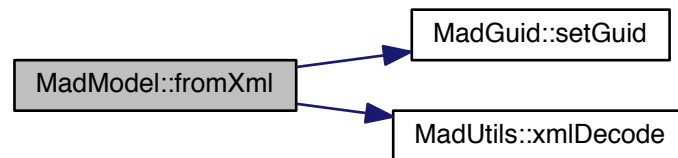
Implements [MadSerialisable](#).

Definition at line 97 of file madmodel.cpp.

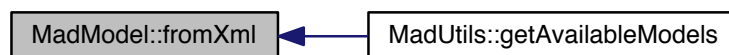
```

98 {
99     QDomDocument myDocument("mydocument");
100     myDocument.setContent(theXml);
101     QDomElement myTopElement = myDocument.firstChildElement("model");
102     if (myTopElement.isNull())
103     {
104         //TODO - just make this a warning
105         qDebug("the top element couldn't be found!");
106         setGuid(myTopElement.attribute("guid"));
107         mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
108         mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").
text());
109         mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
110         return true;
111     }
112     else
113         return false;
114 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



**6.13.3.3** `bool MadSerialisable::fromXmlFile ( const QString theFileName )` `[virtual]`, `[inherited]`

`fromXmlFile` Read this object from xml in a file

## See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

## Parameters

<i>theFileName</i>	
--------------------	--

## Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



#### 6.13.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid](#).

Destructor Retrieve the GUID

## Returns

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

#### 6.13.3.5 QString MadModel::imageFile ( ) const

The image file associated with the model

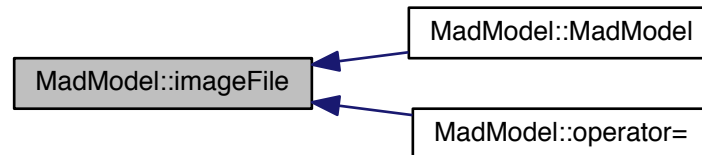
Definition at line 75 of file madmodel.cpp.

```

76 {
77     return mImageFile;
78 }

```

Here is the caller graph for this function:



#### 6.13.3.6 QString MadModel::name ( ) const

The name of this model

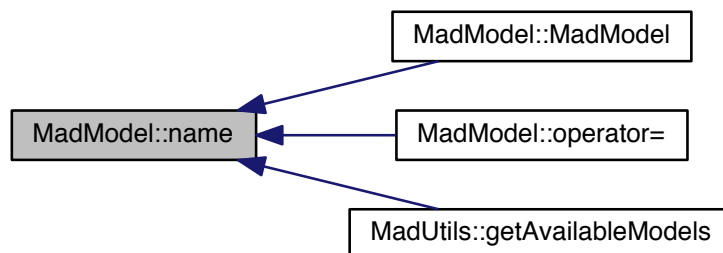
Definition at line 65 of file madmodel.cpp.

```

66 {
67     return mName;
68 }

```

Here is the caller graph for this function:



#### 6.13.3.7 MadModel & MadModel::operator= ( const MadModel & theModel )

Assignment operator

Definition at line 54 of file madmodel.cpp.

```

55 {
56     if (this == &theModel) return *this; // gracefully handles self assignment
57
58     mName=theModel.name();

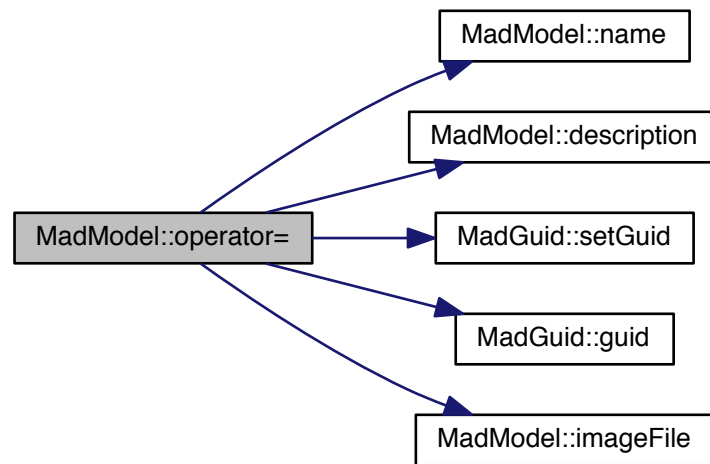
```

```

59     mDescription=theModel.description();
60     setGuid(theModel.guid());
61     mImageFile=theModel.imageFile();
62     return *this;
63 }

```

Here is the call graph for this function:



#### 6.13.3.8 void MadModel::setDescription ( QString *theDescription* )

Set the model description

See Also

[description\(\)](#)

Definition at line 87 of file madmodel.cpp.

```

88 {
89     mDescription=theDescription;
90 }

```

#### 6.13.3.9 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {

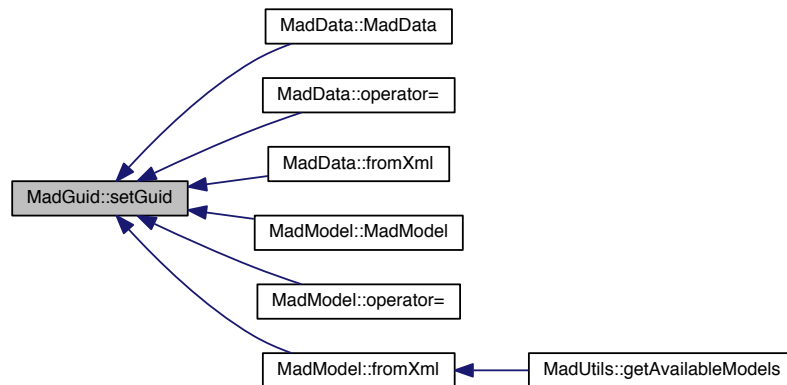
```

```

53         mGuid=QUuid::createUuid().toString().replace("{","").replace("}","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:



#### 6.13.3.10 void MadModel::setImageFile ( QString *theImageFileName* )

Set the image file

See Also

[imageFile\(\)](#)

Definition at line 92 of file `madmodel.cpp`.

```

93 {
94     mImageFile=theImageFileName;
95 }

```

#### 6.13.3.11 void MadModel::setName ( QString *theName* )

Set the modelName

See Also

[name\(\)](#)

Definition at line 82 of file `madmodel.cpp`.

```

83 {
84     mName=theName;
85 }

```



## 6.13.3.12 QString MadModel::toHtml ( )

Return a html text representation of this layer

Definition at line 147 of file madmodel.cpp.

```

148 {
149     QString myString;
150     myString+="

### 


```

Here is the call graph for this function:



## 6.13.3.13 QString MadModel::toText ( )

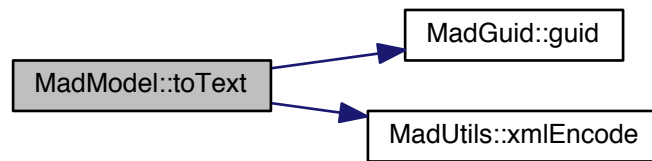
Return a plain text representation of this layer

Definition at line 138 of file madmodel.cpp.

```

139 {
140     QString myString;
141     myString+=QString("guid=>" + guid() + "\n");
142     myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
143     myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
144     return myString;
145 }
```

Here is the call graph for this function:



#### 6.13.3.14 QString MadModel::toXml ( ) [virtual]

Return an xml representation of this layer

##### Note

this class inherits the serialisable interface so it MUST implement this

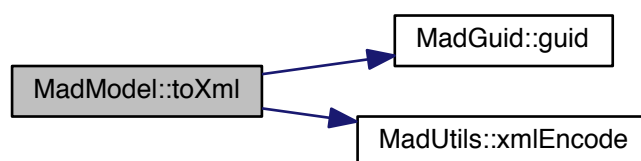
Implements [MadSerialisable](#).

Definition at line 116 of file madmodel.cpp.

```

117 {
118     QString myString;
119     myString+=QString("<model guid=\"" + guid() + "\">\n");
120     myString+=QString("  <name>" + MadUtils::xmlEncode(mName) + "</name>\n");
121     myString+=QString("  <description>" + MadUtils::xmlEncode(mDescription) + "
122     </description>\n");
123
124     // switch (mAreaUnits)
125     // {
126     //     case Dunum:
127     //         myString+=QString("    <areaUnits>Dunum</areaUnits>\n");
128     //         break;
129     //     case Hectare:
130     //         myString+=QString("    <areaUnits>Hectare</areaUnits>\n");
131     //         break;
132     // }
133     myString+=QString("  <imageFile>" + MadUtils::xmlEncode(mImageFile) + "</imageFile>\n"
134     );
135     myString+=QString("</model>\n");
136     return myString;
137 }
```

Here is the call graph for this function:



### 6.13.3.15 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:



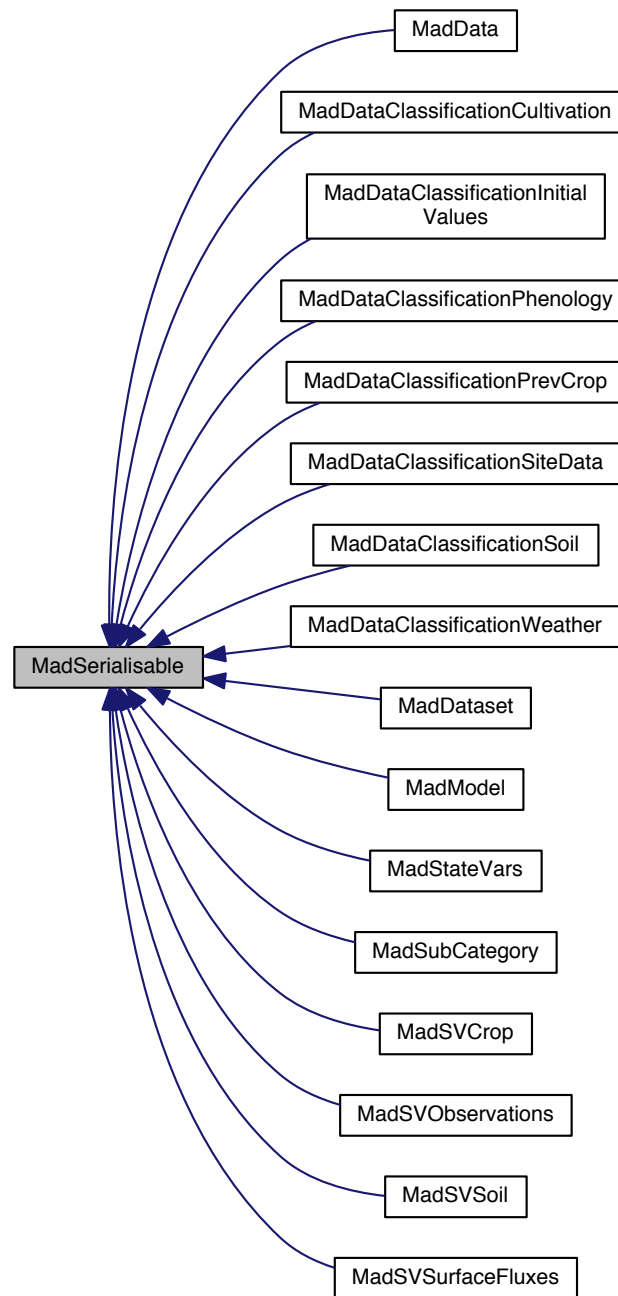
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[madmodel.h](#)
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[madmodel.cpp](#)

## 6.14 MadSerialisable Class Reference

```
#include <madserialisable.h>
```

Inheritance diagram for MadSerialisable:



## Public Member Functions

- [MadSerialisable](#) ()  
*MadSerialisable* Constructor.
- virtual QString [toXml](#) ()=0  
*toXml* Write this object to xml and return result as qstring (virtual)
- virtual bool [toXmlFile](#) (const QString theFileName)

*toXmlFile* writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

- virtual bool [fromXml](#) (const QString theXml)=0  
*fromXml* Read this object from xml
- virtual bool [fromXmlFile](#) (const QString theFileName)  
*fromXmlFile* Read this object from xml in a file

### 6.14.1 Detailed Description

An abstract base class for any class that is serialiseable to xml

#### Author

Tim Sutton, Jason Jorgenson

Definition at line 50 of file madserialisable.h.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 MadSerialisable::MadSerialisable ( )

[MadSerialisable](#) Constructor.

Definition at line 49 of file madserialisable.cpp.

```
50 {
51 }
```

### 6.14.3 Member Function Documentation

#### 6.14.3.1 virtual bool MadSerialisable::fromXml ( const QString *theXml* ) [pure virtual]

*fromXml* Read this object from xml

#### Parameters

<i>theXml</i>	
---------------	--

**Returns**

result as true for success, false for failure (virtual)

Implemented in [MadModel](#), [MadData](#), [MadSVCrop](#), [MadSVSurfaceFluxes](#), [MadDataset](#), [MadStateVars](#), [MadDataClassificationWeather](#), [MadDataClassificationCultivation](#), [MadDataClassificationSoil](#), [MadDataClassificationPrevCrop](#), [MadSVSoil](#), [MadDataClassificationSiteData](#), [MadSVObservations](#), [MadDataClassificationInitialValues](#), [MadSubCategory](#), and [MadDataClassificationPhenology](#).

Here is the caller graph for this function:



### 6.14.3.2 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual]

fromXmlFile Read this object from xml in a file

**See Also**

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

**Parameters**

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
  
```

Here is the call graph for this function:



#### 6.14.3.3 `virtual QString MadSerialisable::toXml ( ) [pure virtual]`

`toXml` Write this object to xml and return result as qstring (virtual)

Destructor .

Returns

Implemented in [MadModel](#), [MadData](#), [MadSVCrop](#), [MadSVSurfaceFluxes](#), [MadDataset](#), [MadStateVars](#), [MadDataClassificationWeather](#), [MadDataClassificationCultivation](#), [MadDataClassificationSoil](#), [MadDataClassificationPrevCrop](#), [MadSVSoil](#), [MadDataClassificationSiteData](#), [MadSVObservations](#), [MadDataClassificationInitialValues](#), [MadSubCategory](#), and [MadDataClassificationPhenology](#).

Here is the caller graph for this function:



#### 6.14.3.4 `bool MadSerialisable::toXmlFile ( const QString theFileName ) [virtual]`

`toXmlFile` writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

See Also

[toXml\(\)](#)

Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```
58 {  
59     bool myResult = false;  
60     QFile myFile( theFileName );  
61     if ( myFile.open( QIODevice::WriteOnly ) )  
62     {  
63         QTextStream myQTextStream( &myFile );  
64         myQTextStream << this->toXml();  
65         myFile.close();  
66         myResult=true;  
67     }  
68     else  
69     {  
70         //@TODO Error handler!  
71         myResult=false;  
72     }  
73     return myResult ;  
74 }
```

Here is the call graph for this function:



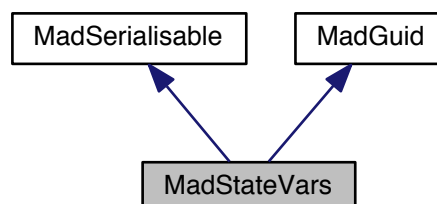
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[madserialisable.h](#)
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/[madserialisable.cpp](#)

## 6.15 MadStateVars Class Reference

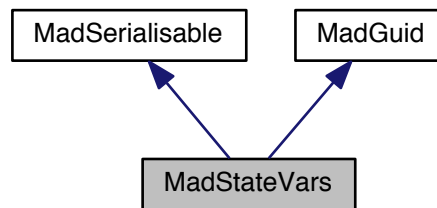
```
#include <madstatevars.h>
```

Inheritance diagram for MadStateVars:





Collaboration diagram for MadStateVars:



## Public Member Functions

- [MadStateVars](#) ()
- [MadStateVars](#) (const [MadStateVars](#) &theData)
- [MadStateVars](#) & [operator=](#) (const [MadStateVars](#) &theData)
- [MadSVCrop](#) [cropCategories](#) () const
- [MadSVSoil](#) [soilCategories](#) () const
- [MadSVSurfaceFluxes](#) [surfaceFluxesCategories](#) () const
- [MadSVObservations](#) [observationCategories](#) () const
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- [bool](#) [fromXml](#) (const [QString](#) theXml)
- [void](#) [setCropCategories](#) ([MadSVCrop](#) theMadSVCrop)
- [void](#) [setSoilCategories](#) ([MadSVSoil](#) theData)
- [void](#) [setSurfaceFluxesCategories](#) ([MadSVSurfaceFluxes](#) theData)
- [void](#) [setObservationCategories](#) ([MadSVObservations](#) theData)
- [virtual bool](#) [toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile* writes *object* to *xml* and return result (virtual *qstring*) We provide a basic default implementation where given a file name, we will write the serialised *xml* to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

- [virtual bool](#) [fromXmlFile](#) (const [QString](#) theFileName)

*fromXmlFile* Read this object from *xml* in a file

- [QString](#) [guid](#) () const
- [void](#) [setGuid](#) ([QString](#) theGuid="")

*MadGuid::guid.*

*MadGuid::setGuid.*

### 6.15.1 Detailed Description

Definition at line 43 of file madstatevars.h.

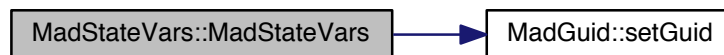
## 6.15.2 Constructor & Destructor Documentation

### 6.15.2.1 MadStateVars::MadStateVars ( )

Definition at line 36 of file madstatevars.cpp.

```
36                                     : MadSerialisable(), MadGuid()  
37 {  
38     setGuid();  
39 }
```

Here is the call graph for this function:

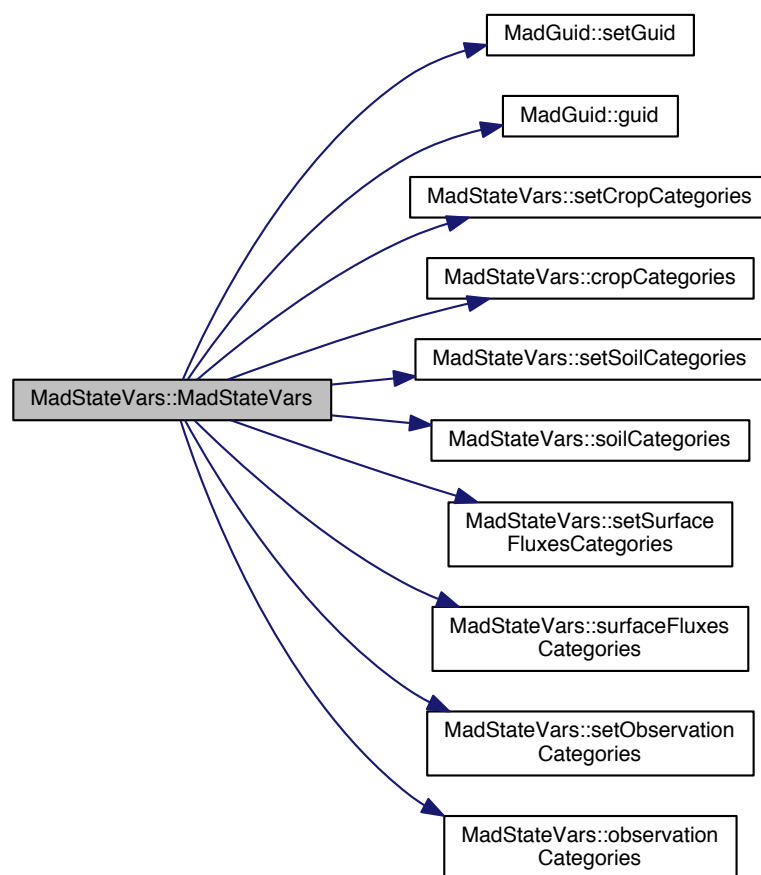


### 6.15.2.2 MadStateVars::MadStateVars ( const MadStateVars & theData )

Definition at line 41 of file madstatevars.cpp.

```
42 {  
43     setGuid(theData.guid());  
44     setCropCategories(theData.cropCategories());  
45     setSoilCategories(theData.soilCategories());  
46     setSurfaceFluxesCategories(theData.  
        surfaceFluxesCategories());  
47     setObservationCategories(theData.observationCategories());  
48 }
```

Here is the call graph for this function:



## 6.15.3 Member Function Documentation

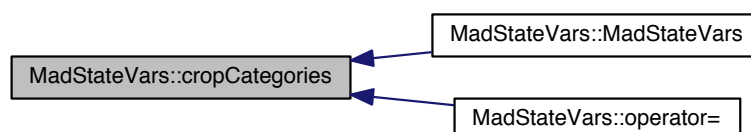
### 6.15.3.1 MadSVCrop MadStateVars::cropCategories ( ) const

Definition at line 64 of file `madstatevars.cpp`.

```

65 {
66     return mCropCategories;
67 }
```

Here is the caller graph for this function:



### 6.15.3.2 bool MadStateVars::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

#### See Also

[MadSerialisable](#)

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 102 of file madstatevars.cpp.

```

103 {
104     QDomDocument myDocument("mydocument");
105     myDocument.setContent(theXml);
106     QDomElement myTopElement = myDocument.firstChildElement("statevars");
107     if (myTopElement.isNull())
108     {
109         //TODO - just make this a warning
110         qDebug("the top element couldn't be found!");
111         setGuid(myTopElement.attribute("guid"));
112         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
113         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
114         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
115         return true;
116     }
117     else
118         return false;
119 }
```

Here is the call graph for this function:



### 6.15.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



#### 6.15.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

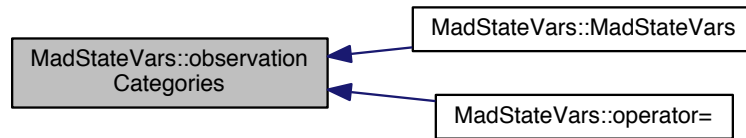
#### 6.15.3.5 MadSVObservations MadStateVars::observationCategories ( ) const

Definition at line 76 of file madstatevars.cpp.

```

77 {
78     return mObservations;
79 }
```

Here is the caller graph for this function:



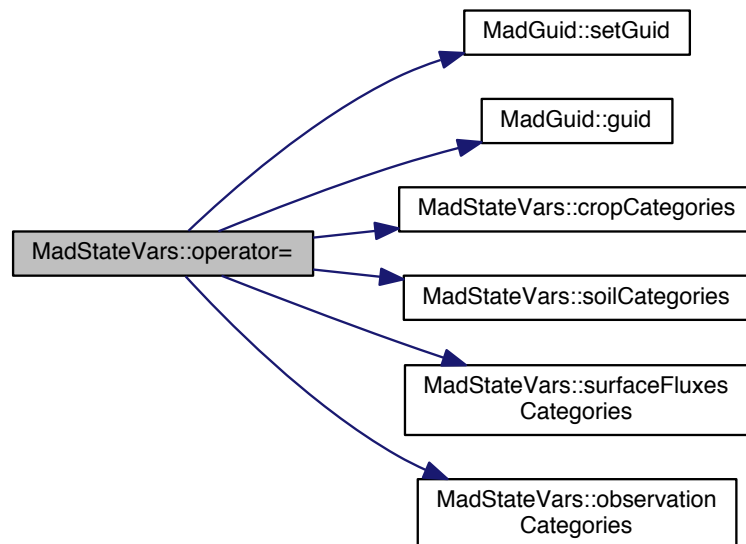
#### 6.15.3.6 MadStateVars & MadStateVars::operator= ( const MadStateVars & theData )

Definition at line 50 of file madstatevars.cpp.

```

51 {
52     // gracefully handles self assignment
53     if (this == &theData) return *this;
54     setGuid(theData.guid());
55     mCropCategories=theData.cropCategories();
56     mSoilCategories=theData.soilCategories();
57     mSurfaceFluxes=theData.surfaceFluxesCategories();
58     mObservations=theData.observationCategories();
59     return *this;
60 }
  
```

Here is the call graph for this function:



#### 6.15.3.7 void MadStateVars::setCropCategories ( MadSVCrop theMadSVCrop )

Definition at line 83 of file madstatevars.cpp.

```

84 {
85     mCropCategories = theMadSVCrop;
86 }

```

Here is the caller graph for this function:



#### 6.15.3.8 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

##### Parameters

<i>theGuid</i>	
----------------	--

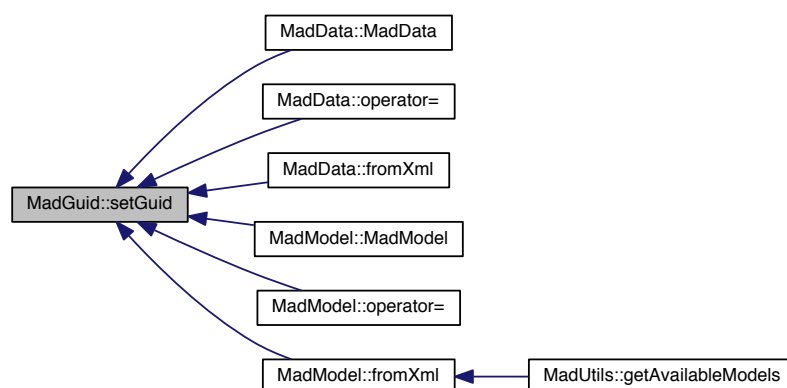
Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:

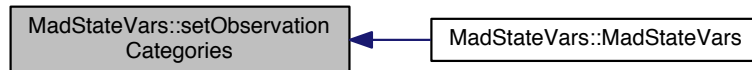


#### 6.15.3.9 void MadStateVars::setObservationCategories ( MadSVObservations *theData* )

Definition at line 97 of file madstatevars.cpp.

```
98 {  
99   mObservations = theData;  
100 }
```

Here is the caller graph for this function:



#### 6.15.3.10 void MadStateVars::setSoilCategories ( MadSVSoil *theData* )

Definition at line 87 of file madstatevars.cpp.

```
88 {  
89   mSoilCategories = theData;  
90 }
```

Here is the caller graph for this function:

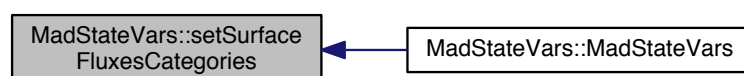


#### 6.15.3.11 void MadStateVars::setSurfaceFluxesCategories ( MadSVSurfaceFluxes *theData* )

Definition at line 92 of file madstatevars.cpp.

```
93 {  
94   mSurfaceFluxes = theMadSVSurfaceFluxes;  
95 }
```

Here is the caller graph for this function:



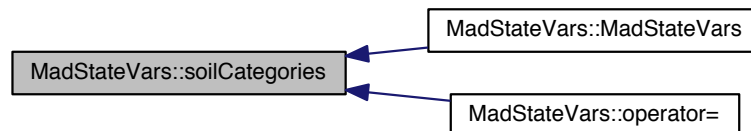


**6.15.3.12 MadSVSoil MadStateVars::soilCategories ( ) const**

Definition at line 68 of file madstatevars.cpp.

```
69 {
70     return mSoilCategories;
71 }
```

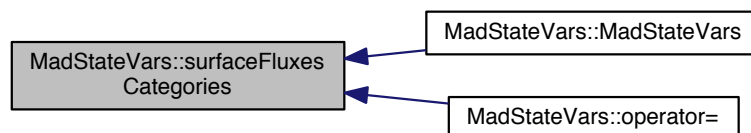
Here is the caller graph for this function:

**6.15.3.13 MadSVSurfaceFluxes MadStateVars::surfaceFluxesCategories ( ) const**

Definition at line 72 of file madstatevars.cpp.

```
73 {
74     return mSurfaceFluxes;
75 }
```

Here is the caller graph for this function:

**6.15.3.14 QString MadStateVars::toHtml ( )**

Return a html text representation of this layer

**6.15.3.15 QString MadStateVars::toText ( )**

Return a plain text representation of this layer

Definition at line 147 of file madstatevars.cpp.

```
148 {
149     QString myString;
150     myString+=QString("guid=>" + guid() + "\n");
151 }
```

```

151 //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
152 //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
153 return myString;
154 }

```

Here is the call graph for this function:



#### 6.15.3.16 QString MadStateVars::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

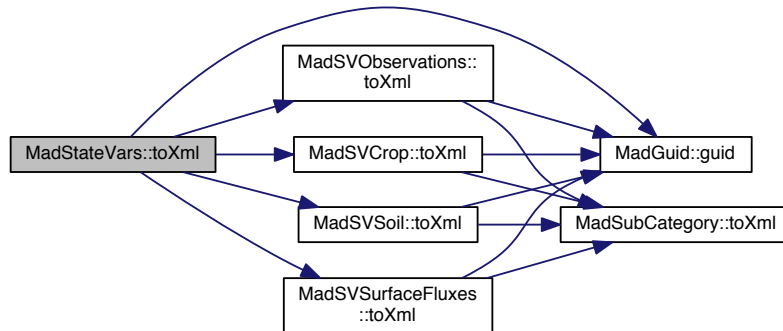
Definition at line 121 of file madstatevars.cpp.

```

122 {
123     QString myString;
124     myString+=QString(" <statevars guid=\"" + guid() + "\">\n");
125
126     // myString+=QString("    <crop>\n");
127     myString+=mCropCategories.toXml();
128     // myString+=QString("    </crop>\n");
129
130     // myString+=QString("    <soil>\n");
131     myString+=mSoilCategories.toXml();
132     // myString+=QString("    </soil>\n");
133
134     // myString+=QString("    <surfacefluxes>\n");
135     myString+=mSurfaceFluxes.toXml();
136     // myString+=QString("    </surfacefluxes>\n");
137
138     // myString+=QString("    <observations>\n");
139     myString+=mObservations.toXml();
140     // myString+=QString("    </observations>\n");
141
142     myString+=QString(" </statevars>\n");
143     return myString;
144 }
145 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.15.3.17 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );

```

```
64     myQTextStream << this->toXml();
65     myFile.close();
66     myResult=true;
67 }
68 else
69 {
70     //TODO Error handler!
71     myResult=false;
72 }
73 return myResult ;
74 }
```

Here is the call graph for this function:



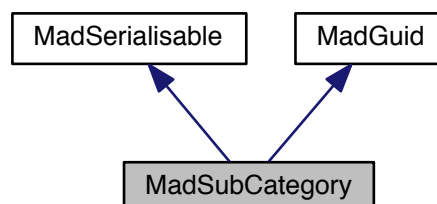
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.-h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.-cpp](#)

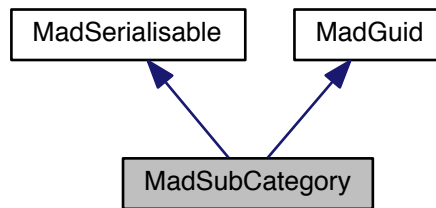
## 6.16 MadSubCategory Class Reference

```
#include <madsubcategory.h>
```

Inheritance diagram for MadSubCategory:



Collaboration diagram for MadSubCategory:



## Public Member Functions

- [MadSubCategory](#) ()
- [MadSubCategory](#) (const [MadSubCategory](#) &theSubCategory)
- [MadSubCategory](#) & [operator=](#) (const [MadSubCategory](#) &theData)
- bool [minData](#) () const
- float [depth](#) () const
- int [observations](#) () const
- float [weightPoints](#) () const
- int [replicates](#) () const
- QString [toXml](#) ()
- QString [toText](#) ()
- QString [toHtml](#) ()
- bool [fromXml](#) (const QString theXml)
- void [setMinData](#) (bool theBool)
- void [setDepth](#) (float theValue)
- void [setObservations](#) (int theValue)
- void [setWeightPoints](#) (float theValue)
- void [setReplicates](#) (int theValue)
- virtual bool [toXmlFile](#) (const QString theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- virtual bool [fromXmlFile](#) (const QString theFileName)

*fromXmlFile Read this object from xml in a file*

- QString [guid](#) () const
- [MadGuid::guid](#).
- void [setGuid](#) (QString theGuid="")
- [MadGuid::setGuid](#).

### 6.16.1 Detailed Description

Definition at line 32 of file `madsubcategory.h`.

## 6.16.2 Constructor & Destructor Documentation

### 6.16.2.1 MadSubCategory::MadSubCategory ( )

Definition at line 33 of file madsubcategory.cpp.

```
33                                     : MadSerialisable(), MadGuid()  
34 {  
35     setGuid();  
36     mMinData= 0;  
37     mDepth = 0.0;  
38     mObservations = 0;  
39     mWeightPoints = 0.0;  
40     mReplicates = 0;  
41 }
```

Here is the call graph for this function:

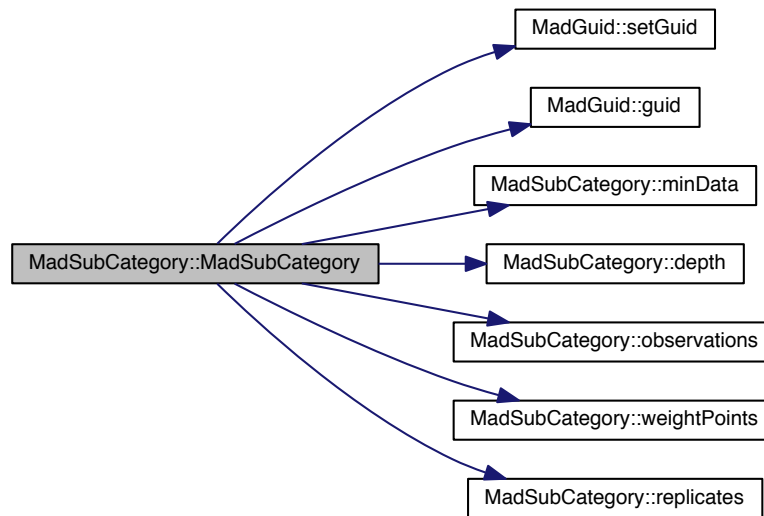


### 6.16.2.2 MadSubCategory::MadSubCategory ( const MadSubCategory & theSubCategory )

Definition at line 44 of file madsubcategory.cpp.

```
45 {  
46     setGuid(theSubCategory.guid());  
47     mMinData = theSubCategory.minData();  
48     mDepth = theSubCategory.depth();  
49     mObservations = theSubCategory.observations();  
50     mWeightPoints = theSubCategory.weightPoints();  
51     mReplicates = theSubCategory.replicates();  
52 }
```

Here is the call graph for this function:



### 6.16.3 Member Function Documentation

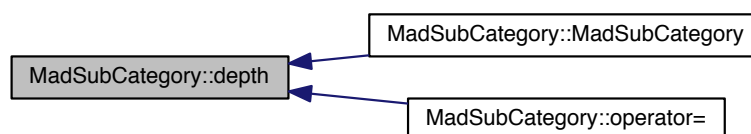
#### 6.16.3.1 float MadSubCategory::depth ( ) const

Definition at line 74 of file `madsubcategory.cpp`.

```

75 {
76     return mDepth;
77 }
  
```

Here is the caller graph for this function:



#### 6.16.3.2 bool MadSubCategory::fromXml ( const QString theXml ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

**Note**

this class inherits the serialisable interface so it MUST implement this

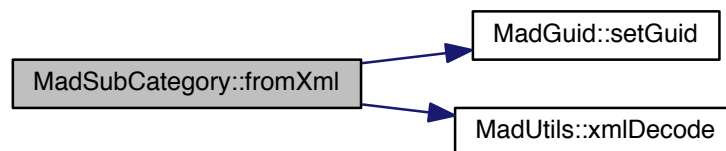
Implements [MadSerialisable](#).

Definition at line 120 of file madsubcategory.cpp.

```

121 {
122     QDomDocument myDocument("mydocument");
123     myDocument.setContent(theXml);
124     QDomElement myTopElement = myDocument.firstChildElement("details");
125     if (myTopElement.isNull())
126     {
127         // TODO - just make this a warning
128         qDebug("the top element couldn't be found!");
129         setGuid(myTopElement.attribute("guid"));
130     }
131     // the line below works and does the same as the line below it.
132     // (QString(myTopElement.firstChildElement("mindata").text())=="0" ? mMinData=false : mMinData=true;
133     mMinData = QString(myTopElement.firstChildElement("mindata").text()).toInt();
134
135     mDepth = MadUtils::xmlDecode(myTopElement.firstChildElement("depth").text()).toFloat();
136     mObservations = MadUtils::xmlDecode(myTopElement.firstChildElement("observations").text()).toInt();
137     mWeightPoints = MadUtils::xmlDecode(myTopElement.firstChildElement("weightpoints").text()).toFloat();
138     mReplicates = MadUtils::xmlDecode(myTopElement.firstChildElement("replicates").text()).toInt();
139
140     return true;
141 }
142 else
143     return false;
144 }
```

Here is the call graph for this function:



### 6.16.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

**See Also**

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

**Parameters**

<i>theFileName</i>	
--------------------	--



**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:

**6.16.3.4 QString MadGuid::guid ( ) const [inherited]**

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

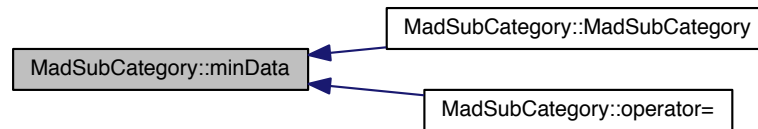
**6.16.3.5 bool MadSubCategory::minData ( ) const**

Definition at line 69 of file madsubcategory.cpp.

```

70 {
71     return mMinData;
72 }
```

Here is the caller graph for this function:



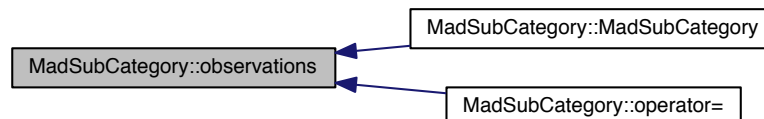
#### 6.16.3.6 int MadSubCategory::observations ( ) const

Definition at line 79 of file madsubcategory.cpp.

```

80 {
81     return mObservations;
82 }
  
```

Here is the caller graph for this function:



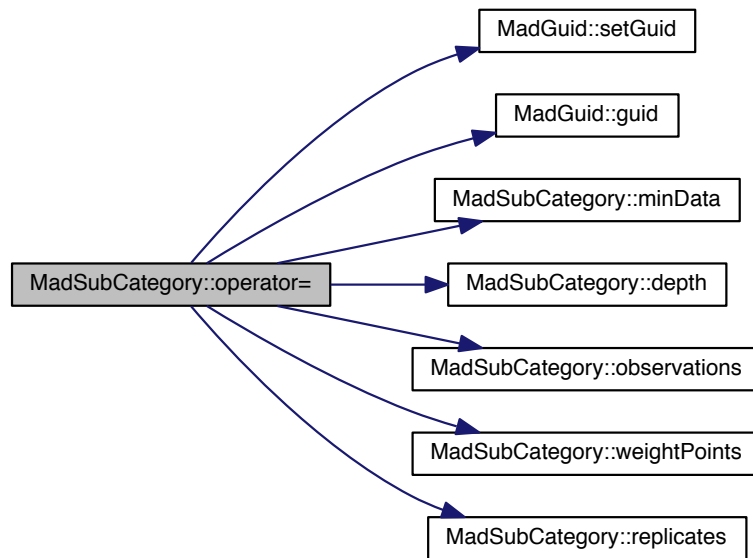
#### 6.16.3.7 MadSubCategory & MadSubCategory::operator= ( const MadSubCategory & theData )

Definition at line 54 of file madsubcategory.cpp.

```

55 {
56     // gracefully handles self assignment
57     if (this == &theSubCategory) return *this;
58     setGuid(theSubCategory.guid());
59     mMinData = theSubCategory.minData();
60     mDepth = theSubCategory.depth();
61     mObservations = theSubCategory.observations();
62     mWeightPoints = theSubCategory.weightPoints();
63     mReplicates = theSubCategory.replicates();
64     return *this;
65 }
  
```

Here is the call graph for this function:



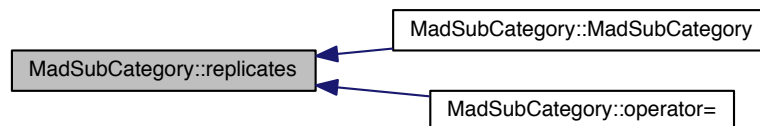
#### 6.16.3.8 `int MadSubCategory::replicates ( ) const`

Definition at line 89 of file `madsubcategory.cpp`.

```

90 {
91     return mReplicates;
92 }
  
```

Here is the caller graph for this function:



#### 6.16.3.9 `void MadSubCategory::setDepth ( float theValue )`

Definition at line 100 of file `madsubcategory.cpp`.

```

101 {
102     mDepth = theValue;
103 }
  
```

Here is the caller graph for this function:



**6.16.3.10** `void MadGuid::setGuid ( QString theGuid = " " )` [inherited]

[MadGuid::setGuid.](#)

#### Parameters

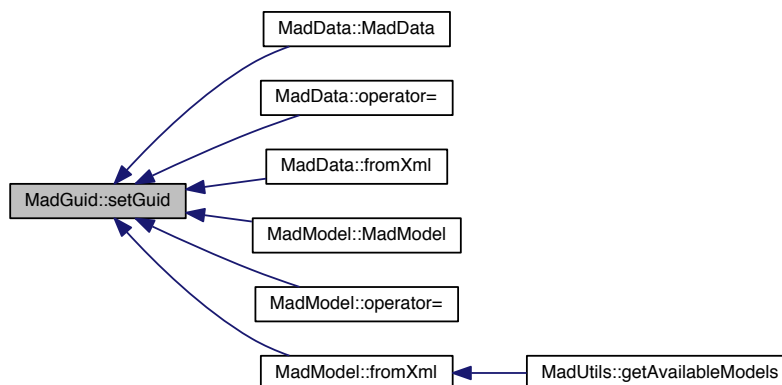
<i>theGuid</i>	
----------------	--

Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{", "").replace("}", "");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }
  
```

Here is the caller graph for this function:



**6.16.3.11** `void MadSubCategory::setMinData ( bool theBool )`

Definition at line 95 of file madsubcategory.cpp.

```

96 {
97     mMinData = theBool;
98 }
  
```

**6.16.3.12 void MadSubCategory::setObservations ( int *theValue* )**

Definition at line 105 of file madsubcategory.cpp.

```
106 {
107     mObservations = theValue;
108 }
```

**6.16.3.13 void MadSubCategory::setReplicates ( int *theValue* )**

Definition at line 115 of file madsubcategory.cpp.

```
116 {
117     mReplicates = theValue;
118 }
```

**6.16.3.14 void MadSubCategory::setWeightPoints ( float *theValue* )**

Definition at line 110 of file madsubcategory.cpp.

```
111 {
112     mWeightPoints = theValue;
113 }
```

**6.16.3.15 QString MadSubCategory::toHtml ( )**

Return a html text representation of this layer

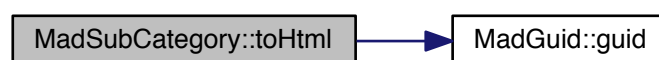
Definition at line 175 of file madsubcategory.cpp.

```
176 {
177     QString myString;
178     myString+="

### 


```

Here is the call graph for this function:



### 6.16.3.16 QString MadSubCategory::toText ( )

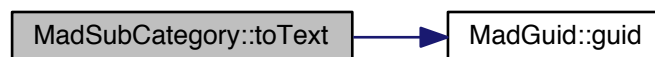
Return a plain text representation of this layer

Definition at line 160 of file madsubcategory.cpp.

```

161 {
162     QString myString;
163     myString+=QString("guid=>" + guid() + "\n");
164     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
165     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
166     //myString+=QString("<dataset guid=\" + guid() + \">\n");
167     myString+=QString("minData=>" + QString::number(mMinData) + "\n");
168     myString+=QString("depth=>" + QString::number(mDepth) + "\n");
169     myString+=QString("observations=>" + QString::number(mObservations) + "<\n");
170     myString+=QString("weightPoints=>" + QString::number(mWeightPoints) + "<\n");
171     myString+=QString("replicates=>" + QString::number(mReplicates) + "<\n");
172     return myString;
173 }
```

Here is the call graph for this function:



### 6.16.3.17 QString MadSubCategory::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

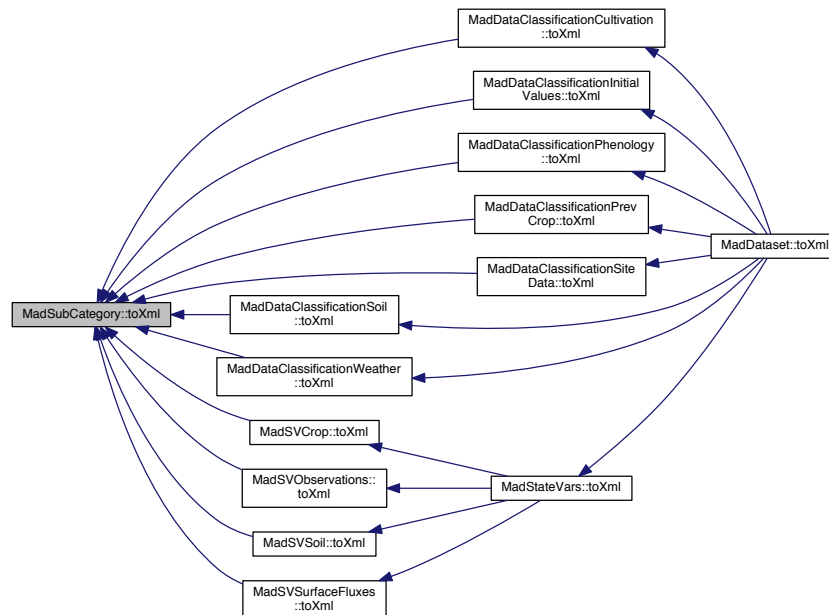
Implements [MadSerialisable](#).

Definition at line 146 of file madsubcategory.cpp.

```

147 {
148     QString myString;
149     myString+=QString("      <details>\n");
150     myString+=QString("      <mindata>" + QString::number(mMinData) + "</mindata>\n");
151     myString+=QString("      <depth>" + QString::number(mDepth) + "</depth>\n");
152     myString+=QString("      <observations>" + QString::number(mObservations) + "</observations>\n");
153     myString+=QString("      <weightpoints>" + QString::number(mWeightPoints) + "</weightpoints>\n");
154     myString+=QString("      <replicates>" + QString::number(mReplicates) + "</replicates>\n");
155     myString+=QString("      </details>\n");
156
157     return myString;
158 }
```

Here is the caller graph for this function:



#### 6.16.3.18 bool MadSerialisable::toXmlFile ( const QString theFileName ) [virtual],[inherited]

toXmlFile writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //@TODO Error handler!
71         myResult=false;
72     }

```

```

73     return myResult ;
74 }

```

Here is the call graph for this function:



#### 6.16.3.19 float MadSubCategory::weightPoints ( ) const

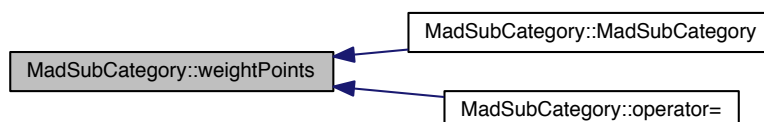
Definition at line 84 of file madsubcategory.cpp.

```

85 {
86     return mWeightPoints;
87 }

```

Here is the caller graph for this function:



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

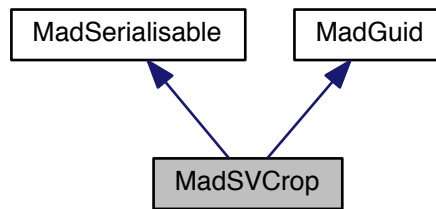
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcategory.cpp](#)

## 6.17 MadSVCrop Class Reference

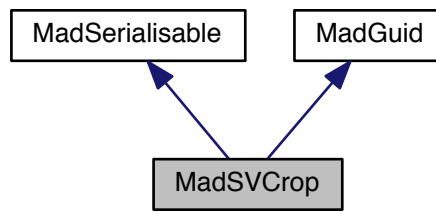
```
#include <madsvcrop.h>
```



Inheritance diagram for MadSVCrop:



Collaboration diagram for MadSVCrop:



## Public Member Functions

- [MadSVCrop](#) ()
- [MadSVCrop](#) (const [MadSVCrop](#) &theData)
- [MadSVCrop](#) & [operator=](#) (const [MadSVCrop](#) &theData)
- [MadSubCategory agrBiomass](#) () const  
*agrBiomass*
- [MadSubCategory weightOrgans](#) () const  
*weightOrgans*
- [MadSubCategory rootBiomass](#) () const  
*rootBiomass*
- [MadSubCategory nInAGrBiomass](#) () const  
*nInAGrBiomass*
- [MadSubCategory nInOrgans](#) () const  
*nInOrgans*
- [MadSubCategory lai](#) () const  
*lai*
- QString [toXml](#) ()
- QString [toText](#) ()
- QString [toHtml](#) ()
- bool [fromXml](#) (const QString theXml)

- void [setAgrBiomass](#) ([MadSubCategory](#) theData)  
*setAgrBiomass*
- void [setWeightOrgans](#) ([MadSubCategory](#) theData)  
*setWeightOrgans*
- void [setRootBiomass](#) ([MadSubCategory](#) theData)  
*setRootBiomass*
- void [setNlnAGrBiomass](#) ([MadSubCategory](#) theData)  
*setNlnAGrBiomass*
- void [setNlnOrgans](#) ([MadSubCategory](#) theData)  
*setNlnOrgans*
- void [setLai](#) ([MadSubCategory](#) theData)  
*setLai*
- virtual bool [toXmlFile](#) (const QString theFileName)  
*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*
- virtual bool [fromXmlFile](#) (const QString theFileName)  
*fromXmlFile Read this object from xml in a file*
- QString [guid](#) () const  
*MadGuid::guid.*
- void [setGuid](#) (QString theGuid="")  
*MadGuid::setGuid.*

### 6.17.1 Detailed Description

Definition at line 35 of file madsvcrop.h.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 MadSVCrop::MadSVCrop ( )

Definition at line 33 of file madsvcrop.cpp.

```

33             : MadSerialisable(), MadGuid()
34 {
35     setGuid();
36 }
```

Here is the call graph for this function:

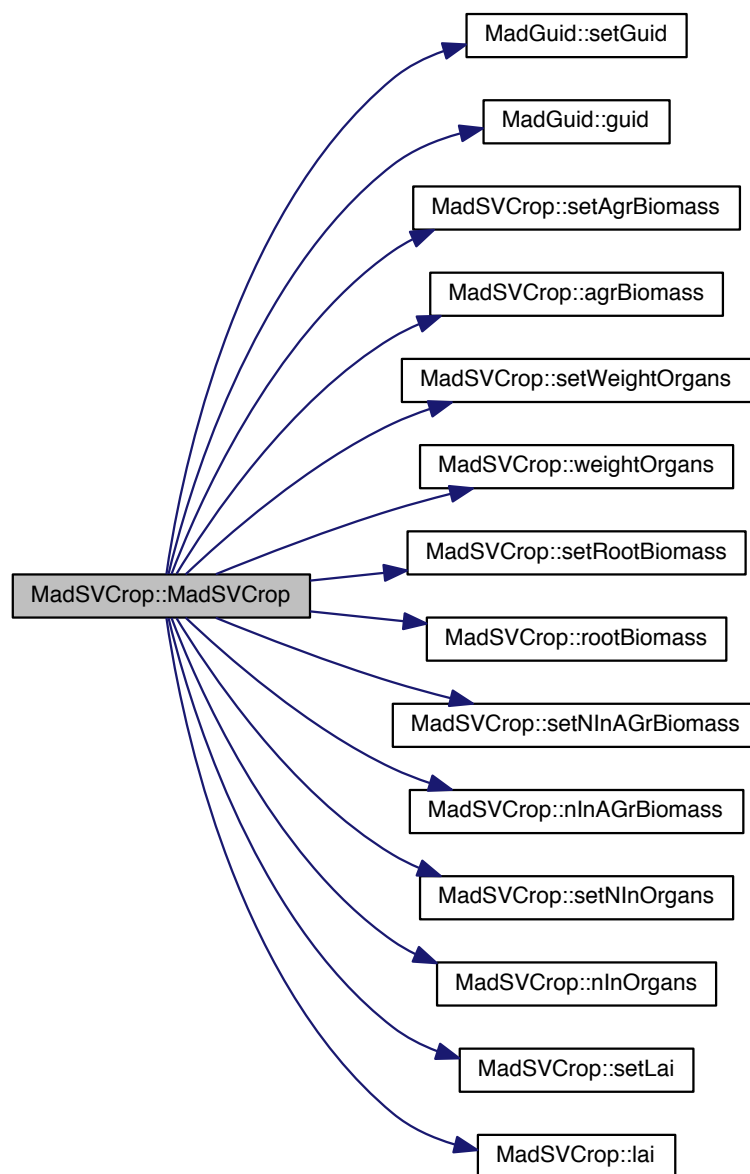


## 6.17.2.2 MadSVCrop::MadSVCrop ( const MadSVCrop &amp; theData )

Definition at line 38 of file madsvcrop.cpp.

```
39 {  
40   setGuid(theData.guid());  
41   setAgrBiomass(theData.agrBiomass());  
42   setWeightOrgans(theData.weightOrgans());  
43   setRootBiomass(theData.rootBiomass());  
44   setNInAgrBiomass(theData.nInAgrBiomass());  
45   setNInOrgans(theData.nInOrgans());  
46   setLai(theData.lai());  
47 }
```

Here is the call graph for this function:



### 6.17.3 Member Function Documentation

#### 6.17.3.1 MadSubCategory MadSVCrop::agrBiomass ( ) const

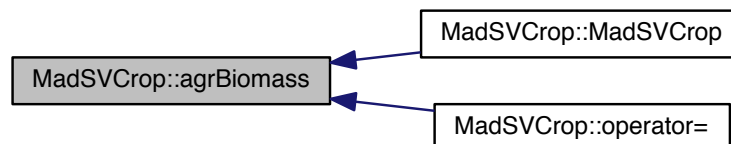
agrBiomass

Returns

Definition at line 64 of file madsvcrop.cpp.

```
65 {
66     return mAgrBiomass;
67 }
```

Here is the caller graph for this function:



#### 6.17.3.2 bool MadSVCrop::fromXml ( const QString theXml ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 125 of file madsvcrop.cpp.

```
126 {
127     QDomDocument myDocument("mydocument");
128     myDocument.setContent(theXml);
129     QDomElement myTopElement = myDocument.firstChildElement("svcrop");
130     if (myTopElement.isNull())
131     {
132         //TODO - just make this a warning
133         qDebug("the top element couldn't be found!");
134         setGuid(myTopElement.attribute("guid"));
135         //QDomElement myCategory;
136         //QDomElement myDetails;
137
138         //myCategory=QString(myTopElement.firstChildElement("agrbiomass").text());
139         //myDetails=QString(myCategory.firstChildElement("details").text());
140         //mWeightOrgans.depth()=QString(myDetails.firstChildElement("weightorgans").text()).toFloat();
141     }
```

```

142         //MadSubCategory mySVCropDetails;
143         //mySVCropDetails = QString(myTopElement.firstChildElement("details").text());
144
145         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
146         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
147         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
148         return true;
149     }
150     else
151         return false;
152 }

```

Here is the call graph for this function:



#### 6.17.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

##### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

##### Parameters

<i>theFileName</i>	
--------------------	--

##### Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }

```

Here is the call graph for this function:



#### 6.17.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

#### 6.17.3.5 MadSubCategory MadSVCrop::lai ( ) const

`lai`

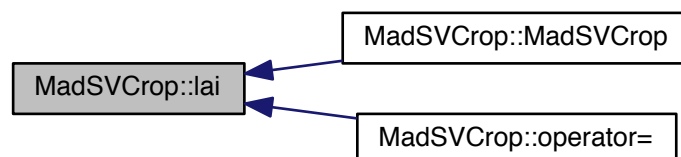
**Returns**

Definition at line 88 of file madsvcrop.cpp.

```

89 {
90     return mLai;
91 }
```

Here is the caller graph for this function:



### 6.17.3.6 MadSubCategory MadSVCrop::nInAGrBiomass ( ) const

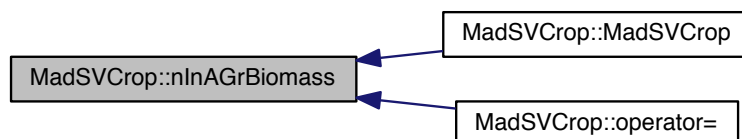
nInAGrBiomass

Returns

Definition at line 78 of file madsvcrop.cpp.

```
79 {  
80     return mNInAGrBiomass;  
81 }
```

Here is the caller graph for this function:



### 6.17.3.7 MadSubCategory MadSVCrop::nInOrgans ( ) const

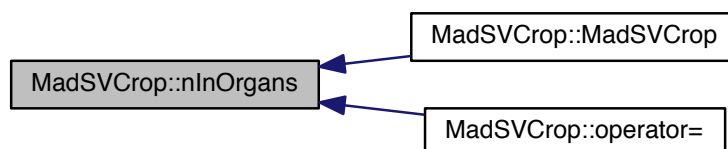
nInOrgans

Returns

Definition at line 83 of file madsvcrop.cpp.

```
84 {  
85     return mNInOrgans;  
86 }
```

Here is the caller graph for this function:



### 6.17.3.8 MadSVCrop & MadSVCrop::operator= ( const MadSVCrop & theData )

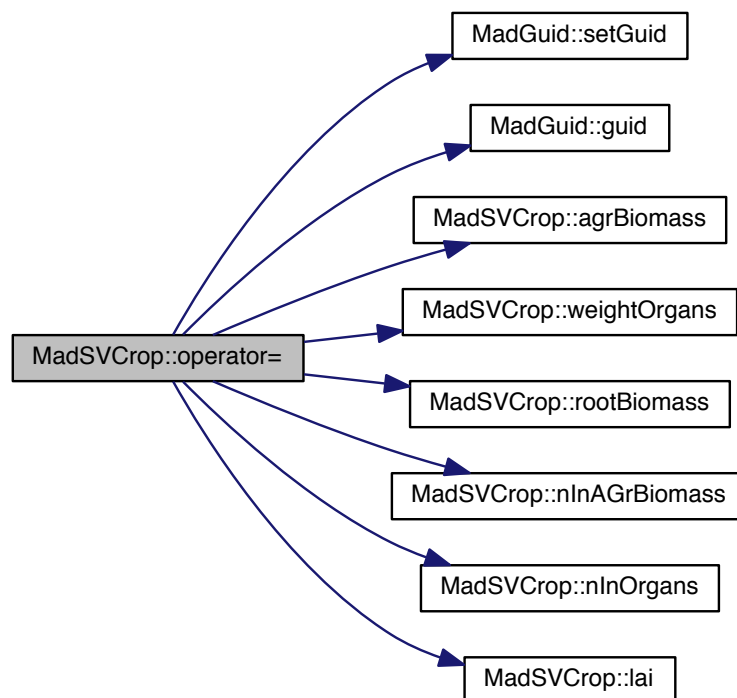
Definition at line 49 of file madsvcrop.cpp.

```

50 {
51     // gracefully handles self assignment
52     if (this == &theData) return *this;
53     setGuid(theData.guid());
54     mAgrBiomass=theData.agrBiomass();
55     mWeightOrgans=theData.weightOrgans();
56     mRootBiomass=theData.rootBiomass();
57     mNInAGrBiomass=theData.nInAGrBiomass();
58     mNInOrgans=theData.nInOrgans();
59     mLai=theData.lai();
60     return *this;
61 }

```

Here is the call graph for this function:



### 6.17.3.9 MadSubCategory MadSVCrop::rootBiomass ( ) const

`rootBiomass`

**Returns**

Definition at line 73 of file madsvcrop.cpp.

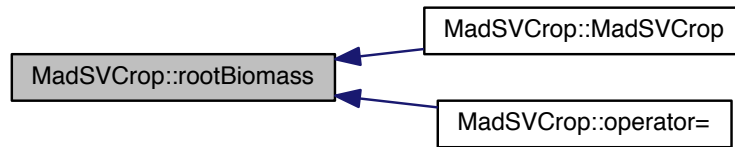
```

74 {
75     return mRootBiomass;
76 }

```



Here is the caller graph for this function:



#### 6.17.3.10 void MadSVCrop::setAgrBiomass ( MadSubCategory *theData* )

setAgrBiomass

##### Parameters

<i>theData</i>	
----------------	--

Definition at line 94 of file `madvscrop.cpp`.

```

95 {
96     mAgrBiomass = theData;
97 }
```

Here is the caller graph for this function:



#### 6.17.3.11 void MadGuid::setGuid ( QString *theGuid* = "" ) [inherited]

[MadGuid::setGuid](#).

##### Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file `madguid.cpp`.

```

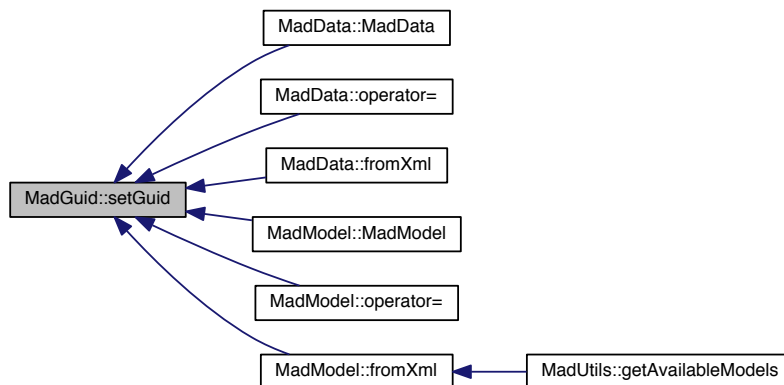
50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("}","");
54     }
55     else
56     {
57         mGuid=theGuid;
```

```

58     }
59 }

```

Here is the caller graph for this function:



#### 6.17.3.12 void MadSVCrop::setLai ( MadSubCategory *theData* )

`setLai`

Parameters

<i>theData</i>	
----------------	--

Definition at line 119 of file `madsvcrop.cpp`.

```

120 {
121     mLai = theData;
122 }

```

Here is the caller graph for this function:



#### 6.17.3.13 void MadSVCrop::setNInAGrBiomass ( MadSubCategory *theData* )

`setNInAGrBiomass`

Parameters

<i>theData</i>	
----------------	--

Definition at line 109 of file madsvcrop.cpp.

```
110 {
111     mNInAGrBiomass = theData;
112 }
```

Here is the caller graph for this function:



#### 6.17.3.14 void MadSVCrop::setNInOrgans ( MadSubCategory *theData* )

setNInOrgans

##### Parameters

<i>theData</i>	
----------------	--

Definition at line 114 of file madsvcrop.cpp.

```
115 {
116     mNInOrgans = theData;
117 }
```

Here is the caller graph for this function:



#### 6.17.3.15 void MadSVCrop::setRootBiomass ( MadSubCategory *theData* )

setRootBiomass

##### Parameters

<i>theData</i>	
----------------	--

Definition at line 104 of file madsvcrop.cpp.

```
105 {
106     mRootBiomass = theData;
107 }
```

Here is the caller graph for this function:



#### 6.17.3.16 void MadSVCrop::setWeightOrgans ( MadSubCategory *theData* )

setWeightOrgans

##### Parameters

<i>theData</i>	
----------------	--

Definition at line 99 of file madsvcrop.cpp.

```

100 {
101     mWeightOrgans = theData;
102 }
```

Here is the caller graph for this function:



#### 6.17.3.17 QString MadSVCrop::toHtml ( )

Return a html text representation of this layer

Definition at line 198 of file madsvcrop.cpp.

```

199 {
200     QString myString;
201     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
202     //myString+="<p>GUID: " + guid() + "</p>";
203     myString+="<table>";
204     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
205
206     //
207     // the following shows example of how to do a couple of things
208     //
209
210     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
211     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
212     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
213     //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
214     //    QString::number(mCropFodderProduction) + "</td></tr>";
214     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
215     //    "</td></tr>";
215     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
216     //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
  
```

```

217   myString+="/table>";
218   return myString;
219 }

```

### 6.17.3.18 QString MadSVCrop::toText ( )

Return a plain text representation of this layer

Definition at line 189 of file madsvcrop.cpp.

```

190 {
191   QString myString;
192   myString+=QString("guid=>" + guid() + "\n");
193   //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
194   //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
195   return myString;
196 }

```

Here is the call graph for this function:



### 6.17.3.19 QString MadSVCrop::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 154 of file madsvcrop.cpp.

```

155 {
156   QString myString;
157   myString+=QString("    <svcrop guid=\" + guid() + "\">\n");
158
159   myString+=QString("    <agrbiomass>\n");
160   myString+=mAgrBiomass.toXml();
161   myString+=QString("    </agrbiomass>\n");
162
163   myString+=QString("    <weightorgans>\n");
164   myString+=mWeightOrgans.toXml();
165   myString+=QString("    </weightorgans>\n");
166
167   myString+=QString("    <rootbiomass>\n");
168   myString+=mRootBiomass.toXml();
169   myString+=QString("    </rootbiomass>\n");
170
171   myString+=QString("    <ninagrbiomass>\n");
172   myString+=mNInAgrBiomass.toXml();
173   myString+=QString("    </ninagrbiomass>\n");
174
175   myString+=QString("    <ninorgans>\n");
176   myString+=mNInOrgans.toXml();
177   myString+=QString("    </ninorgans>\n");
178 }

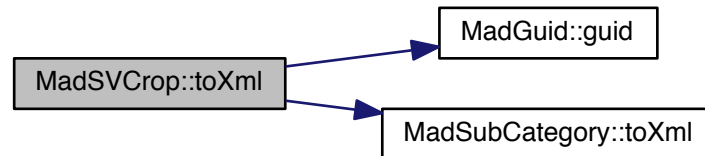
```

```

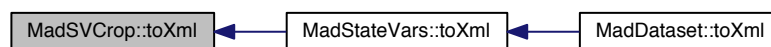
179 myString+=QString("    <lai>\n");
180 myString+=mLai.toXml();
181 myString+=QString("    </lai>\n");
182
183 myString+=QString("    </svcrop>\n");
184 return myString;
185
186
187 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.17.3.20 bool MadSerialisable::toXmlFile ( const QString *theFileName* ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

#### See Also

[toXml\(\)](#)

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );

```

```

61  if ( myFile.open( QIODevice::WriteOnly ) )
62  {
63      QTextStream myQTextStream( &myFile );
64      myQTextStream << this->toXml();
65      myFile.close();
66      myResult=true;
67  }
68  else
69  {
70      //@TODO Error handler!
71      myResult=false;
72  }
73  return myResult ;
74 }

```

Here is the call graph for this function:



#### 6.17.3.21 MadSubCategory MadSVCrop::weightOrgans ( ) const

weightOrgans

Returns

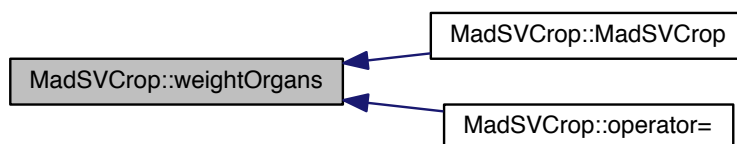
Definition at line 68 of file madsvcrop.cpp.

```

69 {
70     return mWeightOrgans;
71 }

```

Here is the caller graph for this function:



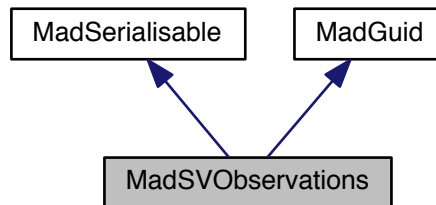
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/[madsvcrop.h](#)
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/[madsvcrop.cpp](#)

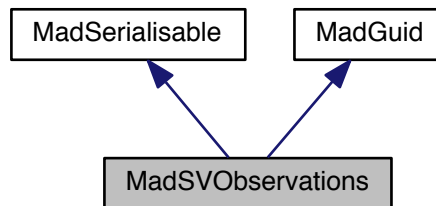
## 6.18 MadSVObservations Class Reference

```
#include <madsvobservations.h>
```

Inheritance diagram for MadSVObservations:



Collaboration diagram for MadSVObservations:



### Public Member Functions

- [MadSVObservations](#) ()
- [MadSVObservations](#) (const [MadSVObservations](#) &theData)
- [MadSVObservations](#) & operator= (const [MadSVObservations](#) &theData)
- [MadSubCategory lodging](#) () const
- [MadSubCategory pestsOrDiseases](#) () const
- [MadSubCategory damages](#) () const
- [QString toXml](#) ()
- [QString toText](#) ()
- [QString toHtml](#) ()
- [bool fromXml](#) (const [QString](#) theXml)
- [void setLodging](#) ([MadSubCategory](#) theData)
- [void setPestsOrDiseases](#) ([MadSubCategory](#) theData)
- [void setDamages](#) ([MadSubCategory](#) theData)
- [virtual bool toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile* writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.



- virtual bool [fromXmlFile](#) (const QString theFileName)

*fromXmlFile* Read this object from xml in a file

- QString [guid](#) () const

*MadGuid::guid.*

- void [setGuid](#) (QString theGuid="")

*MadGuid::setGuid.*

### 6.18.1 Detailed Description

Definition at line 35 of file madsvobservations.h.

### 6.18.2 Constructor & Destructor Documentation

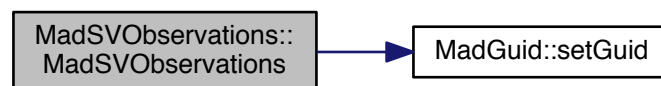
#### 6.18.2.1 MadSVObservations::MadSVObservations ( )

Definition at line 33 of file madsvobservations.cpp.

```

33                                     : MadSerialisable(), MadGuid()
34 {
35     setGuid();
36 }
```

Here is the call graph for this function:



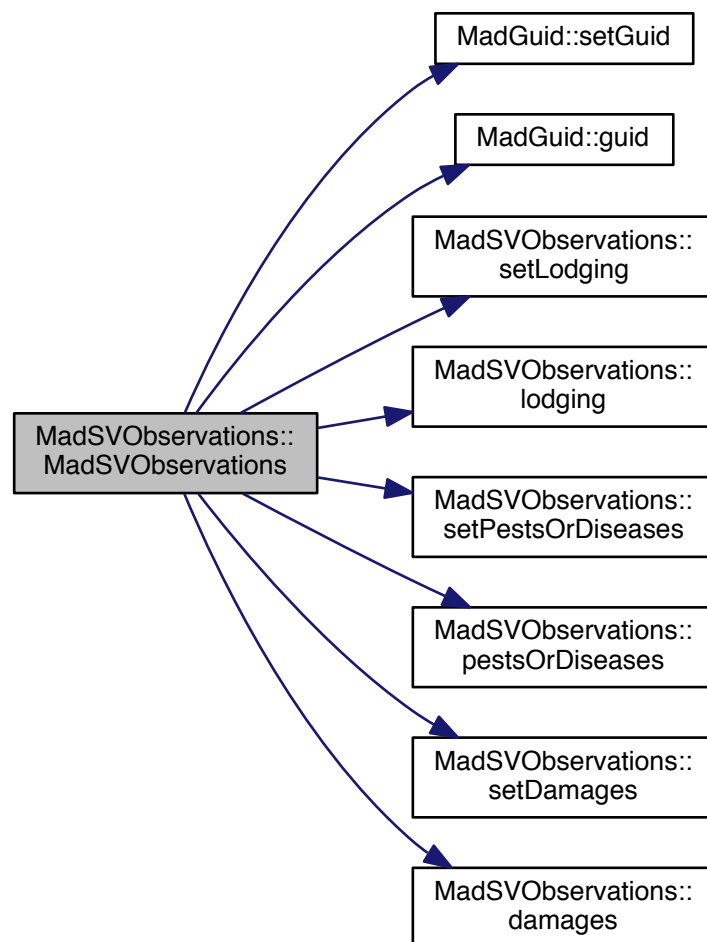
#### 6.18.2.2 MadSVObservations::MadSVObservations ( const MadSVObservations & theData )

Definition at line 38 of file madsvobservations.cpp.

```

39 {
40     setGuid(theData.guid());
41     setLodging(theData.lodging());
42     setPestsOrDiseases(theData.pestsOrDiseases());
43     setDamages(theData.damages());
44 }
```

Here is the call graph for this function:



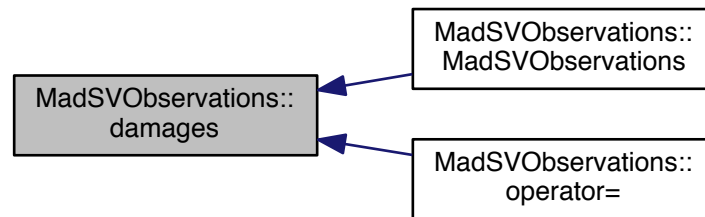
## 6.18.3 Member Function Documentation

### 6.18.3.1 MadSubCategory MadSVObservations::damages ( ) const

Definition at line 66 of file `madvsoobservations.cpp`.

```
67 {  
68     return mDamages;  
69 }
```

Here is the caller graph for this function:



### 6.18.3.2 bool MadSVObservations::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

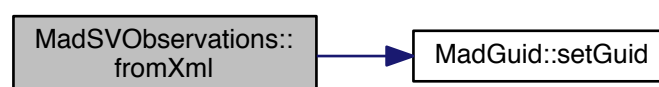
Implements [MadSerialisable](#).

Definition at line 87 of file madsvobservations.cpp.

```

88 {
89     QDomDocument myDocument("mydocument");
90     myDocument.setContent(theXml);
91     QDomElement myTopElement = myDocument.firstChildElement("svobservations");
92     if (myTopElement.isNull())
93     {
94         //TODO - just make this a warning
95         qDebug("the top element couldn't be found!");
96         setGuid(myTopElement.attribute("guid"));
97         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
98         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
99         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
100         return true;
101     }
102     else
103         return false;
104 }
  
```

Here is the call graph for this function:



### 6.18.3.3 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

#### See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

#### Parameters

<i>theFileName</i>	
--------------------	--

#### Returns

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



### 6.18.3.4 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid](#).

Destructor Retrieve the GUID

#### Returns

Definition at line 40 of file madguid.cpp.

```

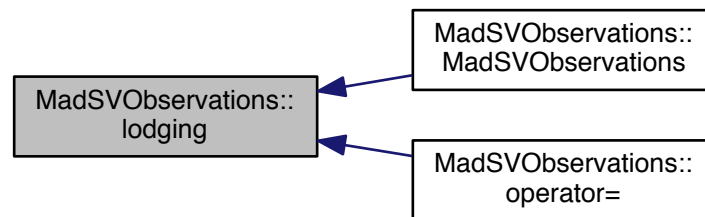
41 {
42     return mGuid;
43 }
```

### 6.18.3.5 MadSubCategory MadSVObservations::lodging ( ) const

Definition at line 58 of file madsvobservations.cpp.

```
59 {  
60     return mLodging;  
61 }
```

Here is the caller graph for this function:

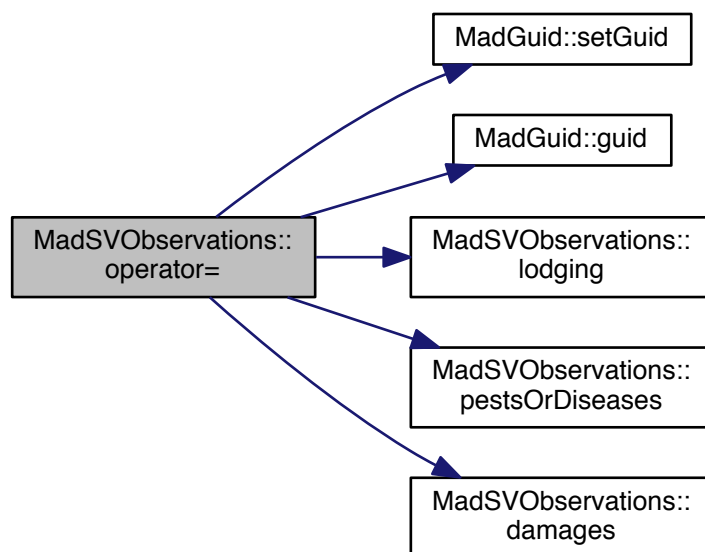


### 6.18.3.6 MadSVObservations & MadSVObservations::operator= ( const MadSVObservations & *theData* )

Definition at line 46 of file madsvobservations.cpp.

```
47 {  
48     // gracefully handles self assignment  
49     if (this == &theData) return *this;  
50     setGuid(theData.guid());  
51     mLodging=theData.lodging();  
52     mPestsOrDiseases=theData.pestsOrDiseases();  
53     mDamages=theData.damages();  
54     return *this;  
55 }
```

Here is the call graph for this function:



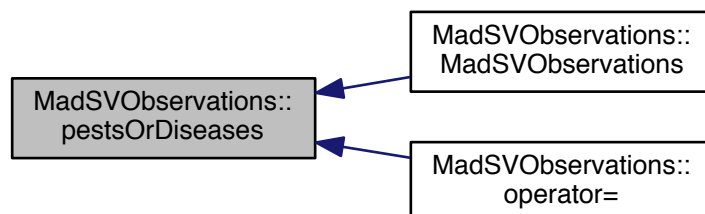
#### 6.18.3.7 MadSubCategory MadSVObservations::pestsOrDiseases ( ) const

Definition at line 62 of file `madvsoobservations.cpp`.

```

63 {
64     return mPestsOrDiseases;
65 }
  
```

Here is the caller graph for this function:



#### 6.18.3.8 void MadSVObservations::setDamages ( MadSubCategory theData )

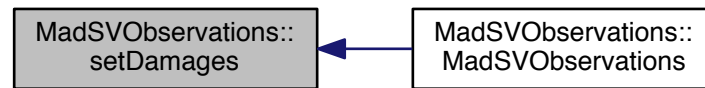
Definition at line 82 of file `madvsoobservations.cpp`.

```

83 {
84     mDamages = theData;
85 }

```

Here is the caller graph for this function:



#### 6.18.3.9 void MadGuid::setGuid ( QString *theGuid* = " " ) [inherited]

[MadGuid::setGuid.](#)

##### Parameters

<i>theGuid</i>	
----------------	--

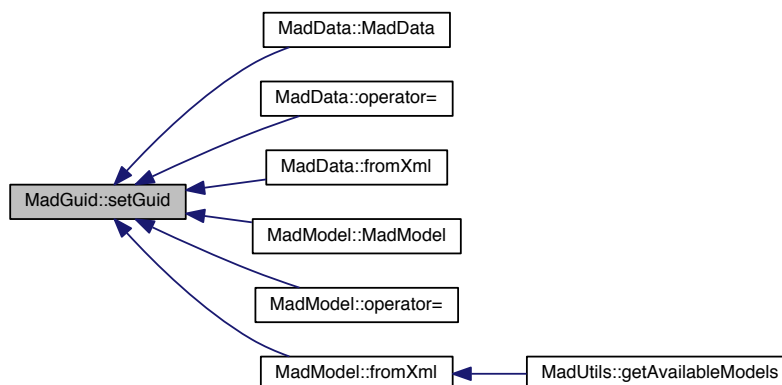
Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("}","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:

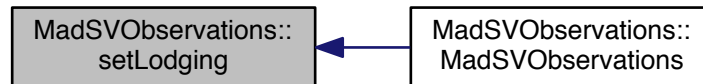


#### 6.18.3.10 void MadSVObservations::setLodging ( MadSubCategory *theData* )

Definition at line 72 of file madsvobservations.cpp.

```
73 {
74     mLodging = theData;
75 }
```

Here is the caller graph for this function:

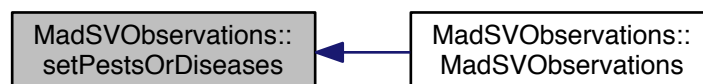


#### 6.18.3.11 void MadSVObservations::setPestsOrDiseases ( MadSubCategory *theData* )

Definition at line 77 of file madsvobservations.cpp.

```
78 {
79     mPestsOrDiseases = theData;
80 }
```

Here is the caller graph for this function:



#### 6.18.3.12 QString MadSVObservations::toHtml ( )

Return a html text representation of this layer

Definition at line 137 of file madsvobservations.cpp.

```
138 {
139     QString myString;
140     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
141     //myString+="<p>GUID: " + guid() + "</p>";
142     myString+="<table>";
143     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
144
145     //
146     // the following shows example of how to do a couple of things
147     //
148
149     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
```



```

150 //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
151 //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
152 //myString+="<tr><td><b>Fodder (kg/" + myUnits + "): </b></td><td>" +
    QString::number(mCropFodderProduction) + "</td></tr>";
153 //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
    "</td></tr>";
154 //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
155 //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
156 myString+="</table>";
157 return myString;
158 }

```

### 6.18.3.13 QString MadSVObservations::toText ( )

Return a plain text representation of this layer

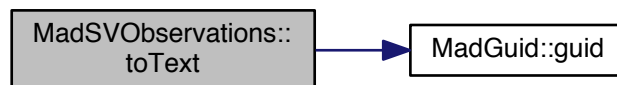
Definition at line 128 of file madsvobservations.cpp.

```

129 {
130     QString myString;
131     myString+=QString("guid=>" + guid() + "\n");
132     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
133     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
134     return myString;
135 }

```

Here is the call graph for this function:



### 6.18.3.14 QString MadSVObservations::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 106 of file madsvobservations.cpp.

```

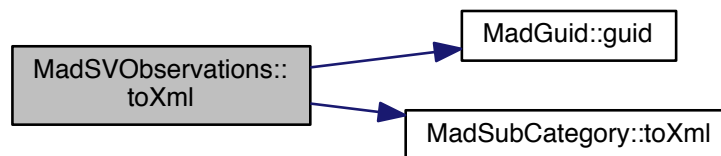
107 {
108     QString myString;
109     myString+=QString("    <svobservations guid=\"\" + guid() + "\">\n");
110
111     myString+=QString("        <lodging>\n");
112     myString+=mLodging.toXml();
113     myString+=QString("        </lodging>\n");
114
115     myString+=QString("        <pestsordiseases>\n");
116     myString+=mPestsOrDiseases.toXml();
117     myString+=QString("        </pestsordiseases>\n");
118
119     myString+=QString("        <damage>\n");
120     myString+=mDamages.toXml();
121     myString+=QString("        </damage>\n");

```

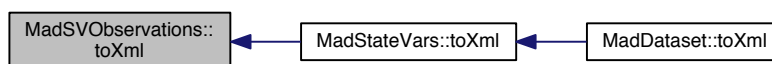
```

122
123   myString+=QString("    </svobservations>\n");
124   return myString;
125
126 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



**6.18.3.15** `bool MadSerialisable::toXmlFile ( const QString theFileName )` [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual QString) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

See Also

[toXml\(\)](#)

Parameters

<i>theFileName</i>	
--------------------	--

Returns

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59   bool myResult = false;
60   QFile myFile( theFileName );
61   if ( myFile.open( QIODevice::WriteOnly ) )
62   {
63     QTextStream myQTextStream( &myFile );
64     myQTextStream << this->toXml();
```

```
65     myFile.close();
66     myResult=true;
67 }
68 else
69 {
70     //@TODO Error handler!
71     myResult=false;
72 }
73 return myResult ;
74 }
```

Here is the call graph for this function:



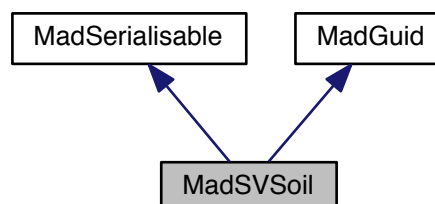
The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvobservations.cpp](#)

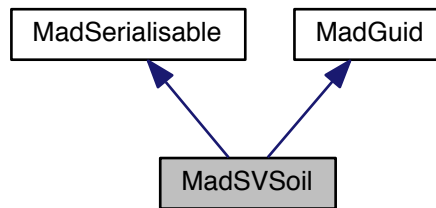
## 6.19 MadSVSoil Class Reference

```
#include <madsvsoil.h>
```

Inheritance diagram for MadSVSoil:



Collaboration diagram for MadSVSoil:



## Public Member Functions

- [MadSVSoil](#) ()
- [MadSVSoil](#) (const [MadSVSoil](#) &theData)
- [MadSVSoil](#) & [operator=](#) (const [MadSVSoil](#) &theData)
- [MadSubCategory](#) [soilWaterGrav](#) () const
- [MadSubCategory](#) [pressureHeads](#) () const
- [MadSubCategory](#) [nMin](#) () const
- [MadSubCategory](#) [soilWaterSensorCal](#) () const
- [MadSubCategory](#) [waterFluxBottomRoot](#) () const
- [MadSubCategory](#) [nitrogenFluxBottomRoot](#) () const
- [QString](#) [toXml](#) ()
- [QString](#) [toText](#) ()
- [QString](#) [toHtml](#) ()
- [bool](#) [fromXml](#) (const [QString](#) theXml)
- [void](#) [setSoilWaterGrav](#) ([MadSubCategory](#) theData)
- [void](#) [setPressureHeads](#) ([MadSubCategory](#) theData)
- [void](#) [setNMin](#) ([MadSubCategory](#) theData)
- [void](#) [setSoilWaterSensorCal](#) ([MadSubCategory](#) theData)
- [void](#) [setWaterFluxBottomRoot](#) ([MadSubCategory](#) theData)
- [void](#) [setNitrogenFluxBottomRoot](#) ([MadSubCategory](#) theData)
- [virtual bool](#) [toXmlFile](#) (const [QString](#) theFileName)

*toXmlFile writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.*

- [virtual bool](#) [fromXmlFile](#) (const [QString](#) theFileName)
- *fromXmlFile Read this object from xml in a file*
- [QString](#) [guid](#) () const
- *[MadGuid::guid](#).*
- [void](#) [setGuid](#) ([QString](#) theGuid="")
- *[MadGuid::setGuid](#).*

### 6.19.1 Detailed Description

Definition at line 35 of file `madsvsoil.h`.

## 6.19.2 Constructor & Destructor Documentation

### 6.19.2.1 MadSVSoil::MadSVSoil ( )

Definition at line 34 of file madsvsoil.cpp.

```
34         : MadSerialisable(), MadGuid()
35 {
36     setGuid();
37 }
```

Here is the call graph for this function:

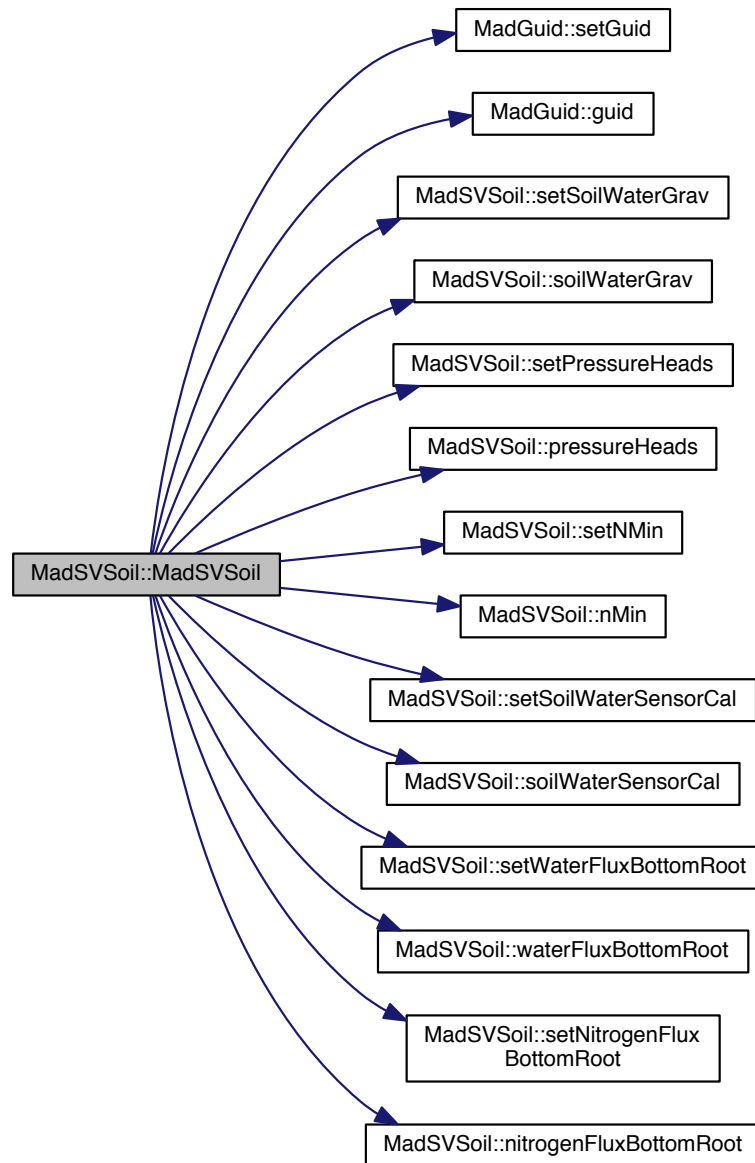


### 6.19.2.2 MadSVSoil::MadSVSoil ( const MadSVSoil & *theData* )

Definition at line 39 of file madsvsoil.cpp.

```
40 {
41     setGuid(theData.guid());
42     setSoilWaterGrav(theData.soilWaterGrav());
43     setPressureHeads(theData.pressureHeads());
44     setNMin(theData.nMin());
45     setSoilWaterSensorCal(theData.soilWaterSensorCal());
46     setWaterFluxBottomRoot(theData.waterFluxBottomRoot());
47     setNitrogenFluxBottomRoot(theData.
48         nitrogenFluxBottomRoot());
48 }
```

Here is the call graph for this function:



### 6.19.3 Member Function Documentation

#### 6.19.3.1 `bool MadSVSoil::fromXml ( const QString theXml ) [virtual]`

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

**Note**

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 127 of file madsvsoil.cpp.

```

128 {
129     QDomDocument myDocument("mydocument");
130     myDocument.setContent(theXml);
131     QDomElement myTopElement = myDocument.firstChildElement("svsoil");
132     if (myTopElement.isNull())
133     {
134         //TODO - just make this a warning
135         qDebug("the top element couldn't be found!");
136         setGuid(myTopElement.attribute("guid"));
137         //mName=MadUtils::xmlDecode(myTopElement.firstChildElement("name").text());
138         //mDescription=MadUtils::xmlDecode(myTopElement.firstChildElement("description").text());
139         //mImageFile=QString(myTopElement.firstChildElement("imageFile").text());
140         return true;
141     }
142     else
143         return false;
144 }
```

Here is the call graph for this function:



### 6.19.3.2 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual],[inherited]

fromXmlFile Read this object from xml in a file

**See Also**

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

**Parameters**

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
```

```

86 {
87     //@TODO Error handler!
88     myResult=false;
89 }
90 return myResult ;
91 }

```

Here is the call graph for this function:



#### 6.19.3.3 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

##### Returns

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }

```

#### 6.19.3.4 MadSubCategory MadSVSoil::nitrogenFluxBottomRoot ( ) const

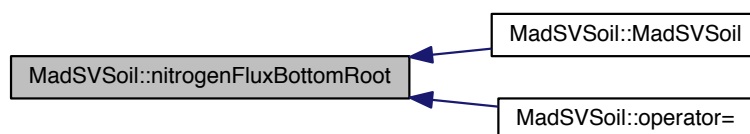
Definition at line 91 of file madsvsoil.cpp.

```

92 {
93     return mNitrogenFluxBottomRoot;
94 }

```

Here is the caller graph for this function:



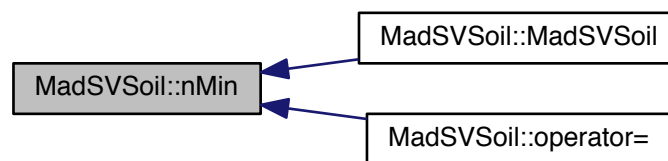


### 6.19.3.5 MadSubCategory MadSVSoil::nMin ( ) const

Definition at line 76 of file madsvsoil.cpp.

```
77 {
78     return mNMin;
79 }
```

Here is the caller graph for this function:

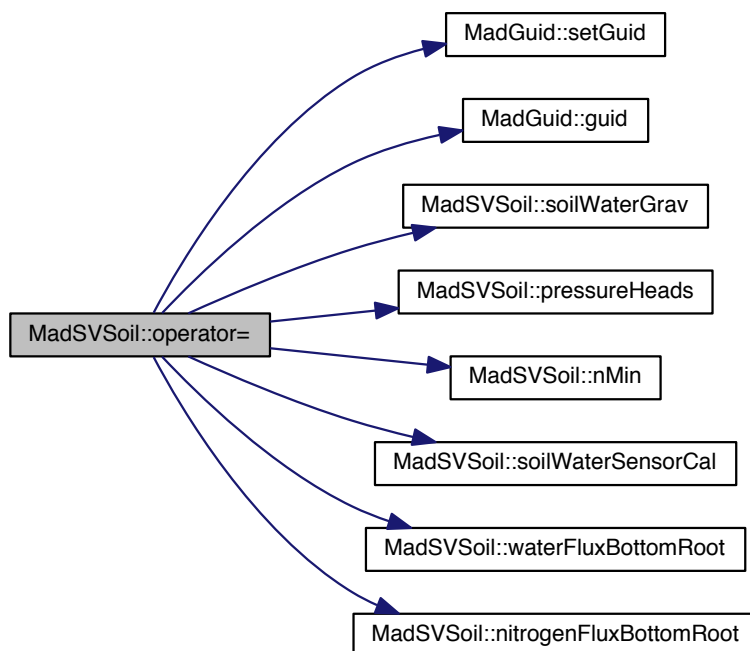


### 6.19.3.6 MadSVSoil & MadSVSoil::operator= ( const MadSVSoil & *theData* )

Definition at line 50 of file madsvsoil.cpp.

```
51 {
52     // gracefully handles self assignment
53     if (this == &theData) return *this;
54     setGuid(theData.guid());
55     mSoilWaterGrav=theData.soilWaterGrav();
56     mPressureHeads=theData.pressureHeads();
57     mNMin=theData.nMin();
58     mSoilWaterSensorCal=theData.soilWaterSensorCal();
59     mWaterFluxBottomRoot=theData.waterFluxBottomRoot();
60     mNitrogenFluxBottomRoot=theData.nitrogenFluxBottomRoot();
61     return *this;
62 }
```

Here is the call graph for this function:



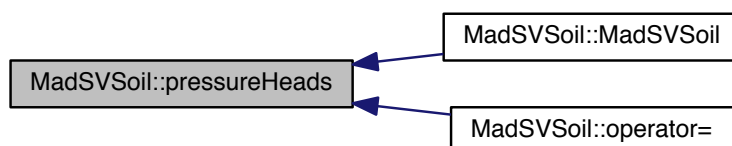
#### 6.19.3.7 MadSubCategory MadSVSoil::pressureHeads ( ) const

Definition at line 71 of file `madsvsoil.cpp`.

```

72 {
73     return mPressureHeads;
74 }
  
```

Here is the caller graph for this function:



#### 6.19.3.8 void MadGuid::setGuid ( QString theGuid = " " ) [inherited]

[MadGuid::setGuid.](#)

## Parameters

<i>theGuid</i>	
----------------	--

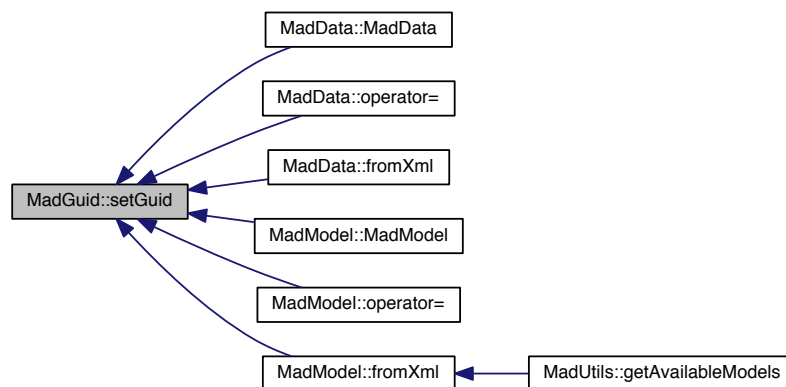
Definition at line 49 of file madguid.cpp.

```

50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("}","");
54     }
55     else
56     {
57         mGuid=theGuid;
58     }
59 }

```

Here is the caller graph for this function:



### 6.19.3.9 void MadSVSoil::setNitrogenFluxBottomRoot ( MadSubCategory *theData* )

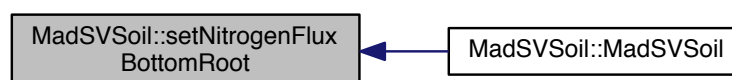
Definition at line 122 of file madsvsoil.cpp.

```

123 {
124     mNitrogenFluxBottomRoot = theData;
125 }

```

Here is the caller graph for this function:



**6.19.3.10 void MadSVSoil::setNMin ( MadSubCategory *theData* )**

Definition at line 107 of file madsvsoil.cpp.

```
108 {  
109     mNMin = theData;  
110 }
```

Here is the caller graph for this function:

**6.19.3.11 void MadSVSoil::setPressureHeads ( MadSubCategory *theData* )**

Definition at line 102 of file madsvsoil.cpp.

```
103 {  
104     mPressureHeads = theData;  
105 }
```

Here is the caller graph for this function:

**6.19.3.12 void MadSVSoil::setSoilWaterGrav ( MadSubCategory *theData* )**

Definition at line 97 of file madsvsoil.cpp.

```
98 {  
99     mSoilWaterGrav = theData;  
100 }
```

Here is the caller graph for this function:

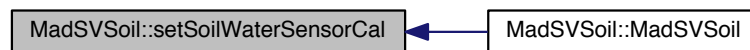


**6.19.3.13 void MadSVSoil::setSoilWaterSensorCal ( MadSubCategory *theData* )**

Definition at line 112 of file madsvsoil.cpp.

```
113 {  
114     mSoilWaterSensorCal = theData;  
115 }
```

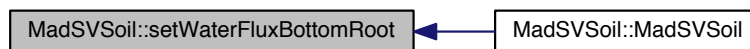
Here is the caller graph for this function:

**6.19.3.14 void MadSVSoil::setWaterFluxBottomRoot ( MadSubCategory *theData* )**

Definition at line 117 of file madsvsoil.cpp.

```
118 {  
119     mWaterFluxBottomRoot = theData;  
120 }
```

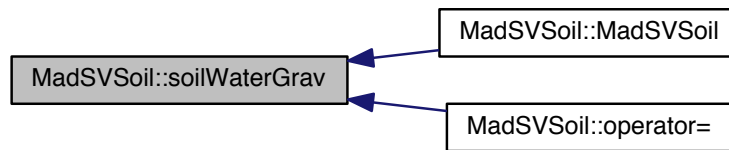
Here is the caller graph for this function:

**6.19.3.15 MadSubCategory MadSVSoil::soilWaterGrav ( ) const**

Definition at line 66 of file madsvsoil.cpp.

```
67 {  
68     return mSoilWaterGrav;  
69 }
```

Here is the caller graph for this function:



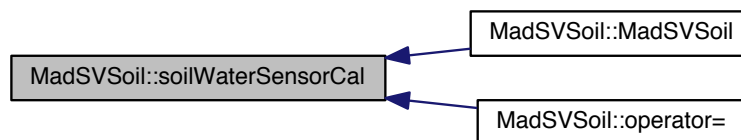
#### 6.19.3.16 MadSubCategory MadSVSoil::soilWaterSensorCal ( ) const

Definition at line 81 of file madsvsoil.cpp.

```

82 {
83     return mSoilWaterSensorCal;
84 }
  
```

Here is the caller graph for this function:



#### 6.19.3.17 QString MadSVSoil::toHtml ( )

Return a html text representation of this layer

Definition at line 190 of file madsvsoil.cpp.

```

191 {
192     QString myString;
193     //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
194     //myString+="<p>GUID: " + guid() + "</p>";
195     myString+="<table>";
196     //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
197
198     //
199     // the following shows example of how to do a couple of things
200     //
201
202     //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
203     //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
204     //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
205     //myString+="<tr><td><b>Fodder (kg/" + myUnits + ") : </b></td><td>" +
206     //    QString::number(mCropFodderProduction) + "</td></tr>";
207     //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
208     //    "</td></tr>";
209     //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
  
```

```

208 //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
209 myString+="</table>";
210 return myString;
211 }

```

#### 6.19.3.18 QString MadSVSoil::toText ( )

Return a plain text representation of this layer

Definition at line 181 of file madsvsoil.cpp.

```

182 {
183     QString myString;
184     myString+=QString("guid=>" + guid() + "\n");
185     //myString+=QString("name=>" + MadUtils::xmlEncode(mName) + "\n");
186     //myString+=QString("description=>" + MadUtils::xmlEncode(mDescription) + "\n");
187     return myString;
188 }

```

Here is the call graph for this function:



#### 6.19.3.19 QString MadSVSoil::toXml ( ) [virtual]

Return an xml representation of this layer

##### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 146 of file madsvsoil.cpp.

```

147 {
148     QString myString;
149     myString+=QString("    <svsoil guid=\"\" + guid() + "\">\n");
150
151     myString+=QString("        <soilwatergrav>\n");
152     myString+=mSoilWaterGrav.toXml();
153     myString+=QString("        </soilwatergrav>\n");
154
155     myString+=QString("        <pressureheads>\n");
156     myString+=mPressureHeads.toXml();
157     myString+=QString("        </pressureheads>\n");
158
159     myString+=QString("        <nmin>\n");
160     myString+=mNMin.toXml();
161     myString+=QString("        </nmin>\n");
162
163     myString+=QString("        <soilwatersensorcal>\n");
164     myString+=mSoilWaterSensorCal.toXml();
165     myString+=QString("        </soilwatersensorcal>\n");
166
167     myString+=QString("        <waterfluxbottomroot>\n");
168     myString+=mWaterFluxBottomRoot.toXml();
169     myString+=QString("        </waterfluxbottomroot>\n");

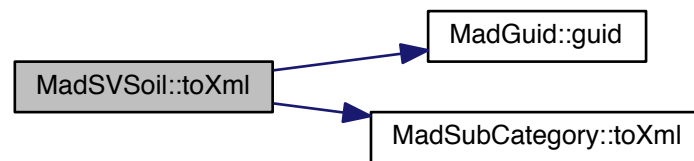
```

```

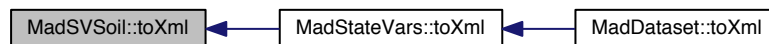
170
171 myString+=QString("    <nitrogenfluxbottomroot>\n");
172 myString+=mNitrogenFluxBottomRoot.toXml();
173 myString+=QString("    </nitrogenfluxbottomroot>\n");
174
175 myString+=QString("  </svsoil>\n");
176 return myString;
177
178
179 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



**6.19.3.20** `bool MadSerialisable::toXmlFile ( const QString theFileName )` `[virtual],[inherited]`

`toXmlFile` writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

**See Also**

[toXml\(\)](#)

**Parameters**

<i>theFileName</i>	
--------------------	--

**Returns**

QString (virtual)

Definition at line 57 of file `madserialisable.cpp`.

```

58 {
59     bool myResult = false;

```



```

60  QFile myFile( theFileName );
61  if ( myFile.open( QIODevice::WriteOnly ) )
62  {
63      QTextStream myQTextStream( &myFile );
64      myQTextStream << this->toXml();
65      myFile.close();
66      myResult=true;
67  }
68  else
69  {
70      //@TODO Error handler!
71      myResult=false;
72  }
73  return myResult ;
74 }

```

Here is the call graph for this function:



#### 6.19.3.21 MadSubCategory MadSVSoil::waterFluxBottomRoot ( ) const

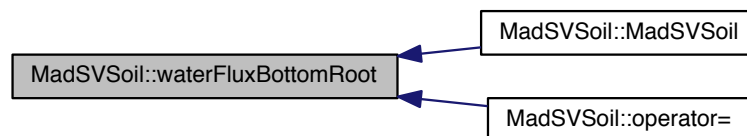
Definition at line 86 of file `madsvsoil.cpp`.

```

87 {
88     return mWaterFluxBottomRoot;
89 }

```

Here is the caller graph for this function:



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

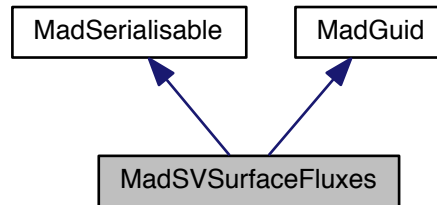
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.cpp](#)

## 6.20 MadSVSurfaceFluxes Class Reference

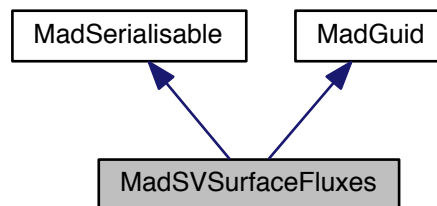
The [MadSVSurfaceFluxes](#) class.

```
#include <madsvsurfacefluxes.h>
```

Inheritance diagram for MadSVSurfaceFluxes:



Collaboration diagram for MadSVSurfaceFluxes:



## Public Member Functions

- [MadSVSurfaceFluxes](#) ()
- [MadSVSurfaceFluxes](#) (const [MadSVSurfaceFluxes](#) &theData)
- [MadSVSurfaceFluxes](#) & operator= (const [MadSVSurfaceFluxes](#) &theData)
- [MadSubCategory](#) et () const  
*et*
- [MadSubCategory](#) nh3Loss () const  
*nh3Loss*
- [MadSubCategory](#) n2oLoss () const  
*n2oLoss*
- [MadSubCategory](#) n2Loss () const  
*n2Loss*
- [MadSubCategory](#) ch4Loss () const  
*ch4Loss*
- QString [toXml](#) ()
- QString [toText](#) ()
- QString [toHtml](#) ()
- bool [fromXml](#) (const QString theXml)

- void [setEt](#) ([MadSubCategory](#) theData)  
*setEt*
- void [setNh3Loss](#) ([MadSubCategory](#) theData)  
*setNh3Loss*
- void [setN2oLoss](#) ([MadSubCategory](#) theData)  
*setN2oLoss*
- void [setN2Loss](#) ([MadSubCategory](#) theData)  
*setN2Loss*
- void [setCh4Loss](#) ([MadSubCategory](#) theData)  
*setCh4Loss*
- virtual bool [toXmlFile](#) (const QString theFileName)  
*toXmlFile* writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses [toXml\(\)](#) method so that must be properly implemented.
- virtual bool [fromXmlFile](#) (const QString theFileName)  
*fromXmlFile* Read this object from xml in a file
- QString [guid](#) () const  
*MadGuid::guid.*
- void [setGuid](#) (QString theGuid="")  
*MadGuid::setGuid.*

### 6.20.1 Detailed Description

The [MadSVSurfaceFluxes](#) class.

Definition at line 38 of file `madsvsurfacefluxes.h`.

### 6.20.2 Constructor & Destructor Documentation

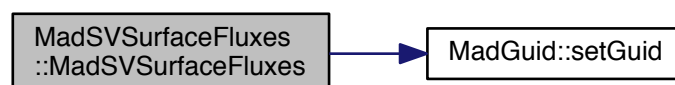
#### 6.20.2.1 MadSVSurfaceFluxes::MadSVSurfaceFluxes ( )

Definition at line 32 of file `madsvsurfacefluxes.cpp`.

```

32                                     : MadSerialisable() ,
33     MadGuid()
34 {
35     setGuid();
36     MadSubCategory mEt;
37     MadSubCategory mNh3Loss;
38     MadSubCategory mN2oLoss;
39     MadSubCategory mN2Loss;
40     MadSubCategory mCh4Loss;
41 }
```

Here is the call graph for this function:

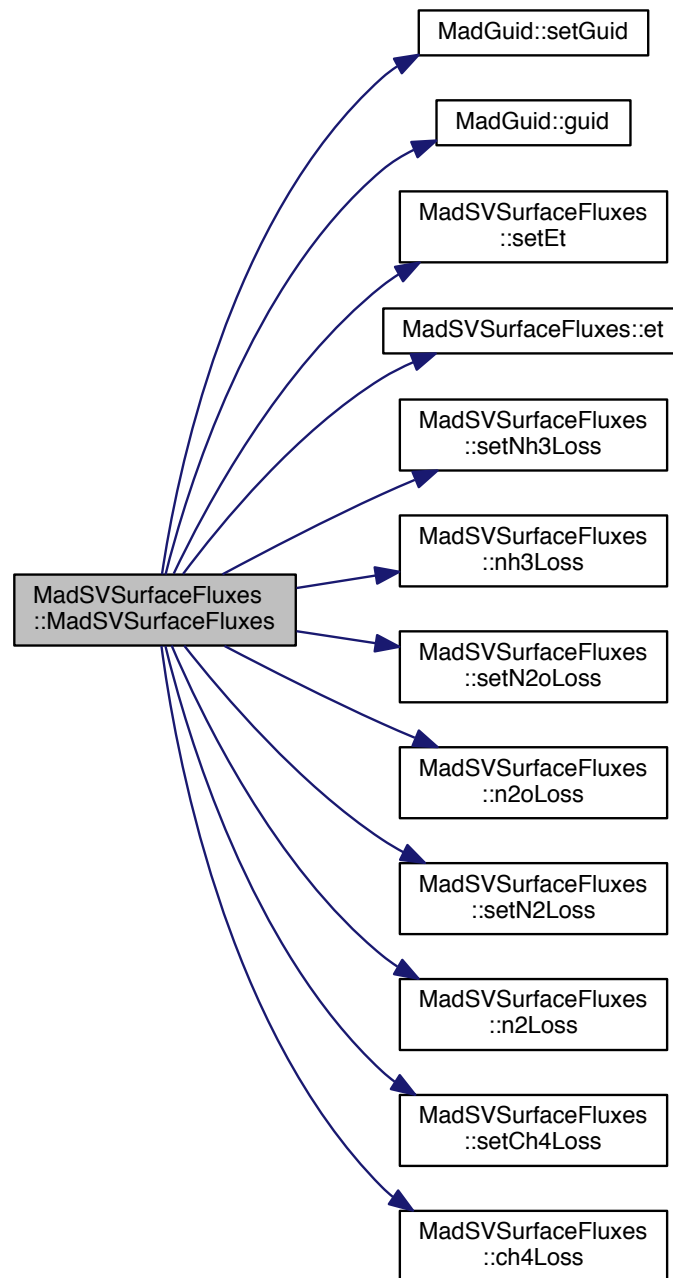


#### 6.20.2.2 MadSVSurfaceFluxes::MadSVSurfaceFluxes ( const MadSVSurfaceFluxes & *theData* )

Definition at line 43 of file madsvsurfacefluxes.cpp.

```
44 {  
45     setGuid(theData.guid());  
46     setEt(theData.et());  
47     setNh3Loss(theData.nh3Loss());  
48     setN2oLoss(theData.n2oLoss());  
49     setN2Loss(theData.n2Loss());  
50     setCh4Loss(theData.ch4Loss());  
51 }
```

Here is the call graph for this function:



### 6.20.3 Member Function Documentation

#### 6.20.3.1 MadSubCategory MadSVSurfaceFluxes::ch4Loss ( ) const

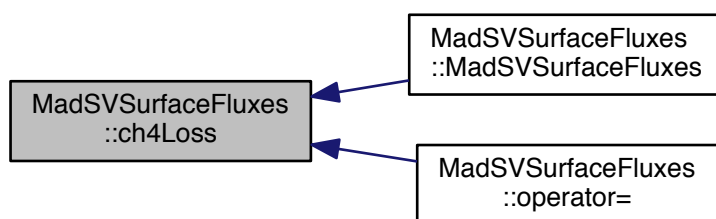
ch4Loss

## Returns

Definition at line 86 of file madsvsurfacefluxes.cpp.

```
87 {  
88     return mCh4Loss;  
89 }
```

Here is the caller graph for this function:



### 6.20.3.2 MadSubCategory MadSVSurfaceFluxes::et ( ) const

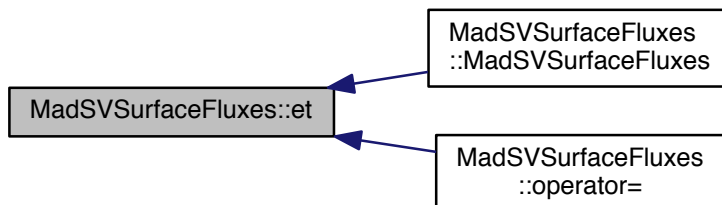
et

## Returns

Definition at line 67 of file madsvsurfacefluxes.cpp.

```
68 {  
69     return mEt;  
70 }
```

Here is the caller graph for this function:



### 6.20.3.3 bool MadSVSurfaceFluxes::fromXml ( const QString *theXml* ) [virtual]

Read this object from xml and return result as true for success, false for failure.

See Also

[MadSerialisable](#)

Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

Definition at line 117 of file madsvsurfacefluxes.cpp.

```

118 {
119     QDomDocument myDocument("mydocument");
120     myDocument.setContent(theXml);
121     QDomElement myTopElement = myDocument.firstChildElement("surfacefluxes");
122     if (myTopElement.isNull())
123     {
124         //TODO - just make this a warning
125         qDebug("the top element couldn't be found!");
126         setGuid(myTopElement.attribute("guid"));
127         //mEt=MadUtils::xmlDecode(myTopElement.firstChildElement("et").text());
128         //mNh3Loss=MadUtils::xmlDecode(myTopElement.firstChildElement("nh3Loss").text());
129         //mN2oLoss=MadUtils::xmlDecode(myTopElement.firstChildElement("n2oLoss").text());
130         //mN2Loss=MadUtils::xmlDecode(myTopElement.firstChildElement("n2Loss").text());
131         //ch4Loss()=MadUtils::xmlDecode(myTopElement.firstChildElement("ch4Loss").text());
132         return true;
133     }
134     else
135         return false;
136 }
```

Here is the call graph for this function:



### 6.20.3.4 bool MadSerialisable::fromXmlFile ( const QString *theFileName* ) [virtual], [inherited]

fromXmlFile Read this object from xml in a file

See Also

[fromXmlFile\(\)](#) Internally it uses [fromXml\(QString\)](#) so that must be properly implemented

Parameters

<i>theFileName</i>	
--------------------	--

**Returns**

result as true for success, false for failure.

Definition at line 76 of file madserialisable.cpp.

```

77 {
78     bool myResult = false;
79     QFile myFile( theFileName );
80     if ( myFile.open( QIODevice::ReadOnly ) )
81     {
82         myResult=this->fromXml(myFile.readAll());
83         myFile.close();
84     }
85     else
86     {
87         //@TODO Error handler!
88         myResult=false;
89     }
90     return myResult ;
91 }
```

Here is the call graph for this function:



#### 6.20.3.5 QString MadGuid::guid ( ) const [inherited]

[MadGuid::guid.](#)

Destructor Retrieve the GUID

**Returns**

Definition at line 40 of file madguid.cpp.

```

41 {
42     return mGuid;
43 }
```

#### 6.20.3.6 MadSubCategory MadSVSurfaceFluxes::n2Loss ( ) const

n2Loss

**Returns**

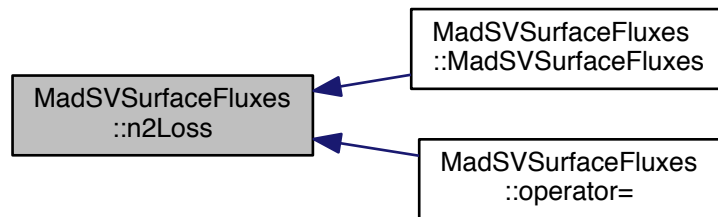
Definition at line 81 of file madsvsurfacefluxes.cpp.

```

82 {
83     return mN2Loss;
84 }
```



Here is the caller graph for this function:



#### 6.20.3.7 MadSubCategory MadSVSurfaceFluxes::n2oLoss ( ) const

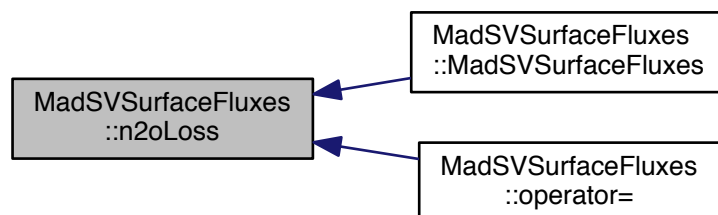
`n2oLoss`

Returns

Definition at line 76 of file `madvsvsurfacefluxes.cpp`.

```
77 {  
78     return mN2oLoss;  
79 }
```

Here is the caller graph for this function:



#### 6.20.3.8 MadSubCategory MadSVSurfaceFluxes::nh3Loss ( ) const

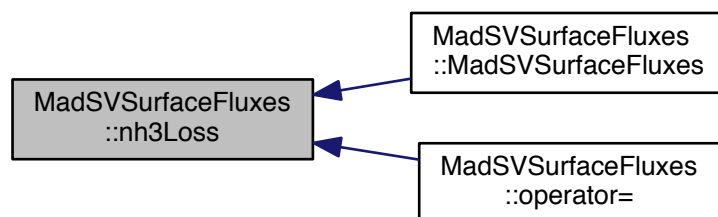
`nh3Loss`

## Returns

Definition at line 71 of file madsvsurfacefluxes.cpp.

```
72 {
73     return mNh3Loss;
74 }
```

Here is the caller graph for this function:

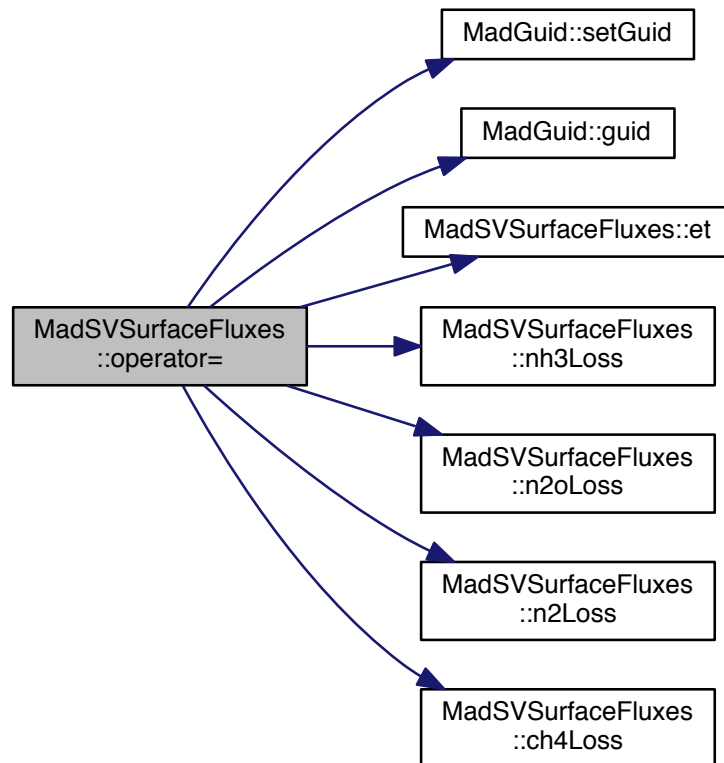


### 6.20.3.9 MadSVSurfaceFluxes & MadSVSurfaceFluxes::operator= ( const MadSVSurfaceFluxes & *theData* )

Definition at line 53 of file madsvsurfacefluxes.cpp.

```
54 {
55     // gracefully handles self assignment
56     if (this == &theData) return *this;
57     setGuid(theData.guid());
58     mEt=theData.et();
59     mNh3Loss=theData.nh3Loss();
60     mN2oLoss=theData.n2oLoss();
61     mN2Loss=theData.n2Loss();
62     mCh4Loss=theData.ch4Loss();
63     return *this;
64 }
```

Here is the call graph for this function:



#### 6.20.3.10 void MadSVSurfaceFluxes::setCh4Loss ( MadSubCategory *theData* )

setCh4Loss

Parameters

<i>theData</i>	
----------------	--

Definition at line 112 of file madsvsurfacefluxes.cpp.

```

113 {
114     mCh4Loss = theData;
115 }
  
```

Here is the caller graph for this function:



#### 6.20.3.11 void MadSVSurfaceFluxes::setEt ( MadSubCategory *theData* )

setEt

##### Parameters

<i>theData</i>	
----------------	--

Definition at line 92 of file `madvsvsurfacefluxes.cpp`.

```

93 {
94     mEt = theData;
95 }
```

Here is the caller graph for this function:



#### 6.20.3.12 void MadGuid::setGuid ( QString *theGuid* = "" ) [inherited]

[MadGuid::setGuid](#).

##### Parameters

<i>theGuid</i>	
----------------	--

Definition at line 49 of file `madguid.cpp`.

```

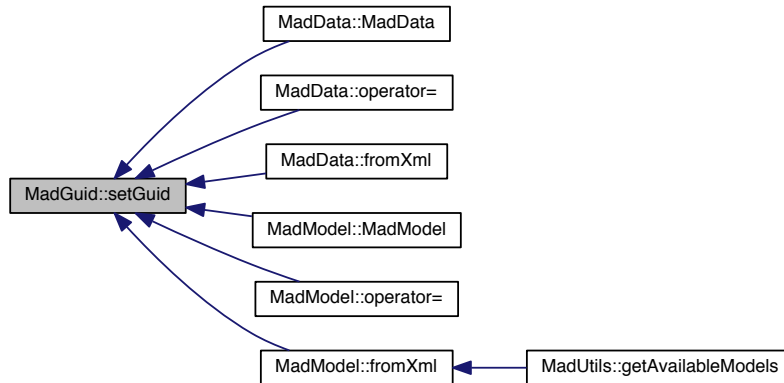
50 {
51     if (theGuid.isEmpty())
52     {
53         mGuid=QUuid::createUuid().toString().replace("{","").replace("}","");
54     }
55     else
56     {
57         mGuid=theGuid;
```

```

58     }
59 }

```

Here is the caller graph for this function:



#### 6.20.3.13 void MadSVSurfaceFluxes::setN2Loss ( MadSubCategory *theData* )

setN2Loss

Parameters

<i>theData</i>	
----------------	--

Definition at line 107 of file `madsvsurfacefluxes.cpp`.

```

108 {
109     mN2Loss = theData;
110 }

```

Here is the caller graph for this function:



#### 6.20.3.14 void MadSVSurfaceFluxes::setN2oLoss ( MadSubCategory *theData* )

setN2oLoss

## Parameters

<i>theData</i>	
----------------	--

Definition at line 102 of file madsvsurfacefluxes.cpp.

```
103 {
104     mN2oLoss = theData;
105 }
```

Here is the caller graph for this function:



#### 6.20.3.15 void MadSVSurfaceFluxes::setNh3Loss ( MadSubCategory *theData* )

setNh3Loss

## Parameters

<i>theData</i>	
----------------	--

Definition at line 97 of file madsvsurfacefluxes.cpp.

```
98 {
99     mNh3Loss = theData;
100 }
```

Here is the caller graph for this function:



#### 6.20.3.16 QString MadSVSurfaceFluxes::toHtml ( )

Return a html text representation of this layer

Definition at line 180 of file madsvsurfacefluxes.cpp.

```
181 {
```

```

182   QString myString;
183   //myString+="<h3>Details for " + MadUtils::xmlEncode(mName) + "</h3>";
184   //myString+="<p>GUID: " + guid() + "</p>";
185   myString+="<table>";
186   //myString+="<tr><td><b>Description: </b></td><td>" + mDescription + "</td></tr>";
187
188   //
189   // the following shows example of how to do a couple of things
190   //
191
192   //myString+="<tr><td><b>Cals/Kg: </b></td><td>" + QString::number(mCropCalories) + "</td></tr>";
193   //QString myCropFodderEnergyType = (mCropFodderEnergyType==0) ? "KCalories" : "TDN";
194   //QString myUnits = (mAreaUnits==0) ? "Dunum" : "Hectare";
195   //myString+="<tr><td><b>Fodder (kg/" + myUnits + ")</b></td><td>" +
196   //    QString::number(mCropFodderProduction) + "</td></tr>";
197   //myString+="<tr><td><b>Fodder Value/Kg: </b></td><td>" + QString::number(mCropFodderValue) +
198   //    "</td></tr>";
199   //myString+="<tr><td><b>FodderEnergyType: </b></td><td>" + myCropFodderEnergyType + "</td></tr>";
200   //myString+="<tr><td><b>AreaUnits: </b></td><td>" + myUnits + "</td></tr>";
201   myString+="</table>";
202   return myString;
203 }

```

### 6.20.3.17 QString MadSVSurfaceFluxes::toText ( )

Return a plain text representation of this layer I need to figure out how to turn the sub category into text

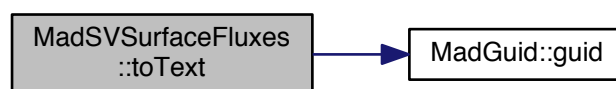
Definition at line 167 of file madsvsurfacefluxes.cpp.

```

168 {
169   QString myString;
170   myString+=QString("guid=>" + guid() + "\n");
171   //myString+=QString("et=>" + MadUtils::xmlEncode(mEt) + "</et>\n");
172   //myString+=QString("nh3Loss=>" + MadUtils::xmlEncode(mNh3Loss) + "</nh3Loss>\n");
173   //myString+=QString("n2oLoss=>" + MadUtils::xmlEncode(mN2oLoss) + "</n2oLoss>\n");
174   //myString+=QString("n2Loss=>" + MadUtils::xmlEncode(mN2Loss) + "</n2Loss>\n");
175   //myString+=QString("ch4Loss=>" + MadUtils::xmlEncode(mCh4Loss) + "</ch4Loss>\n");
176   return myString;
177 }

```

Here is the call graph for this function:



### 6.20.3.18 QString MadSVSurfaceFluxes::toXml ( ) [virtual]

Return an xml representation of this layer

#### Note

this class inherits the serialisable interface so it MUST implement this

Implements [MadSerialisable](#).

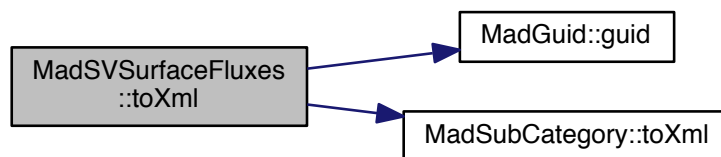
Definition at line 138 of file madsvsurfacefluxes.cpp.

```

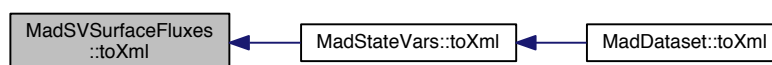
139 {
140     QString myString;
141     myString+=QString("      <surfacefluxes guid=\"\" + guid() + "\">\n");
142
143     myString+=QString("      <et>\n");
144     myString+=mEt.toXml();
145     myString+=QString("      </et>\n");
146
147     myString+=QString("      <nh3loss>");
148     myString+=mNh3Loss.toXml();
149     myString+=QString("      </nh3loss>\n");
150
151     myString+=QString("      <n2oloss>");
152     myString+=mN2oLoss.toXml();
153     myString+=QString("      </n2oloss>\n");
154
155     myString+=QString("      <n2loss>");
156     myString+=mN2Loss.toXml();
157     myString+=QString("      </n2loss>\n");
158
159     myString+=QString("      <ch4loss>");
160     myString+=mCh4Loss.toXml();
161     myString+=QString("      </ch4loss>\n");
162
163     myString+=QString("    </surfacefluxes>\n");
164     return myString;
165 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.20.3.19 bool MadSerialisable::toXmlFile ( const QString theFileName ) [virtual],[inherited]

`toXmlFile` writes object to xml and return result (virtual qstring) We provide a basic default implementation where given a file name, we will write the serialised xml to that file. Internally it uses `toXml()` method so that must be properly implemented.

See Also

[toXml\(\)](#)



## Parameters

<i>theFileName</i>	
--------------------	--

## Returns

QString (virtual)

Definition at line 57 of file madserialisable.cpp.

```

58 {
59     bool myResult = false;
60     QFile myFile( theFileName );
61     if ( myFile.open( QIODevice::WriteOnly ) )
62     {
63         QTextStream myQTextStream( &myFile );
64         myQTextStream << this->toXml();
65         myFile.close();
66         myResult=true;
67     }
68     else
69     {
70         //TODO Error handler!
71         myResult=false;
72     }
73     return myResult ;
74 }
```

Here is the call graph for this function:



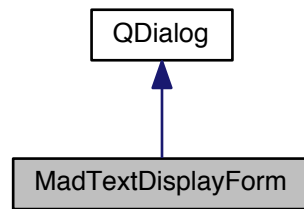
The documentation for this class was generated from the following files:

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/[madsvsurfacefluxes.h](#)
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/[madsvsurfacefluxes.cpp](#)

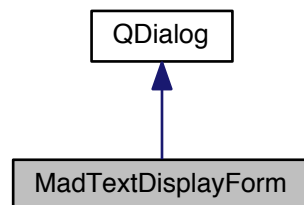
## 6.21 MadTextDisplayForm Class Reference

```
#include <madtextdisplayform.h>
```

Inheritance diagram for MadTextDisplayForm:



Collaboration diagram for MadTextDisplayForm:



## Public Member Functions

- [MadTextDisplayForm](#) (QWidget \*parent=0)
- [~MadTextDisplayForm](#) ()
- void [setText](#) (const QString &theText)

### 6.21.1 Detailed Description

Definition at line 39 of file madtextdisplayform.h.

### 6.21.2 Constructor & Destructor Documentation

#### 6.21.2.1 MadTextDisplayForm::MadTextDisplayForm ( QWidget \* *parent* = 0 ) [explicit]

Definition at line 25 of file madtextdisplayform.cpp.

```
26                                     :  
27   QDialog(parent),  
28   ui(new Ui::MadTextDisplayForm)  
29 {  
30   ui->setupUi(this);  
}
```

### 6.21.2.2 MadTextDisplayForm::~~MadTextDisplayForm ( )

Definition at line 32 of file madtextdisplayform.cpp.

```
33 {
34     delete ui;
35 }
```

## 6.21.3 Member Function Documentation

### 6.21.3.1 void MadTextDisplayForm::setText ( const QString & theText )

Definition at line 37 of file madtextdisplayform.cpp.

```
38 {
39     ui->textBrowser->setText (theText);
40 }
```

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

- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.h
- /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.cpp

## 6.22 MadUtils Class Reference

```
#include <madutils.h>
```

### Public Types

- typedef QMap< QString, [MadModel](#) > [ModelMap](#)  
*ModelMap (typedef) This typedef is used to refer to a collection of layersets. the key is the layerset name the value is the layerset itself.*

### Public Member Functions

- [MadUtils](#) ()
- QString [openGraphicFile](#) ()
- QString [saveFile](#) ()

### Static Public Member Functions

- static const QString [userSettingsDirPath](#) ()  
*userSettingsDirPath Find the place on the filesystem where user data is stored*
- static const QString [userModelProfilesDirPath](#) ()  
*userModelProfilesDirPath Find the place on the filesystem where user defined model profiles are stored.*
- static const QString [userModelParametersDirPath](#) ()  
*userModelParametersDirPath Find the place on the filesystem where user defined model parameter profiles are stored.*
- static const QString [getModelOutputDir](#) ()  
*getModelOutputDir Get the place where model outputs are to be stored. By default this is in ~/.macsurAdapter/model-Outputs But if modelOutputsDir is specified in QSettings, it will override the default.*
- static const QString [userImagesDirPath](#) ()  
*userImagesDirPath Find the place on the filesystem where user images are stored.*

- static [MadUtils::ModelMap](#) [getAvailableModels](#) ()  
*getAvailableModels* Get a QMap of the available layersets in the users layersets directory
- static [MadModel](#) [getModel](#) (QString theGuid)  
*getModel* Get a [MadModel](#) given its GUID. If no matching model is found, a blank one is returned.
- static QStringList [sortList](#) (QStringList theList)  
*sortList* Sort a string list into descending alphabetic order and return the result.
- static QStringList [uniqueList](#) (QStringList theList)  
*uniqueList* Remove any duplicate entries from a sorted list
- static bool [createTextFile](#) (QString theFileName, QString theData)  
*createTextFile* A helper method to easily write a file to disk.
- static QString [xmlEncode](#) (QString theString)  
*xmlEncode* A helper method to xml encode any special chars in a string (< > & etc) will become (< > & etc)
- static QString [xmlDecode](#) (QString theString)  
*xmlDecode* A helper method to xml decode any special chars in a string (< > & etc) will become (< > & etc)
- static QString [getStandardCss](#) ()  
*getStandardCss* Get the standard style sheet for reports. Typically this will be used like this: `QString myStyle = getStandardCss(); textBrowserFoo->document()->setDefaultStylesheet(myStyle);`
- static const QString [userConversionTablesDirPath](#) ()

### 6.22.1 Detailed Description

Definition at line 41 of file madutils.h.

### 6.22.2 Member Typedef Documentation

#### 6.22.2.1 typedef QMap<QString,MadModel> MadUtils::ModelMap

ModelMap (typedef) This typedef is used to refer to a collection of layersets. the key is the layerset name the value is the layerset itself.

Definition at line 101 of file madutils.h.

### 6.22.3 Constructor & Destructor Documentation

#### 6.22.3.1 MadUtils::MadUtils ( )

Definition at line 44 of file madutils.cpp.

```
45 {
46 }
```

### 6.22.4 Member Function Documentation

#### 6.22.4.1 bool MadUtils::createTextFile ( QString *theFileName*, QString *theData* ) [static]

*createTextFile* A helper method to easily write a file to disk.

#### Parameters

<i>theFileName</i>	- the filename to be created or overwritten
<i>theData</i>	- the data that will be written into the file

**Returns**

bool - false if the file could not be written

Definition at line 126 of file madutils.cpp.

```

127 {
128     //create the txt file
129     QFile myFile( theFileName );
130     if ( myFile.open( QIODevice::WriteOnly ) )
131     {
132         QTextStream myQTextStream( &myFile );
133         myQTextStream << theData;
134     }
135     else
136     {
137         return false;
138     }
139     myFile.close();
140     return true ;
141 }
```

**6.22.4.2 MadUtils::ModelMap MadUtils::getAvailableModels ( ) [static]**

getAvailableModels Get a QMap of the available layersets in the users layersets directory

**Returns**

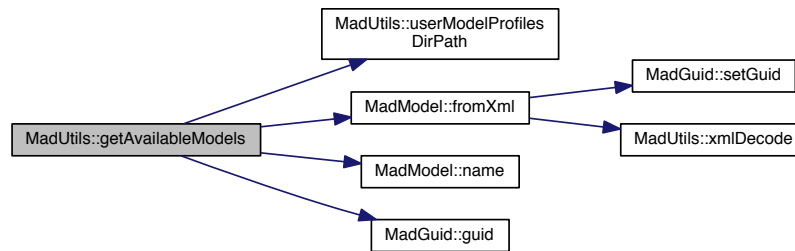
a QMap<QString,OmgLayerSet> where the QString key is the layerset name

Definition at line 93 of file madutils.cpp.

```

94 {
95     MadUtils::ModelMap myMap;
96     QDir myDirectory(userModelProfilesDirPath());
97     myDirectory.setFilter(QDir::Dirs | QDir::Files | QDir::NoSymLinks );
98     QFileInfoList myList = myDirectory.entryInfoList();
99     for (unsigned int i = 0; i < static_cast<unsigned int>(myList.size()); ++i)
100     {
101         QFileInfo myFileInfo = myList.at(i);
102         //Ignore directories
103         if(myFileInfo.fileName() == "." ||myFileInfo.fileName() == ".." )
104         {
105             continue;
106         }
107         //if the filename ends in .xml try to load it into our layerSets listing
108         if(myFileInfo.completeSuffix()=="xml")
109         {
110             //QDebug("Loading model: " + myList.at(i).absoluteFilePath().toLocal8Bit());
111             MadModel myModel;
112             myModel.fromXml(myFileInfo.absoluteFilePath());
113             if (myModel.name().isEmpty())
114             {
115                 //QDebug("Model name was empty!");
116                 continue;
117             }
118             //QDebug("Adding " + myModel.name());
119             myMap[myModel.guid()]=myModel;
120             //QDebug(myModel.toText().toLocal8Bit());
121         }
122     }
123     return myMap;
124 }
```

Here is the call graph for this function:



#### 6.22.4.3 static MadModel MadUtils::getModel ( QString *theGuid* ) [static]

getModel Get a [MadModel](#) given its GUID. If no matching model is found, a blank one is returned.

#### 6.22.4.4 const QString MadUtils::getModelOutputDir ( ) [static]

getModelOutputDir Get the place where model outputs are to be stored. By default this is in ~/.macsur-Adapter/modelOutputs But if modelOutputsDir is specified in QSettings, it will override the default.

Definition at line 60 of file madutils.cpp.

```

61 {
62     QString myPath = userSettingsDirPath()+QDir::separator()+"modelOutputs"+
        QDir::separator();
63     QDir().mkpath(myPath);
64     return myPath;
65 }
  
```

Here is the call graph for this function:



#### 6.22.4.5 QString MadUtils::getStandardCss ( ) [static]

getStandardCss Get the standard style sheet for reports. Typically this will be used like this: `QString myStyle = getStandardCss\(\); textBrowserFoo->document()->setDefaultStyleSheet(myStyle);`

Definition at line 159 of file madutils.cpp.

```

160 {
161     QString myStyle = ".glossy{ background-color: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 #616161,
        stop: 0.5 #505050, stop: 0.6 #434343, stop:1 #656565); color: white; padding-left: 4px; border: 1px solid
        #6c6c6c; }";
162     myStyle += "body {background: white;}";
163     myStyle += "h1 {font-size : 22pt; color: #0063F7; }";
164     myStyle += "h2 {font-size : 18pt; color: #0063F7; }";
  
```

```

165 myStyle += "h3 {font-size : 14pt; color: #0063F7; }";
166 myStyle += ".cellHeader {color:#466aa5; font-size : 12pt;}";
167 myStyle += ".parameterHeader {font-weight: bold;}";
168 myStyle += ".largeCell {color:#000000; font-size : 12pt;}";
169 myStyle += ".table {"
170     " border-width: 1px 1px 1px 1px;"
171     " border-spacing: 2px;"
172     " border-style: solid solid solid solid;"
173     " border-color: black black black black;"
174     " border-collapse: separate;"
175     " background-color: white;"
176     "}";
177 return myStyle;
178 }

```

#### 6.22.4.6 QString MadUtils::openGraphicFile ( )

Definition at line 180 of file madutils.cpp.

```

181 {
182     QString myHomePath = QDir::homePath();
183     QString myFileName = QFileDialog::getOpenFileName(0, "Choose an image", myHomePath, "Images (*.png *.xpm
        *.jpg)");
184     QFileInfo fi(myFileName);
185     QString myName = fi.fileName();
186     QString myDestinationFilePathName = userImagesDirPath() + myName;
187     QFile::copy(myFileName, myDestinationFilePathName);
188     return myDestinationFilePathName;
189 }

```

Here is the call graph for this function:



#### 6.22.4.7 QString MadUtils::saveFile ( )

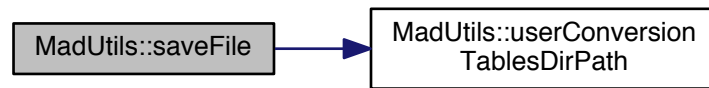
Definition at line 191 of file madutils.cpp.

```

192 {
193     QString myHomePath = QDir::homePath();
194     QString myFileName = QFileDialog::getSaveFileName(0, "Choose a file name",
        userConversionTablesDirPath(), "*.csv");
195     QFileInfo fi(myFileName);
196     QString myName = fi.fileName();
197     QString myDestinationFilePathName = userConversionTablesDirPath() + myName;
198     return myDestinationFilePathName;
199 }

```

Here is the call graph for this function:



#### 6.22.4.8 static QStringList MadUtils::sortList ( QStringList *theList* ) [static]

sortList Sort a string list into descending alphabetic order and return the result.

##### Parameters

<i>theList</i>	- the QStringList to be sorted
----------------	--------------------------------

##### Returns

QStringList - sorted in descending alphabetical order

#### 6.22.4.9 static QStringList MadUtils::uniqueList ( QStringList *theList* ) [static]

uniqueList Remove any duplicate entries from a sorted list

##### Parameters

<i>theList</i>	- the QStringList to be sorted
----------------	--------------------------------

##### Returns

QStringList - a list with no sequential duplicates

#### 6.22.4.10 const QString MadUtils::userConversionTablesDirPath ( ) [static]

Find the place on the filesystem where user created conversion tables in csv format are stored

Typically this will be ~/.macsurAdapter/conversionTables

##### Returns

QString containing the relevant directory name

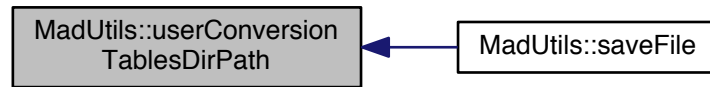
Definition at line 201 of file madutils.cpp.

```

202 {
203     // always saved in the users home dir under .macsurAdapter/
204     QString myPath = QDir::homePath() + QString("/.macsurAdapter/") +
205         QDir::separator() + "conversionTables" + QDir::separator();
206     QDir().mkpath(myPath);
207     return myPath;
208 }
  
```



Here is the caller graph for this function:



#### 6.22.4.11 `const QString MadUtils::userImagesDirPath ( ) [static]`

`userImagesDirPath` Find the place on the filesystem where user images are stored.

Typically this will be `~/macsurAdapter/images`

##### Returns

QString containing the relevant directory name

Definition at line 85 of file `madutils.cpp`.

```

86 {
87     QString myPath = QDir::homePath() + QString("/.macsurAdapter") +
88         QDir::separator() + "images" + QDir::separator();
89     QDir().mkpath(myPath);
90     return myPath;
91 }
  
```

Here is the caller graph for this function:



#### 6.22.4.12 `const QString MadUtils::userModelParametersDirPath ( ) [static]`

`userModelParametersDirPath` Find the place on the filesystem where user defined model parameter profiles are stored.

Typically this will be `~/macsurAdapter/animalParameters`

##### Returns

QString containing the relevant directory name

Definition at line 76 of file `madutils.cpp`.

```

77 {
78     //alg profiles are always saved in the users home dir under .macsurAdapter/
79     QString myPath = QDir::homePath() + QString("/.macsurAdapter/") +
80         QDir::separator() + "modelParameterProfiles" + QDir::separator();
81     QDir().mkpath(myPath);
82     return myPath;
83 }

```

#### 6.22.4.13 const QString MadUtils::userModelProfilesDirPath ( ) [static]

userModelProfilesDirPath Find the place on the filesystem where user defined model profiles are stored.

Typically this will be ~/.macsurAdapter/modelProfiles

##### Returns

QString containing the relevant directory name

Definition at line 67 of file madutils.cpp.

```

68 {
69     //alg profiles are always saved in the users home dir under .macsurAdapter
70     QString myPath = QDir::homePath() + QString("/.macsurAdapter/") +
71         QDir::separator() + "animalProfiles" + QDir::separator();
72     QDir().mkpath(myPath);
73     return myPath;
74 }

```

Here is the caller graph for this function:



#### 6.22.4.14 const QString MadUtils::userSettingsDirPath ( ) [static]

userSettingsDirPath Find the place on the filesystem where user data is stored

Typically, this will be ~/.macsurAdapter

##### Returns

QString containing the relevant directory name

Returns the path to the settings directory in user's home dir

Definition at line 51 of file madutils.cpp.

```

52 {
53     QSettings mySettings;
54     QString myPath=
55         mySettings.value("dataDirs/dataDir", QDir::homePath() + QString("/.macsurAdapter/") ).toString();
56     // Make sure the users settings dir actually exists
57     QDir().mkpath(myPath);
58     return myPath;
59 }

```

Here is the caller graph for this function:



#### 6.22.4.15 QString MadUtils::xmlDecode ( QString *theString* ) [static]

xmlDecode A helper method to xml deencode any special chars in a string (< > & etc) will become (< > & etc)

##### Parameters

QString	- the string to be properly decoded
---------	-------------------------------------

##### Returns

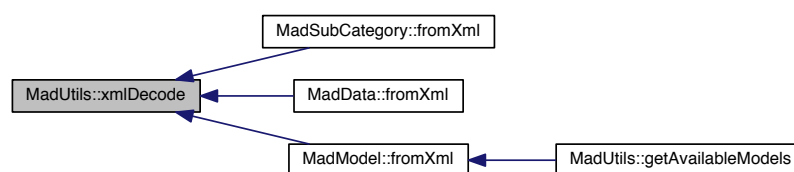
A QString with the encoded chars properly decoded

Definition at line 151 of file madutils.cpp.

```

152 {
153     theString = theString.replace("<", "<");
154     theString = theString.replace(">", ">");
155     theString = theString.replace("&", "&");
156     return theString;
157 }
  
```

Here is the caller graph for this function:



#### 6.22.4.16 QString MadUtils::xmlEncode ( QString *theString* ) [static]

xmlEncode A helper method to xml encode any special chars in a string (< > & etc) will become (< > & etc)

##### Parameters

QString	- the string to be properly encoded
---------	-------------------------------------

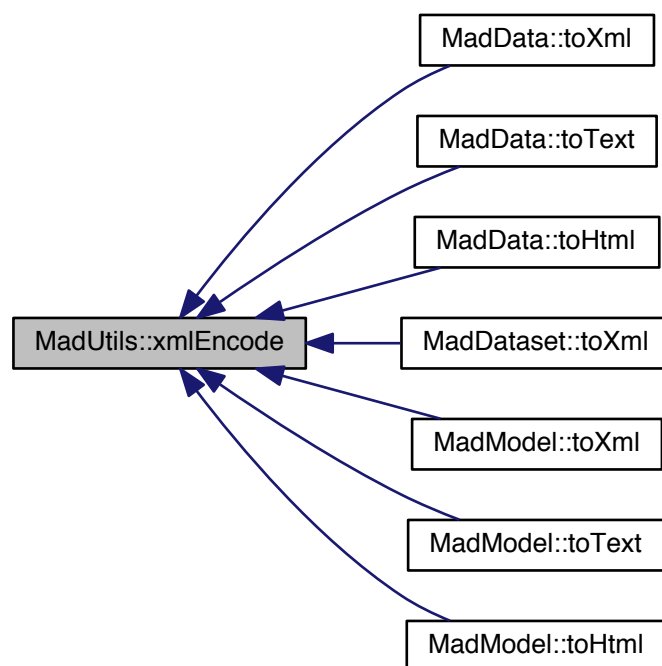
### Returns

A QString with the special chars properly encoded

Definition at line 143 of file madutils.cpp.

```
144 {  
145     theString = theString.replace("<", "&lt;");  
146     theString = theString.replace(">", "&gt;");  
147     theString = theString.replace("&", "&amp;");  
148     return theString;  
149 }
```

Here is the caller graph for this function:

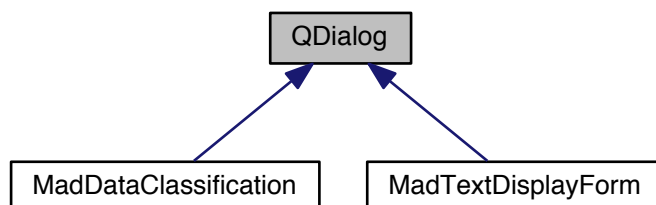


The documentation for this class was generated from the following files:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.h](#)
- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.cpp](#)

## 6.23 QDialog Class Reference

Inheritance diagram for QDialog:

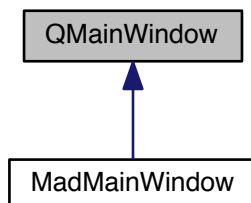


The documentation for this class was generated from the following file:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.h](#)

## 6.24 QMainWindow Class Reference

Inheritance diagram for QMainWindow:



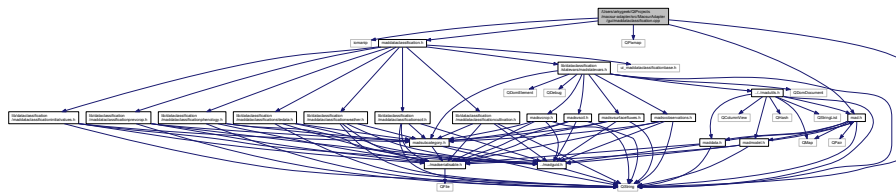
The documentation for this class was generated from the following file:

- [/Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.h](#)



## File Documentation

```
#include <iomanip>
#include <QString>
#include <QPixmap>
#include "maddataclassification.h"
#include "lib/mad.h"
Include dependency graph for maddataclassification.cpp:
```



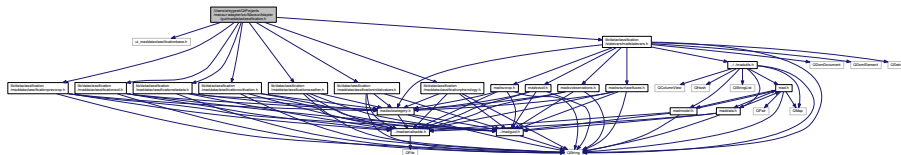
- QString **makeString** ()

#### 7.1.1.1 QString makeString ( )

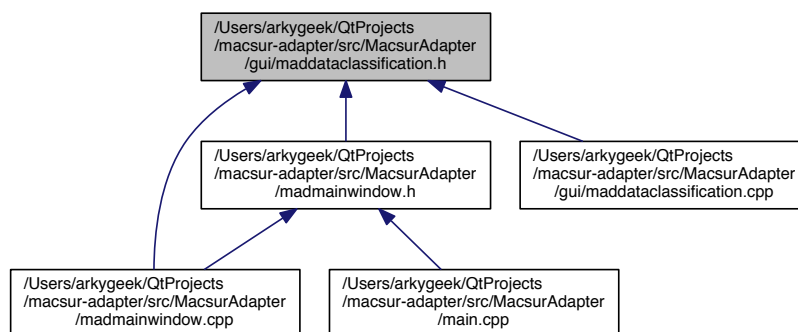
```
#include "ui_maddataclassificationbase.h"
```

```
#include "lib/dataclassification/maddataclassificationcultivation.h"
#include "lib/dataclassification/maddataclassificationinitialvalues.h"
#include "lib/dataclassification/maddataclassificationphenology.h"
#include "lib/dataclassification/maddataclassificationprevcrop.h"
#include "lib/dataclassification/maddataclassificationsoil.h"
#include "lib/dataclassification/maddataclassificationweather.h"
#include "lib/dataclassification/statevars/madstatevars.h"
```

Include dependency graph for maddataclassification.h:



This graph shows which files directly or indirectly include this file:



## Classes

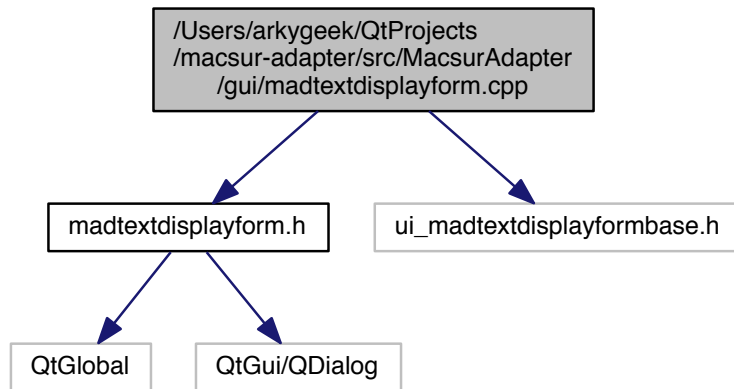
- class [MadDataClassification](#)

## 7.3 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.cpp File Reference

```
#include "madtextdisplayform.h"
#include "ui_madtextdisplayformbase.h"
```



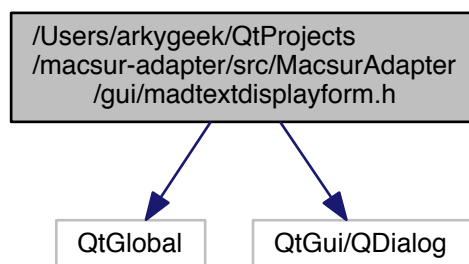
Include dependency graph for madtextdisplayform.cpp:



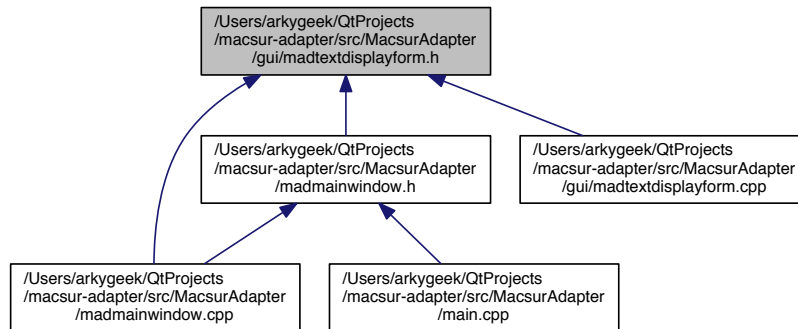
## 7.4 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/gui/madtextdisplayform.h File Reference

```
#include <QtGlobal>
#include <QtGui/QDialog>
```

Include dependency graph for madtextdisplayform.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadTextDisplayForm](#)

## Namespaces

- namespace [Ui](#)

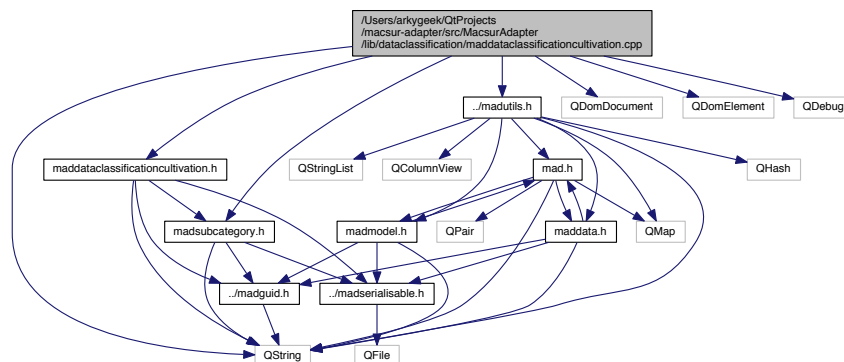
## 7.5 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationcultivation.h File Reference

```

#include "maddataclassificationcultivation.h"
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>

```

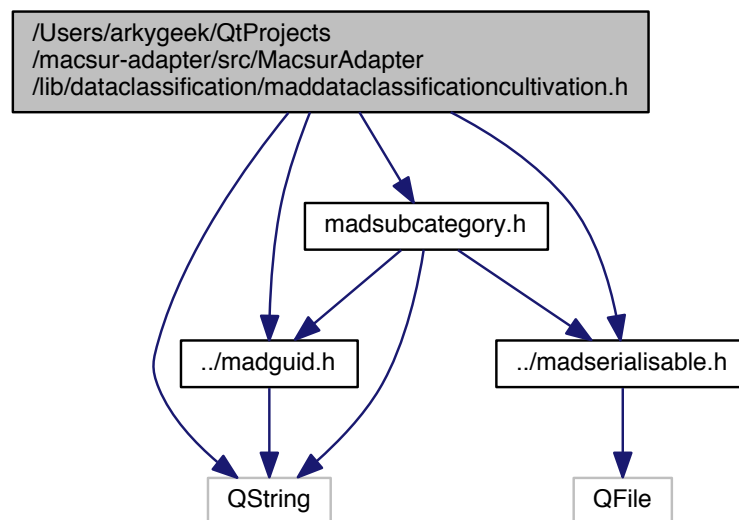
Include dependency graph for maddataclassificationcultivation.cpp:



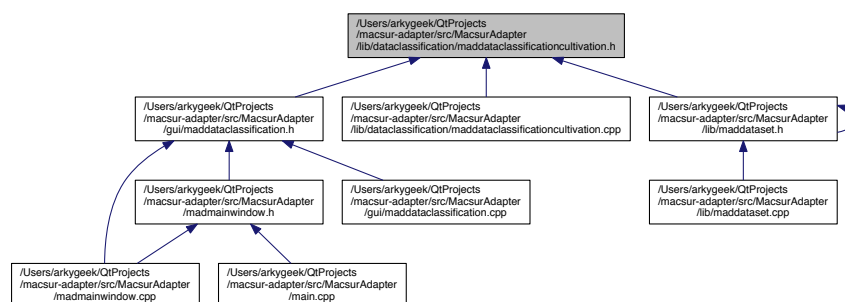
## File Reference

```
#include "madsubcategory.h"
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
```

Include dependency graph for maddataclassificationcultivation.h:



This graph shows which files directly or indirectly include this file:



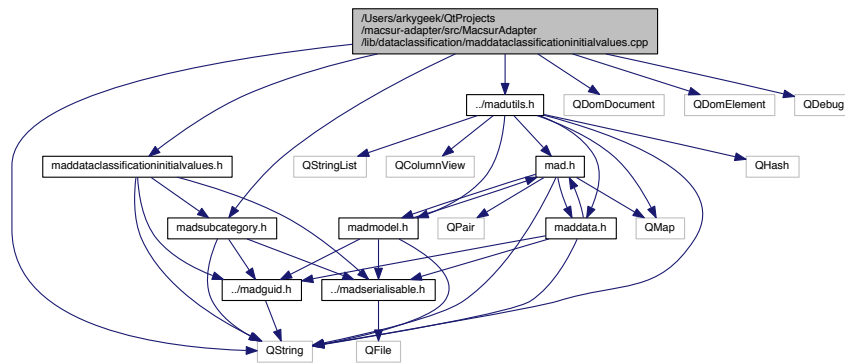
## Classes

- class [MadDataClassificationCultivation](#)

## 7.7 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclass File Reference

```
#include "maddataclassificationinitialvalues.h"
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

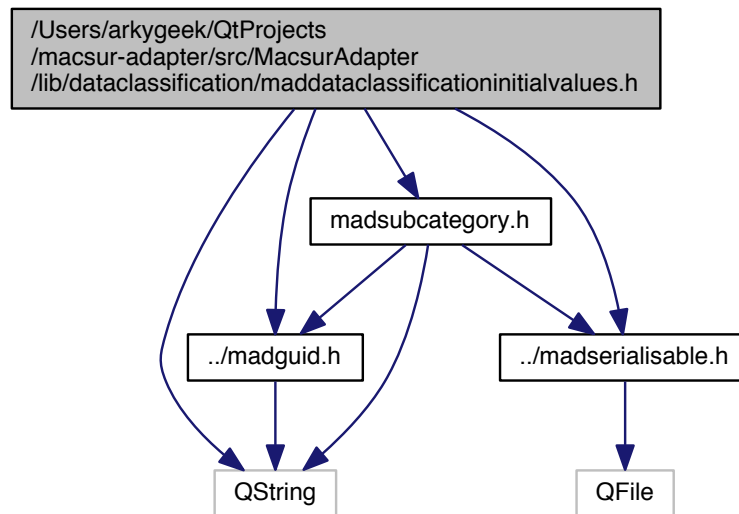
Include dependency graph for maddataclassificationinitialvalues.cpp:



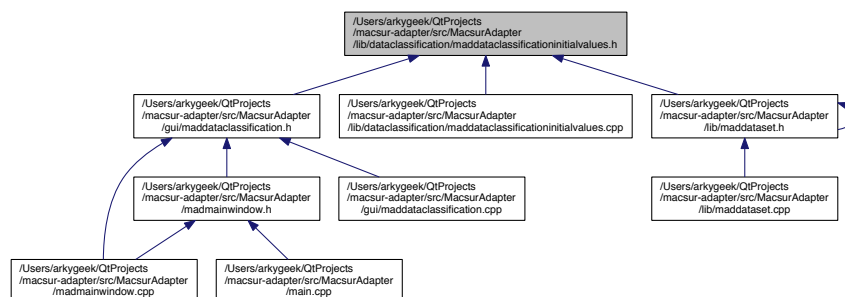
## 7.8 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclass File Reference

```
#include "madsubcategory.h"
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
```

Include dependency graph for maddataclassificationinitialvalues.h:



This graph shows which files directly or indirectly include this file:



## Classes

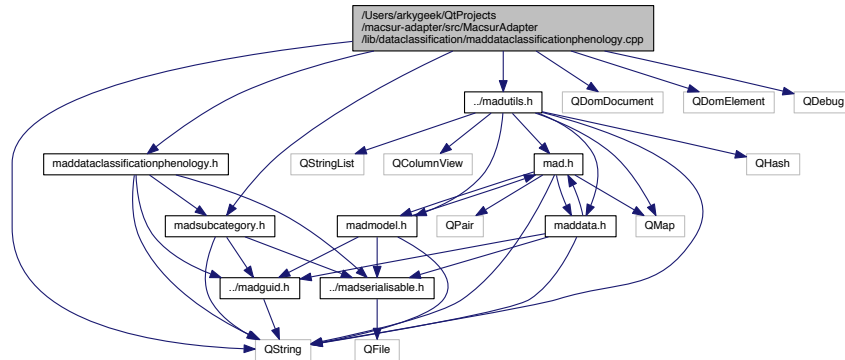
- class [MadDataClassificationInitialValues](#)

## 7.9 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationphenology.h File Reference

```
#include "maddataclassificationphenology.h"
```

```
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

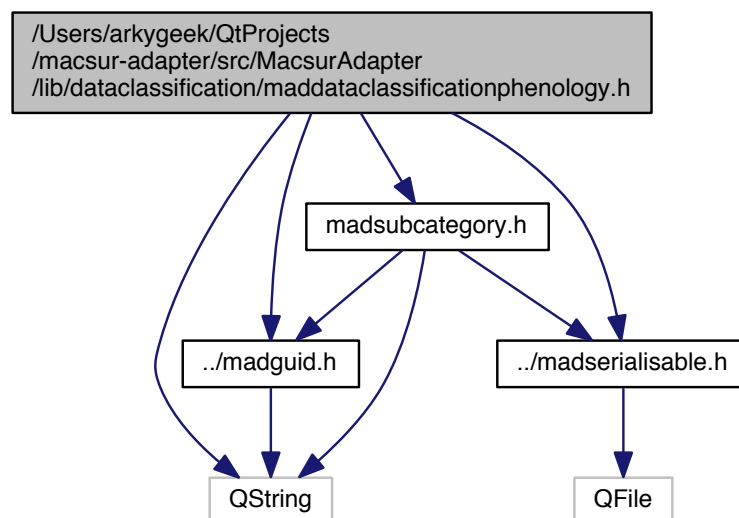
Include dependency graph for maddataclassificationphenology.cpp:



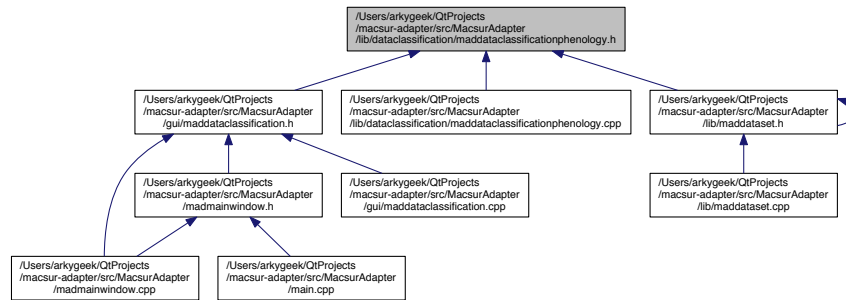
## 7.10 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```
#include "madsubcategory.h"
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
```

Include dependency graph for maddataclassificationphenology.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadDataClassificationPhenology](#)

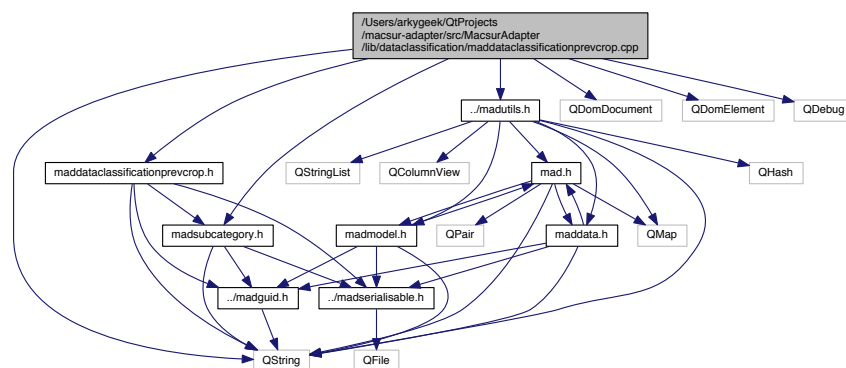
## 7.11 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```

#include "maddataclassificationprevcrop.h"
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>

```

Include dependency graph for maddataclassificationprevcrop.cpp:



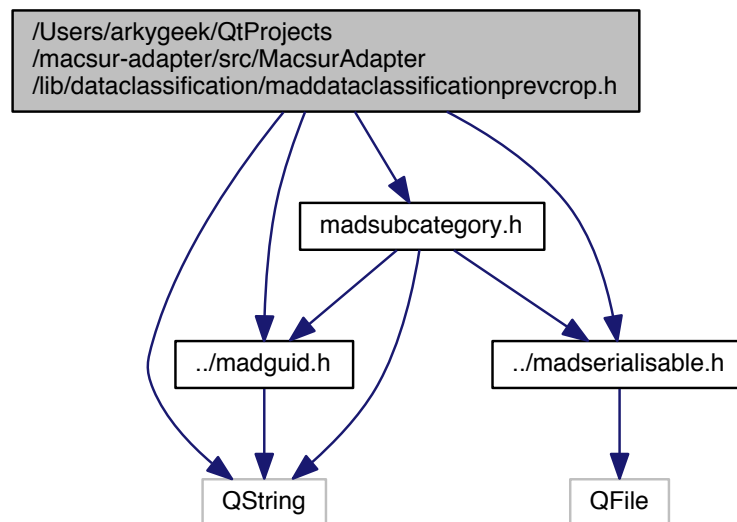
## 7.12 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```

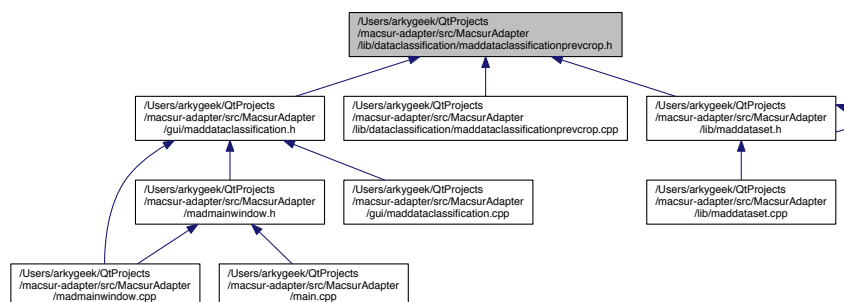
#include "madsubcategory.h"

```

```
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
Include dependency graph for maddataclassificationprevcrop.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadDataClassificationPrevCrop](#)

## 7.13 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

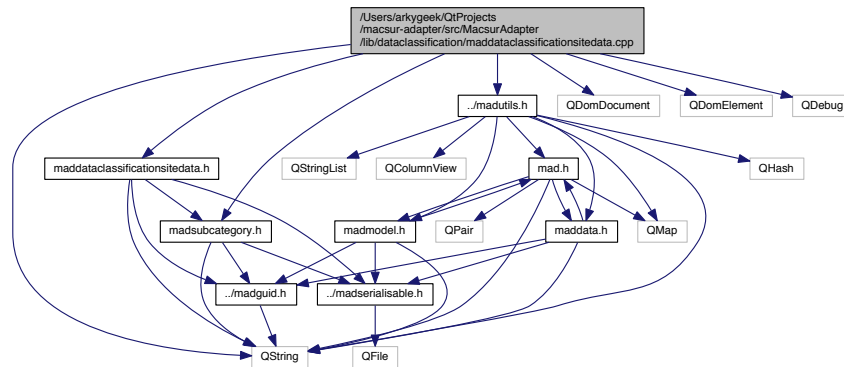
```
#include "maddataclassificationssitedata.h"
```



## Reference

```
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

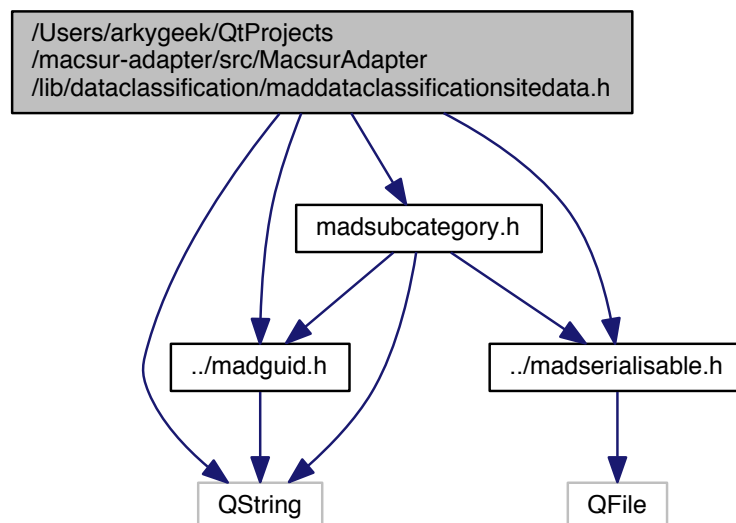
Include dependency graph for maddataclassificationsitedata.cpp:



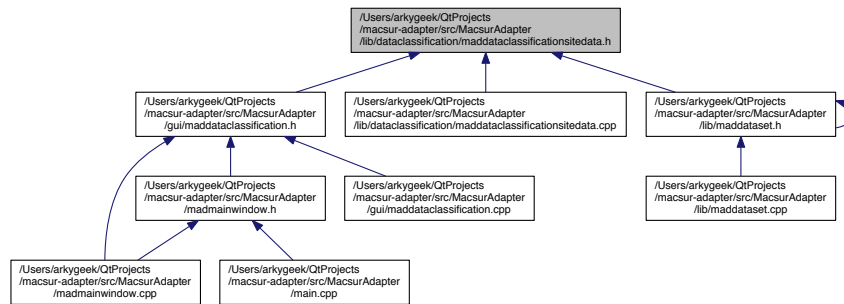
## 7.14 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```
#include "madsubcategory.h"
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
```

Include dependency graph for maddataclassificationsitedata.h:



This graph shows which files directly or indirectly include this file:



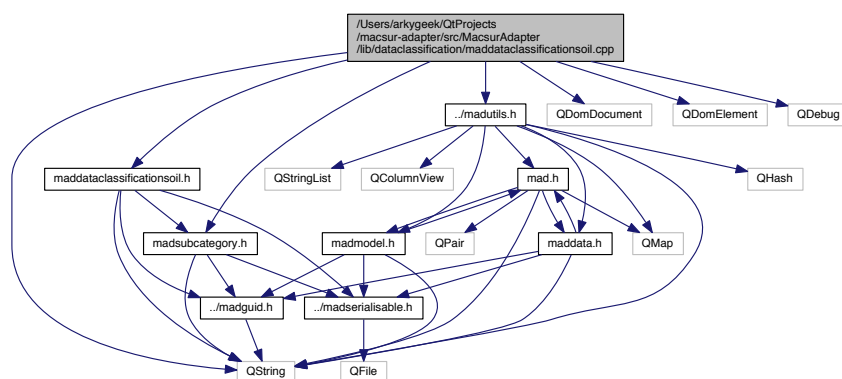
## Classes

- class [MadDataClassificationSiteData](#)

## 7.15 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```
#include "maddataclassificationsoil.h"
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

Include dependency graph for maddataclassificationsoil.cpp:



## 7.16 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

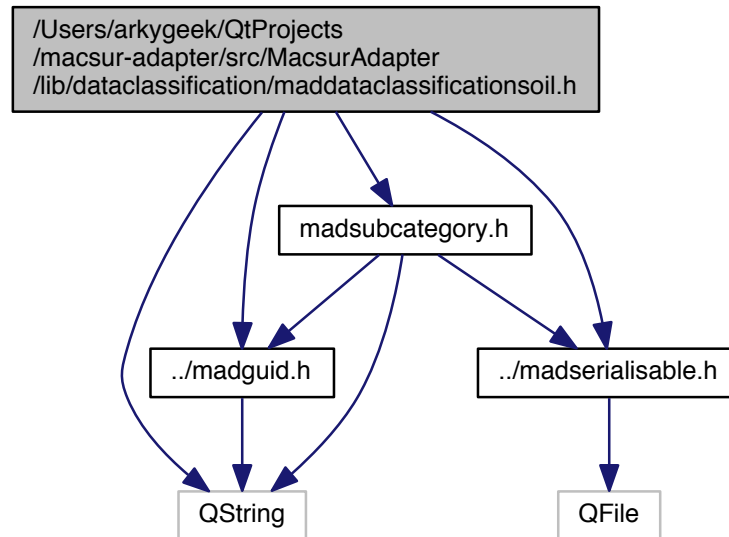
```
#include "madsubcategory.h"
```

```
#include "../madguid.h"
```

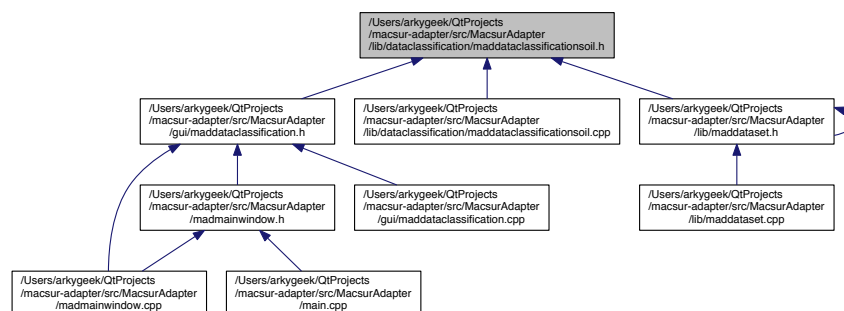
```
#include "../madserialisable.h"
```

```
#include <QString>
```

Include dependency graph for maddataclassificationsoil.h:



This graph shows which files directly or indirectly include this file:



## Classes

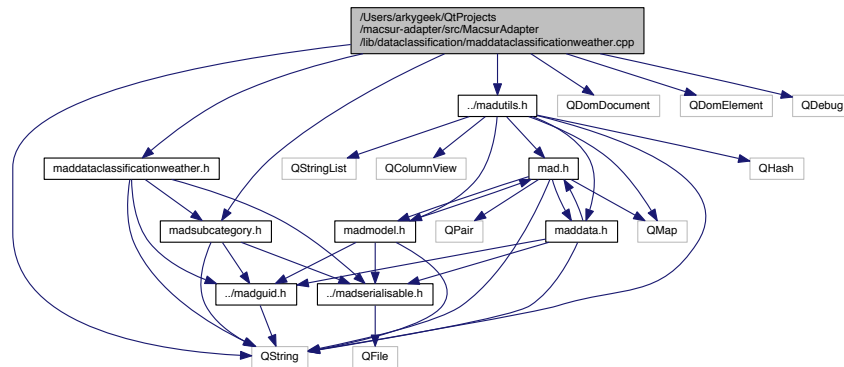
- class [MadDataClassificationSoil](#)

## 7.17 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclassificationweather.h File Reference

```
#include "maddataclassificationweather.h"
```

```
#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

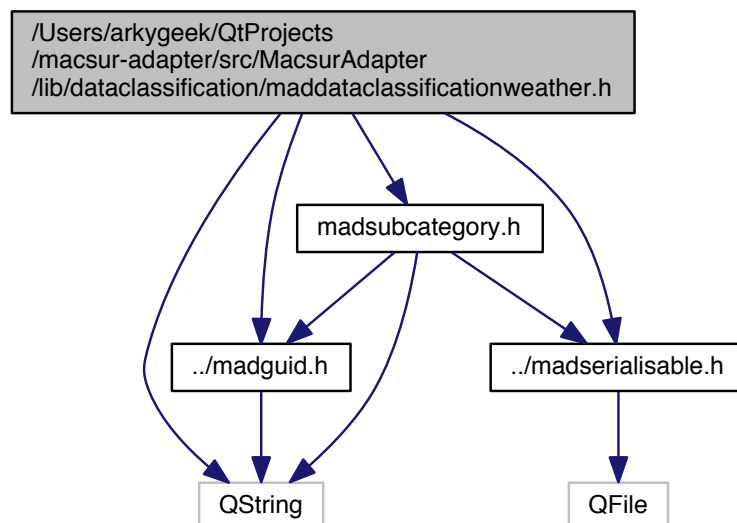
Include dependency graph for maddataclassificationweather.cpp:



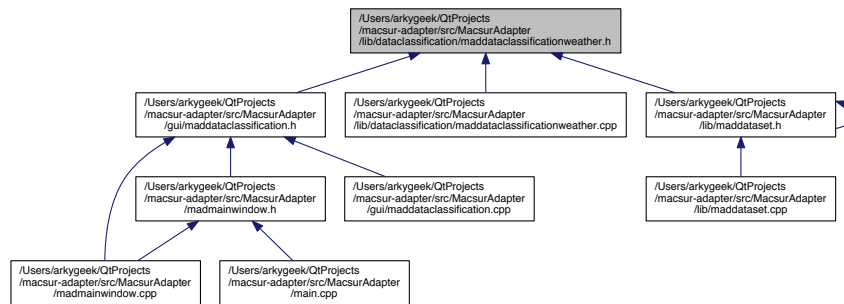
## 7.18 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/maddataclas File Reference

```
#include "madsubcategory.h"
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>
```

Include dependency graph for maddataclassificationweather.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadDataClassificationWeather](#)

## 7.19 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcate

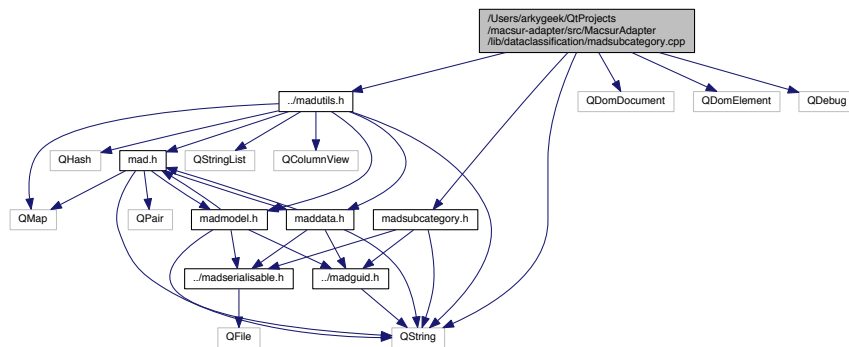
### File Reference

```

#include "madsubcategory.h"
#include "../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>

```

Include dependency graph for madsubcategory.cpp:



## 7.20 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/madsubcate

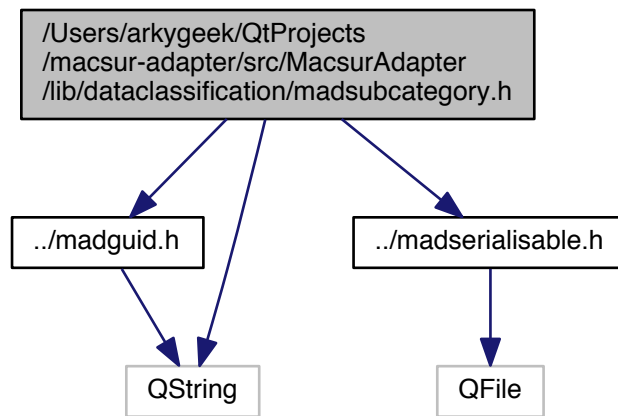
### File Reference

```

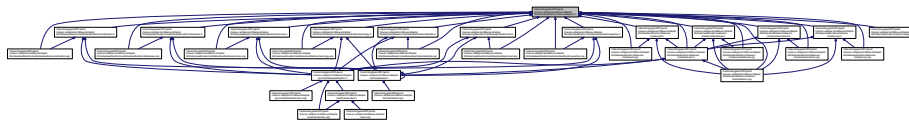
#include "../madguid.h"
#include "../madserialisable.h"
#include <QString>

```

Include dependency graph for madsubcategory.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadSubCategory](#)

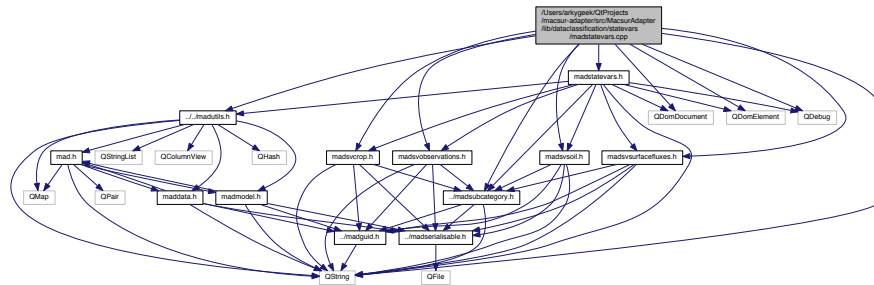
## 7.21 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.h File Reference

```

#include "madstatevars.h"
#include "madsvcrop.h"
#include "madsvsoil.h"
#include "madsvsurfacefluxes.h"
#include "madsvobservations.h"
#include "../madsubcategory.h"
#include ../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>

```

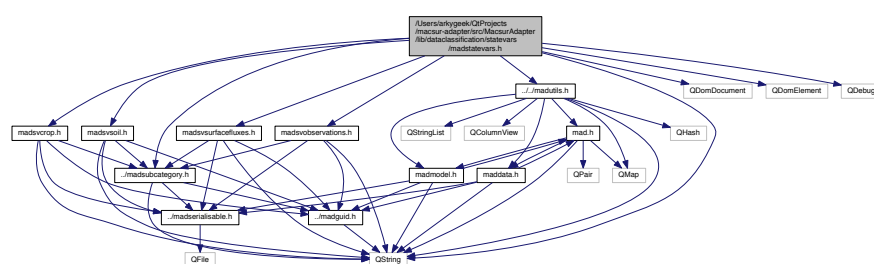
Include dependency graph for madstatevars.cpp:



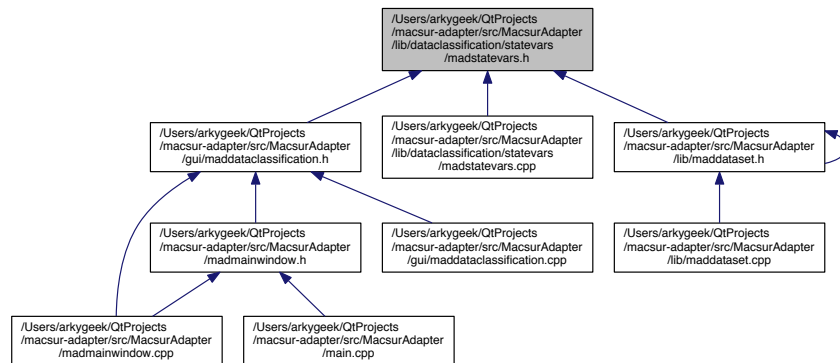
## 7.22 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.h File Reference

```
#include "madvcrop.h"
#include "madsoil.h"
#include "madvsurfacefluxes.h"
#include "madvobservations.h"
#include "../madsubcategory.h"
#include "../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

Include dependency graph for madstatevars.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadStateVars](#)

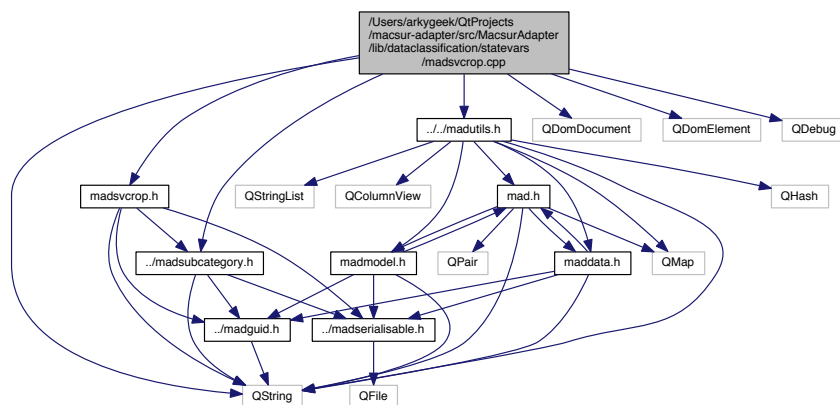
## 7.23 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madstatevars.h File Reference

```

#include "madsvcrop.h"
#include "../madsubcategory.h"
#include "../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>

```

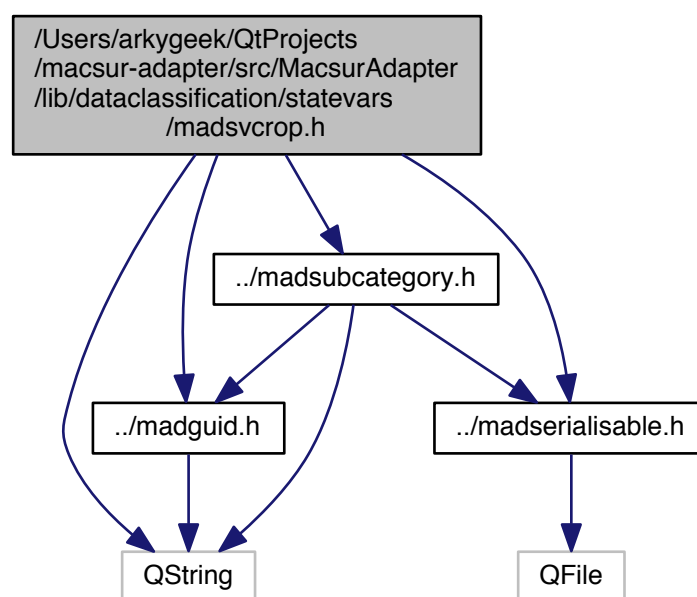
Include dependency graph for madsvcrop.cpp:



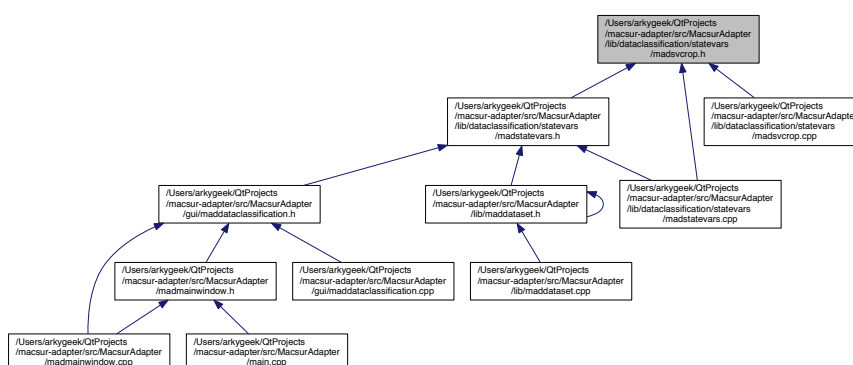


```
#include "../madsubcategory.h"
#include "../../madguid.h"
#include "../../madserialisable.h"
#include <QString>
```

Include dependency graph for madsvcrop.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [MadSVCrop](#)

### 7.24.1 Detailed Description

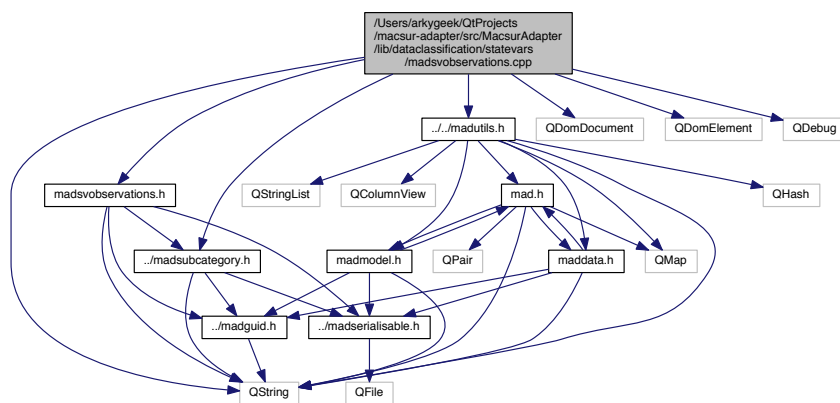
The [MadStateVars](#) class. This contains 4 sub categories.

Definition in file [madsvcrop.h](#).

## 7.25 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.h

```
#include "madsvobservations.h"
#include "../madsubcategory.h"
#include "../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

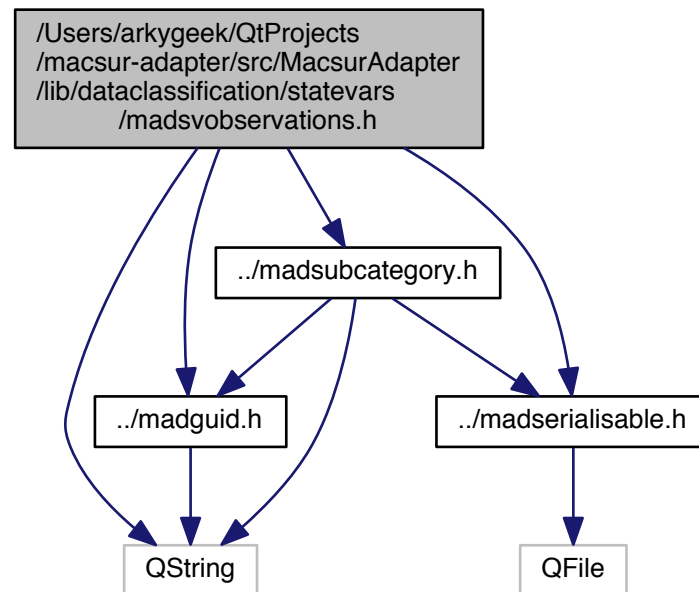
Include dependency graph for madsvobservations.cpp:



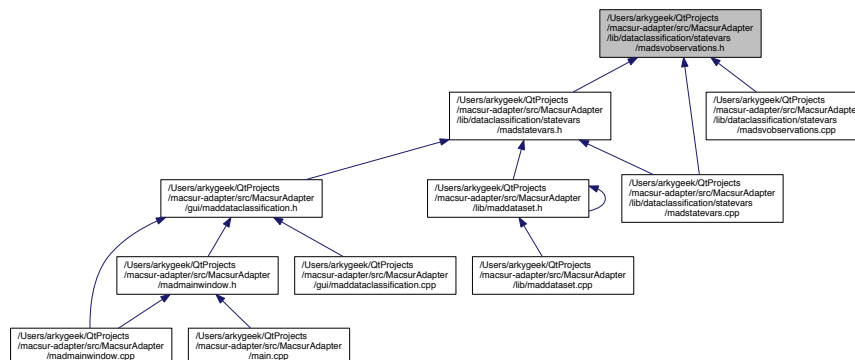
## 7.26 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvcrop.h

```
#include "../madsubcategory.h"
#include "../../madguid.h"
#include "../../madserialisable.h"
#include <QString>
```

Include dependency graph for madsvobservations.h:



This graph shows which files directly or indirectly include this file:



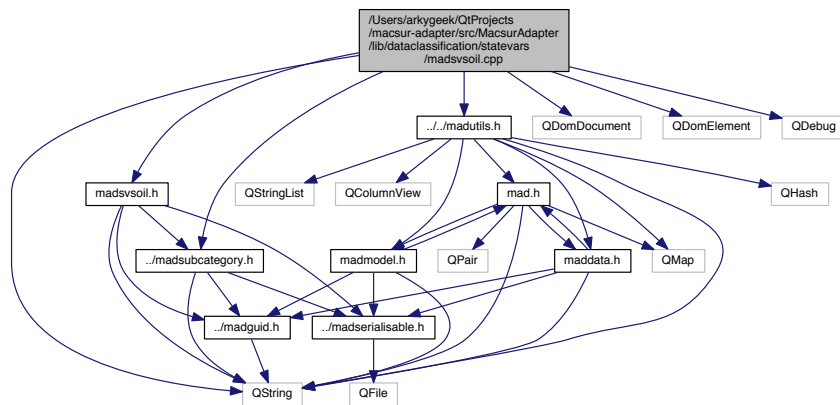
## Classes

- class [MadSVObservations](#)

## 7.27 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madvsoil.cpp File Reference

```
#include "madvsoil.h"
```

```
#include "../madsubcategory.h"
#include "../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
Include dependency graph for madsvsoil.cpp:
```

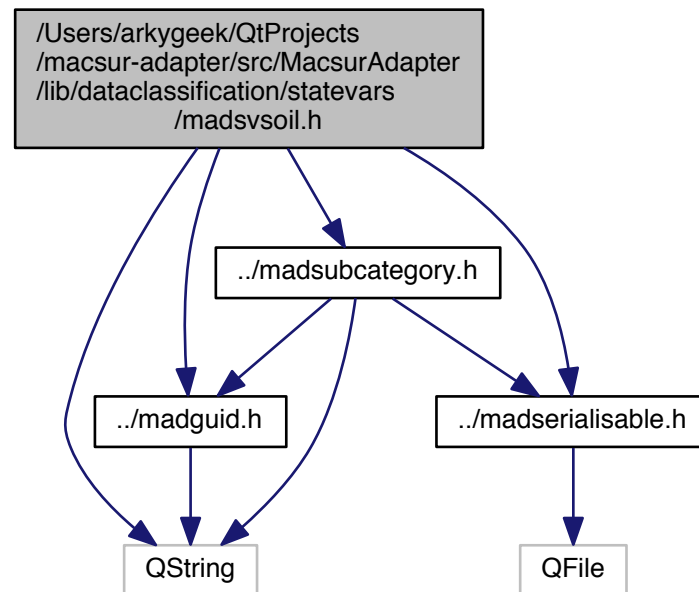


## 7.28 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsoil.cpp

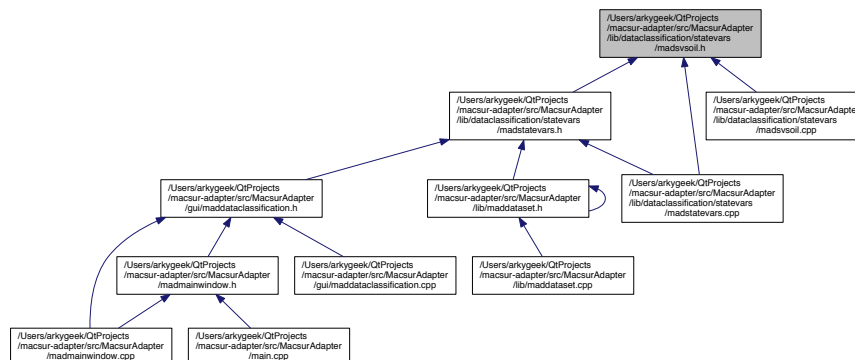
### File Reference

```
#include "../madsubcategory.h"
#include "../../madguid.h"
#include "../../madserialisable.h"
#include <QString>
```

Include dependency graph for madsvsoil.h:



This graph shows which files directly or indirectly include this file:



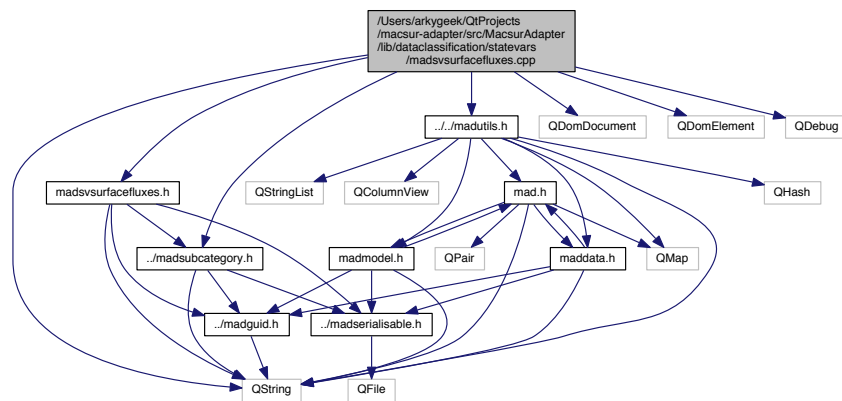
## Classes

- class [MadSVSoil](#)

## 7.29 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.cpp File Reference

```
#include "madsvsurfacefluxes.h"
```

```
#include "../madsubcategory.h"
#include "../../madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
Include dependency graph for madsvsurfacefluxes.cpp:
```

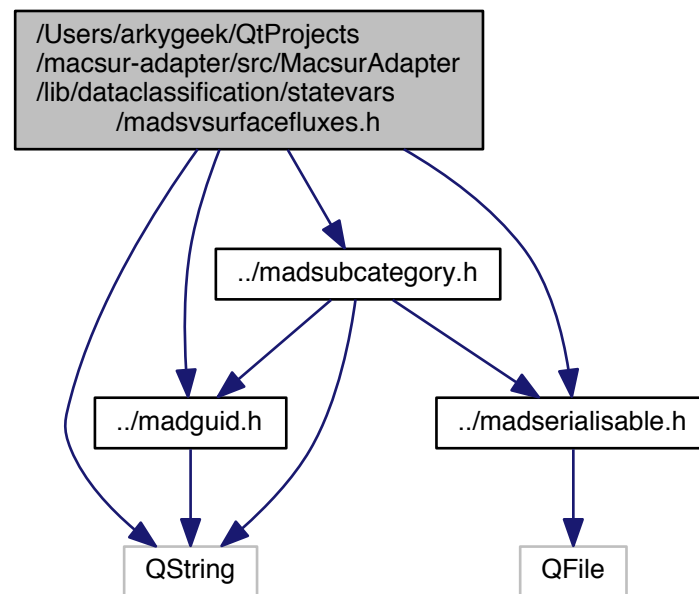


### 7.30 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/dataclassification/statevars/madsvsurfacefluxes.h

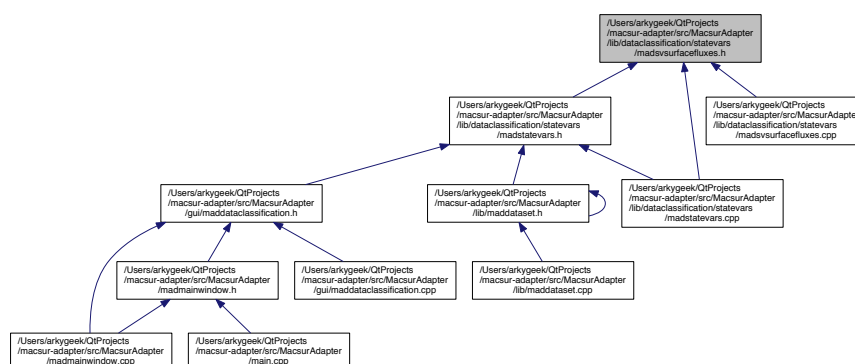
#### File Reference

```
#include "../madsubcategory.h"
#include "../../madguid.h"
#include "../../madserialisable.h"
#include <QString>
```

Include dependency graph for madsvsurfacefluxes.h:



This graph shows which files directly or indirectly include this file:



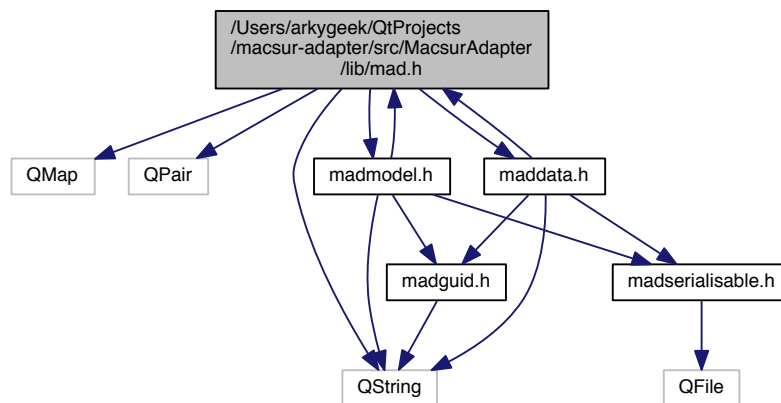
## Classes

- class [MadSVSurfaceFluxes](#)

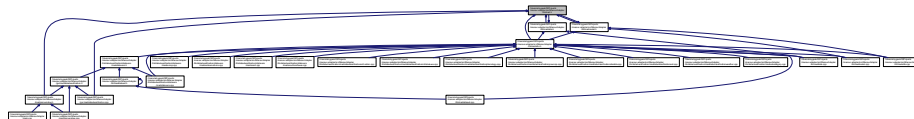
The *MadSVSurfaceFluxes* class.

## 7.31 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/mad.h File Reference

```
#include <QMap>
#include <QPair>
#include <QString>
#include "madmodel.h"
#include "maddata.h"
Include dependency graph for mad.h:
```



This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef QMap< QString, QPair< bool, QString > > [MadTripleMap](#)  
*MadTripleMap.*
- typedef QPair< QPair< QString, QString >, QPair< QString, QString > > [MadModelInfo](#)  
*MadModelInfo.*

### Enumerations

- enum [ModelTheme](#) { [CropM](#), [LiveM](#), [TradeM](#) }  
*MadModelMap.*
- enum [Scale](#) { [Farm](#), [Locality](#), [Regional](#), [National](#), [International](#), [Global](#) }



*The Scale enum.*

- enum [Nuts](#) { [Nuts1](#), [Nuts2](#), [Nuts3](#) }

*The Nuts enum.*

- enum [AreaUnits](#) {  
[Dunum](#), [Hectare](#), [Acre](#), [SquareKm](#),  
[SquareMile](#) }

*The AreaUnits enum.*

- enum [FileType](#) { [CSV](#), [TAB](#), [OtherDelimited](#), [Binary](#) }

*The FileType enum.*

- enum [EnergyType](#) { [KCalories](#), [TDN](#) }

*The EnergyType enum.*

- enum [DataClass](#) { [Platinum](#), [Gold](#), [Silver](#), [Bronze](#) }

*The DataClass enum.*

### 7.31.1 Typedef Documentation

#### 7.31.1.1 typedef QPair<QPair<QString,QString>, QPair<QString,QString> > MadModelInfo

MadModelInfo.

Definition at line 51 of file mad.h.

#### 7.31.1.2 typedef QMap<QString,QPair<bool,QString> > MadTripleMap

MadTripleMap.

Definition at line 47 of file mad.h.

### 7.31.2 Enumeration Type Documentation

#### 7.31.2.1 enum AreaUnits

The AreaUnits enum.

Enumerator

***Dunum***

***Hectare***

***Acre***

***SquareKm***

***SquareMile***

Definition at line 72 of file mad.h.

```
72 {Dunum, Hectare, Acre, SquareKm, SquareMile};
```

#### 7.31.2.2 enum DataClass

The DataClass enum.

Enumerator

***Platinum***

***Gold***

***Silver***

***Bronze***

Definition at line 84 of file mad.h.

```
84 {Platinum, Gold, Silver, Bronze};
```

### 7.31.2.3 enum EnergyType

The EnergyType enum.

Enumerator

***KCalories***

***TDN***

Definition at line 80 of file mad.h.

```
80 {KCalories, TDN};
```

### 7.31.2.4 enum FileType

The FileType enum.

Enumerator

***CSV***

***TAB***

***OtherDelimited***

***Binary***

Definition at line 76 of file mad.h.

```
76 {CSV, TAB, OtherDelimited, Binary};
```

### 7.31.2.5 enum ModelTheme

MadModelMap.

The ModelTheme enum

Enumerator

***CropM***

***LiveM***

***TradeM***

Definition at line 60 of file mad.h.

```
60 {CropM, LiveM, TradeM};
```

## 7.31.2.6 enum Nuts

The Nuts enum.

Enumerator

**Nuts1**

**Nuts2**

**Nuts3**

Definition at line 68 of file mad.h.

```
68 {Nuts1, Nuts2, Nuts3};
```

## 7.31.2.7 enum Scale

The Scale enum.

Enumerator

**Farm**

**Locality**

**Regional**

**National**

**International**

**Global**

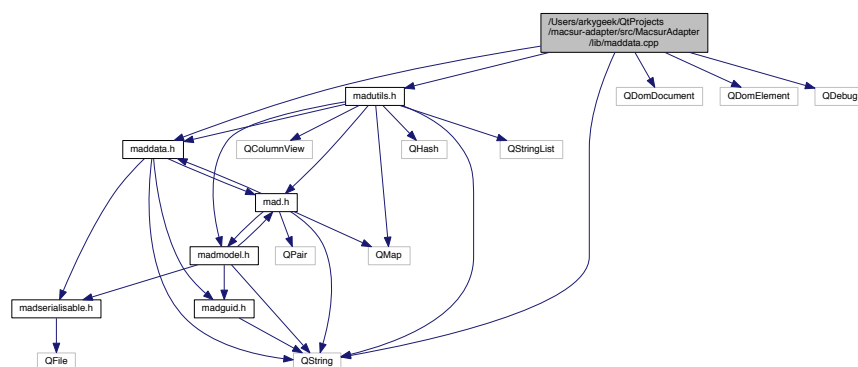
Definition at line 64 of file mad.h.

```
64 {Farm, Locality, Regional, National, International,
    Global};
```

## 7.32 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.cpp File Reference

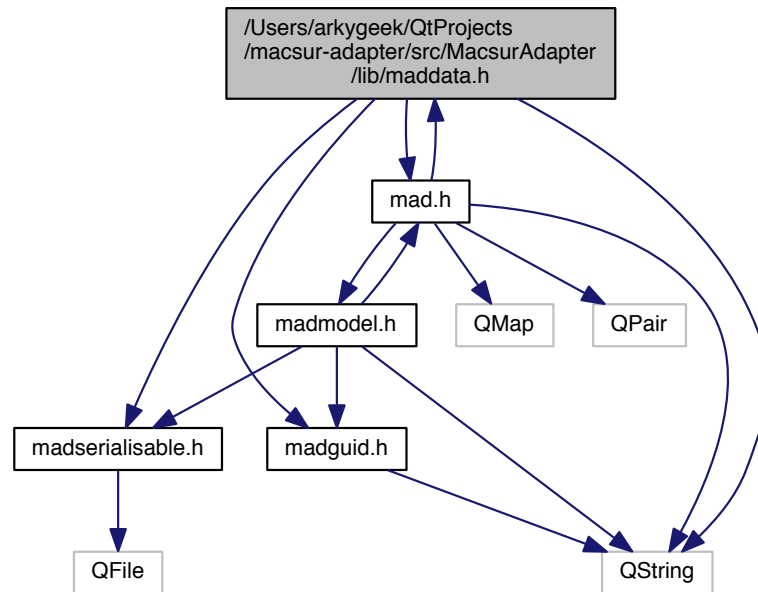
```
#include "maddata.h"
#include "madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

Include dependency graph for maddata.cpp:

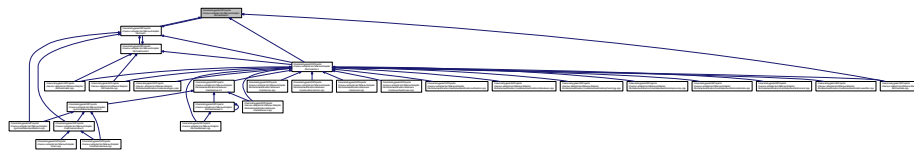


### 7.33 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddata.h File Reference

```
#include "madserialisable.h"
#include "madguid.h"
#include "mad.h"
#include <QString>
Include dependency graph for maddata.h:
```



This graph shows which files directly or indirectly include this file:



#### Classes

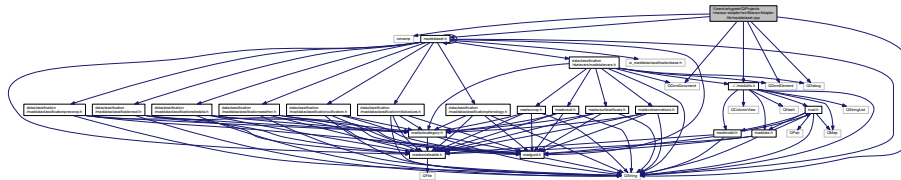
- class [MadData](#)

The [MadData](#) class.

### 7.34 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.cpp File Reference

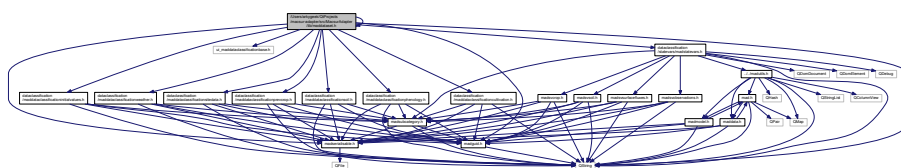
```
#include <iomanip>
```

```
#include "maddataset.h"
#include "madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
Include dependency graph for maddataset.cpp:
```

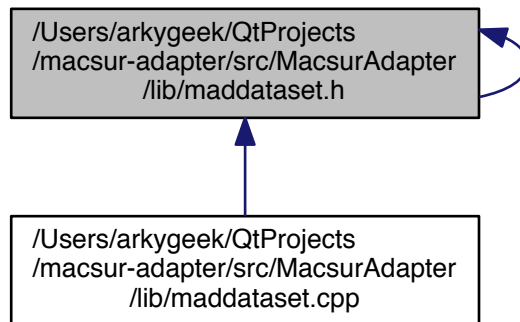


## 7.35 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/maddataset.h File Reference

```
#include "madguid.h"
#include "madserialisable.h"
#include "ui_maddataclassificationbase.h"
#include "maddataset.h"
#include "dataclassification/maddataclassificationcultivation.h"
#include "dataclassification/maddataclassificationinitialvalues.h"
#include "dataclassification/maddataclassificationphenology.h"
#include "dataclassification/maddataclassificationprevcrop.h"
#include "dataclassification/maddataclassificationssitedata.h"
#include "dataclassification/maddataclassificationsoil.h"
#include "dataclassification/maddataclassificationweather.h"
#include "dataclassification/statevars/madstatevars.h"
#include <QString>
Include dependency graph for maddataset.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

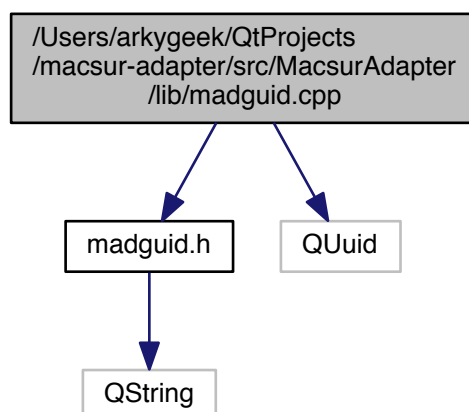
- class [MadDataset](#)

## 7.36 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.cpp File Reference

```
#include "madguid.h"
```

```
#include <QUuid>
```

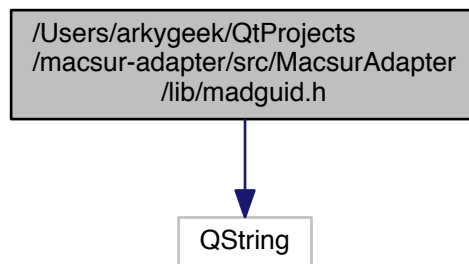
Include dependency graph for madguid.cpp:



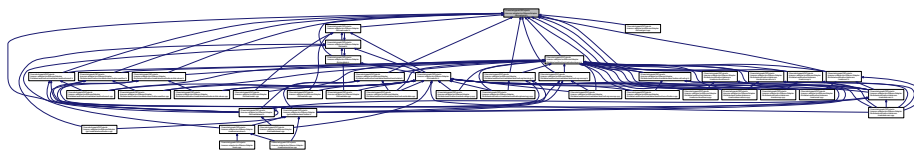
## 7.37 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madguid.h File Reference

```
#include <QString>
```

Include dependency graph for madguid.h:



This graph shows which files directly or indirectly include this file:



### Classes

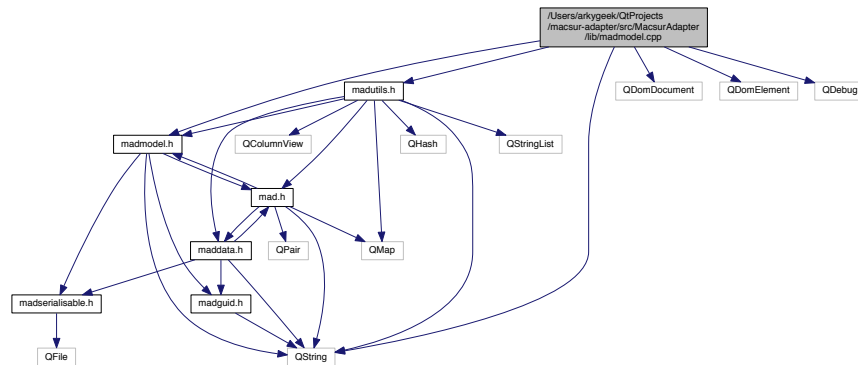
- class [MadGuid](#)

The [MadGuid](#) class An abstract base class that has a Globally Unique Identifier (GUID) to represent a unique instance.

## 7.38 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.cpp File Reference

```
#include "madmodel.h"
#include "madutils.h"
#include <QString>
#include <QDomDocument>
#include <QDomElement>
#include <QDebug>
```

Include dependency graph for madmodel.cpp:



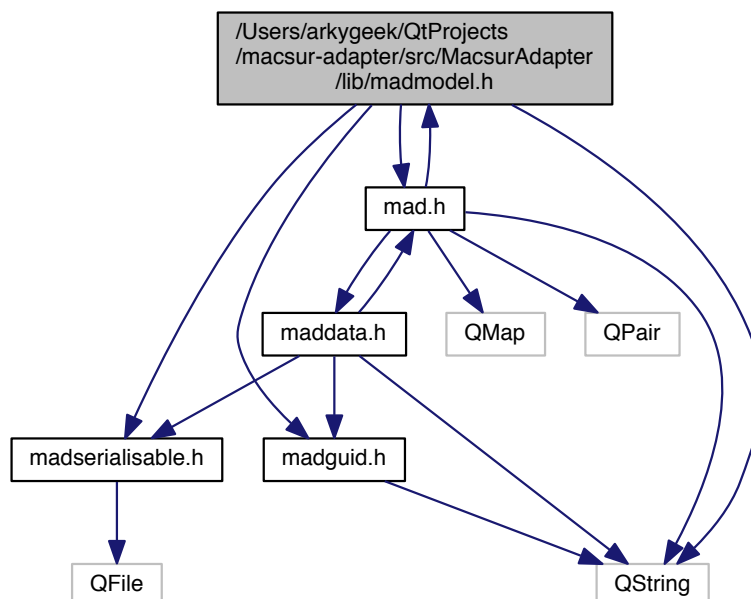
### 7.39 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madmodel.h File Reference

```

#include "madserialisable.h"
#include "madguid.h"
#include "mad.h"
#include <QString>

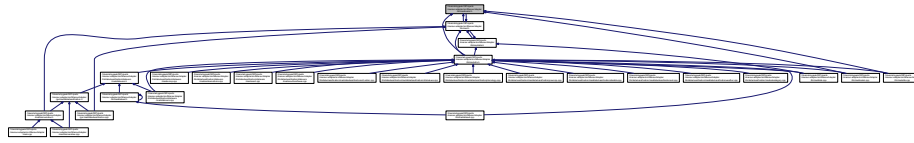
```

Include dependency graph for madmodel.h:





This graph shows which files directly or indirectly include this file:



## Classes

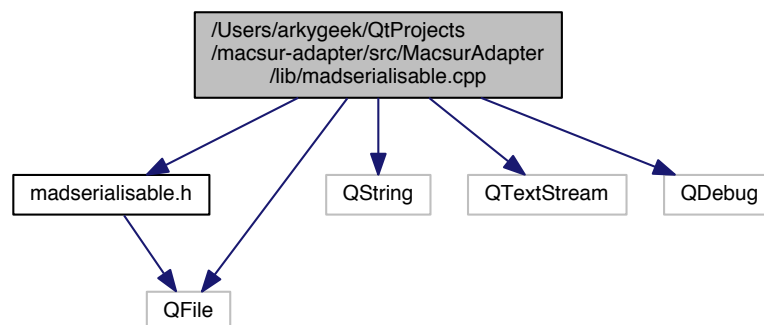
- class [MadModel](#)

The [MadModel](#) class, to represent a *ModelTheme*.

## 7.40 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.cpp File Reference

```
#include "madserialisable.h"  
#include <QFile>  
#include <QString>  
#include <QTextStream>  
#include <QDebug>
```

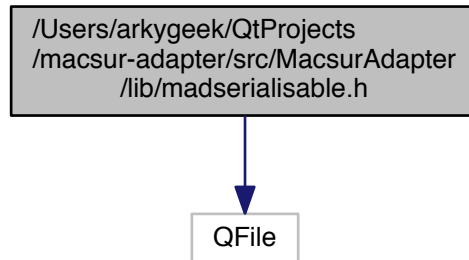
Include dependency graph for madserialisable.cpp:



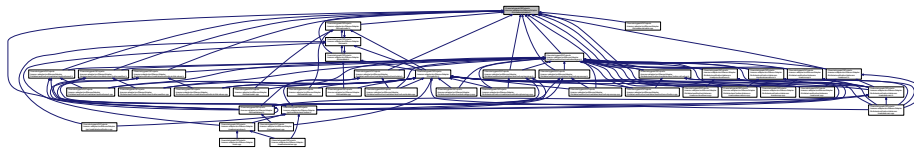
## 7.41 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madserialisable.h File Reference

```
#include <QFile>
```

Include dependency graph for madserialisable.h:



This graph shows which files directly or indirectly include this file:



## Classes

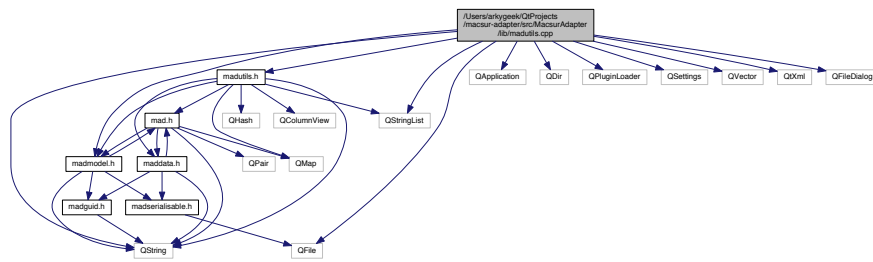
- class [MadSerialisable](#)

## 7.42 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.cpp File Reference

```

#include "madutils.h"
#include "madmodel.h"
#include <QApplication>
#include <QDir>
#include <QFile>
#include <QPluginLoader>
#include <QSettings>
#include <QString>
#include <QStringList>
#include <QVector>
#include <QtXml>
#include <QFileDialog>
  
```

Include dependency graph for madutils.cpp:



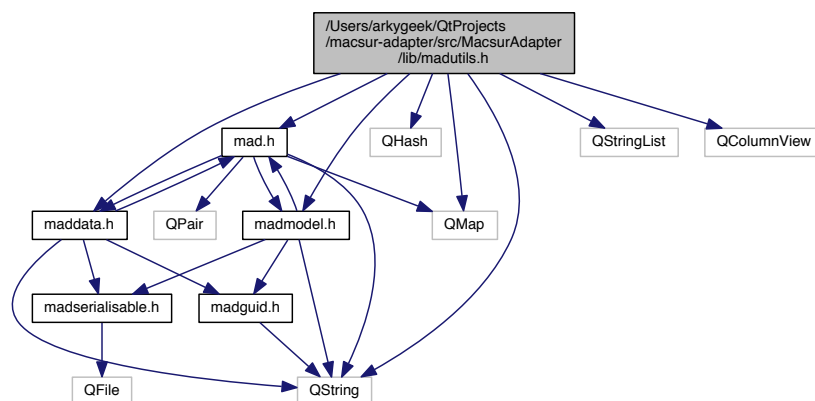
## 7.43 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madutils.h File Reference

```

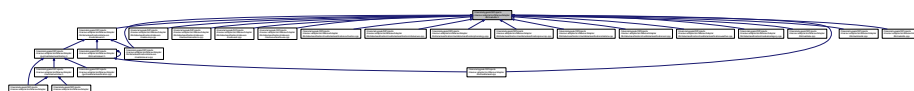
#include "mad.h"
#include "madmodel.h"
#include "maddata.h"
#include <QHash>
#include <QMap>
#include <QString>
#include <QStringList>
#include <QColumnView>

```

Include dependency graph for madutils.h:



This graph shows which files directly or indirectly include this file:

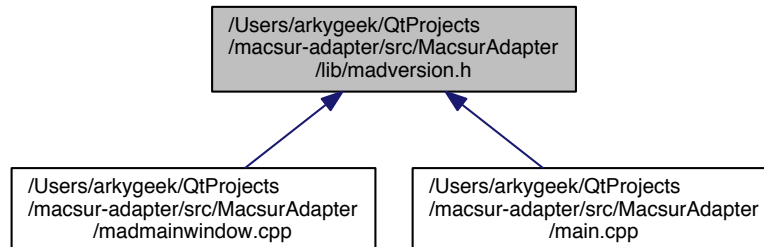


## Classes

- class [MadUtils](#)

## 7.44 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/lib/madversion.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- `#define VERSION "0.1"`

#### 7.44.1 Macro Definition Documentation

##### 7.44.1.1 `#define VERSION "0.1"`

Definition at line 23 of file `madversion.h`.

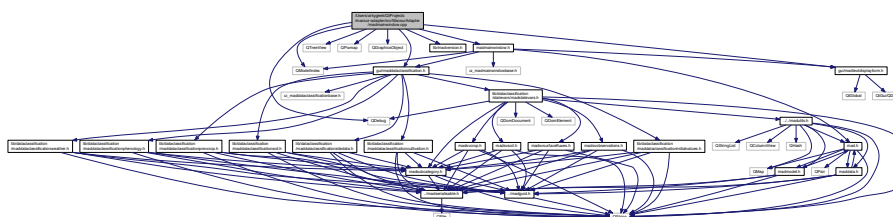
## 7.45 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.cpp File Reference

```

#include <QModelIndex>
#include <QDebug>
#include <QTreeView>
#include <QPixmap>
#include <QGraphicsObject>
#include "madmainwindow.h"
#include "lib/madversion.h"
#include "gui/maddataclassification.h"
#include "gui/madtextdisplayform.h"

```

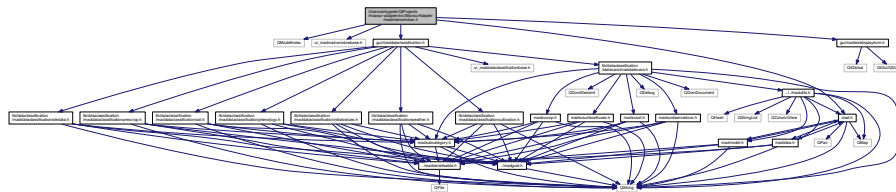
Include dependency graph for `madmainwindow.cpp`:



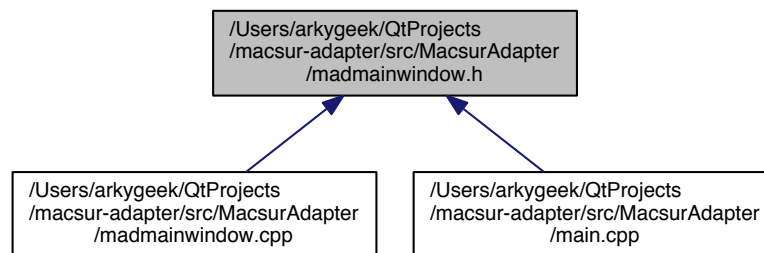
## 7.46 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/madmainwindow.h File Reference

```
#include <QModelIndex>
#include "ui_madmainwindowbase.h"
#include "gui/maddataclassification.h"
#include "gui/madtextdisplayform.h"
#include "lib/mad.h"
```

Include dependency graph for madmainwindow.h:



This graph shows which files directly or indirectly include this file:



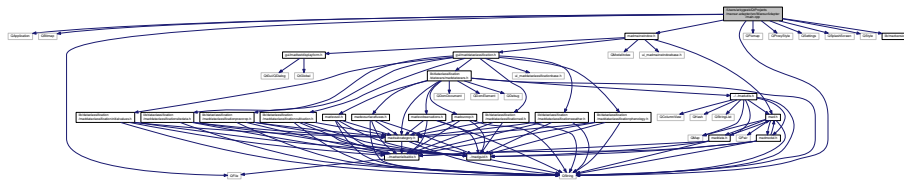
### Classes

- class [MadMainWindow](#)

## 7.47 /Users/arkygeek/QtProjects/macsur-adapter/src/MacsurAdapter/main.cpp File Reference

```
#include <QApplication>
#include <QBitmap>
#include <QFile>
#include <QPixmap>
#include <QProxyStyle>
#include <QSettings>
#include <QSplashScreen>
#include <QString>
#include <QStyle>
#include "madmainwindow.h"
#include "lib/madversion.h"
```

Include dependency graph for main.cpp:



## Functions

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

### 7.47.1 Function Documentation

#### 7.47.1.1 bool bundleclicked ( int *argc*, char \* *argv*[] )

Definition at line 77 of file main.cpp.

```
78 {
79     return ( argc > 1 && memcmp(argv[1], "-psn_", 5) == 0 );
80 }
```

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

Definition at line 47 of file main.cpp.

```
48 {
49     QApplication a(argc, argv);
50
51     #ifdef Q_WS_WIN
52         //for windows lets use plastique syle!
53         QApplication::setStyle(new QPlastiqueStyle);
54     #endif
55
56     #ifdef Q_OS_MACX
57         QString bundledQtCore(QCoreApplication::applicationDirPath().append
58                               ("/lib/QtCore.framework"));
59         if (QFile::exists(bundledQtCore))
60         {
61             QCoreApplication::setLibraryPaths
62                 (QStringList(QCoreApplication::applicationDirPath()));
63         }
64     #endif
65
66     MadMainWindow w;
67     w.show();
68
69     return a.exec();
70 }
```

# Index

~MadTextDisplayForm  
    MadTextDisplayForm, [248](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/gui/maddataclassification.-  
        cpp, [261](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/gui/maddataclassification.h, [261](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/gui/madtextdisplayform.cpp, [262](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/gui/madtextdisplayform.h, [263](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [264](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [265](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [266](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [266](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [267](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [268](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [269](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [269](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [270](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [271](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [272](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [272](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        cpp, [273](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/maddataclassificationinitialvalues.-  
        h, [274](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/madsubcategory.-  
        cpp, [275](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/madsubcategory.-  
        h, [275](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madstatevars.-  
        cpp, [276](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madstatevars.-  
        h, [277](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvcrop.-  
        cpp, [278](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvcrop.-  
        h, [279](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvobservation.-  
        cpp, [280](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvobservation.-  
        h, [280](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvsoil.-  
        cpp, [281](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvsoil.-  
        h, [282](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvsurfaceflux.-  
        cpp, [283](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/dataclassification/statevars/madsvsurfaceflux.-  
        h, [284](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/mad.h, [286](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/maddata.cpp, [289](#)  
/Users/arkygeek/QtProjects/macsur-adapter/src/-  
    MacsurAdapter/lib/maddata.h, [290](#)

- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/maddataset.cpp, 290
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/maddataset.h, 291
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madguid.cpp, 292
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madguid.h, 293
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madmodel.cpp, 293
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madmodel.h, 294
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madserialisable.cpp, 295
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madserialisable.h, 295
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madutils.cpp, 296
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madutils.h, 297
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/lib/madversion.h, 298
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/madmainwindow.cpp, 298
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/madmainwindow.h, 299
- /Users/arkygeek/QtProjects/macsur-adapter/src/-  
  MacsurAdapter/main.cpp, 299
- Acre
  - mad.h, 287
- agrBiomass
  - MadSVCrop, 194
- altitude
  - MadDataClassificationSiteData, 84
- AreaUnits
  - mad.h, 287
- Binary
  - mad.h, 288
- Bronze
  - mad.h, 288
- bulkDensity
  - MadDataClassificationSoil, 96
- bundleclicked
  - main.cpp, 300
- CSV
  - mad.h, 288
- carbonOrganic
  - MadDataClassificationSoil, 97
- ch4Loss
  - MadSVSurfaceFluxes, 235
- changeEvent
  - MadDataClassification, 27
  - MadMainWindow, 150
- createTextFile
  - MadUtils, 250
- crop
  - MadDataClassificationPrevCrop, 69
- CropM
  - mad.h, 288
- cropCategories
  - MadStateVars, 169
- cultivation
  - MadDataset, 132
- damages
  - MadSVObservations, 208
- DataClass
  - mad.h, 287
- depth
  - MadSubCategory, 181
- description
  - MadData, 13
  - MadDataset, 133
  - MadModel, 153
- Dunum
  - mad.h, 287
- earEmergence
  - MadDataClassificationPhenology, 56
- emergence
  - MadDataClassificationPhenology, 57
- EnergyType
  - mad.h, 288
- et
  - MadSVSurfaceFluxes, 236
- Farm
  - mad.h, 289
- fertilisation
  - MadDataClassificationCultivation, 30
  - MadDataClassificationPrevCrop, 70
- fieldCapacityMeas
  - MadDataClassificationSoil, 97
- FileType
  - mad.h, 288
- flowering
  - MadDataClassificationPhenology, 57
- fromXml
  - MadData, 14
  - MadDataClassificationCultivation, 31
  - MadDataClassificationInitialValues, 45
  - MadDataClassificationPhenology, 58
  - MadDataClassificationPrevCrop, 70
  - MadDataClassificationSiteData, 84
  - MadDataClassificationSoil, 98
  - MadDataClassificationWeather, 114
  - MadDataset, 133
  - MadModel, 153
  - MadSerialisable, 163
  - MadStateVars, 170
  - MadSubCategory, 181
  - MadSVCrop, 194
  - MadSVObservations, 209
  - MadSVSoil, 220
  - MadSVSurfaceFluxes, 236



- fromXmlFile
  - MadData, [15](#)
  - MadDataClassificationCultivation, [31](#)
  - MadDataClassificationInitialValues, [46](#)
  - MadDataClassificationPhenology, [58](#)
  - MadDataClassificationPrevCrop, [71](#)
  - MadDataClassificationSiteData, [85](#)
  - MadDataClassificationSoil, [98](#)
  - MadDataClassificationWeather, [114](#)
  - MadDataset, [134](#)
  - MadModel, [154](#)
  - MadSerialisable, [164](#)
  - MadStateVars, [170](#)
  - MadSubCategory, [182](#)
  - MadSVCrop, [195](#)
  - MadSVObservations, [209](#)
  - MadSVSoil, [221](#)
  - MadSVSurfaceFluxes, [237](#)
- getAvailableModels
  - MadUtils, [251](#)
- getModel
  - MadUtils, [252](#)
- getModelOutputDir
  - MadUtils, [252](#)
- getStandardCss
  - MadUtils, [252](#)
- Global
  - mad.h, [289](#)
- globalRadiation
  - MadDataClassificationWeather, [115](#)
- Gold
  - mad.h, [287](#)
- guid
  - MadData, [15](#)
  - MadDataClassificationCultivation, [32](#)
  - MadDataClassificationInitialValues, [47](#)
  - MadDataClassificationPhenology, [59](#)
  - MadDataClassificationPrevCrop, [72](#)
  - MadDataClassificationSiteData, [86](#)
  - MadDataClassificationSoil, [99](#)
  - MadDataClassificationWeather, [116](#)
  - MadDataset, [135](#)
  - MadGuid, [147](#)
  - MadModel, [155](#)
  - MadStateVars, [171](#)
  - MadSubCategory, [183](#)
  - MadSVCrop, [196](#)
  - MadSVObservations, [210](#)
  - MadSVSoil, [222](#)
  - MadSVSurfaceFluxes, [238](#)
- harvest
  - MadDataClassificationCultivation, [32](#)
- harvestDate
  - MadDataClassificationPrevCrop, [72](#)
- Hectare
  - mad.h, [287](#)
- hydrCondCurve
  - MadDataClassificationSoil, [99](#)
- imageFile
  - MadData, [16](#)
  - MadModel, [155](#)
- initialValues
  - MadDataset, [135](#)
- International
  - mad.h, [289](#)
- irrigation
  - MadDataClassificationCultivation, [33](#)
  - MadDataClassificationPrevCrop, [72](#)
- KCalories
  - mad.h, [288](#)
- lai
  - MadSVCrop, [196](#)
- latitude
  - MadDataClassificationSiteData, [86](#)
- leafWetness
  - MadDataClassificationWeather, [116](#)
- LiveM
  - mad.h, [288](#)
- Locality
  - mad.h, [289](#)
- lodging
  - MadSVObservations, [210](#)
- longitude
  - MadDataClassificationSiteData, [87](#)
- mad.h
  - Acre, [287](#)
  - Binary, [288](#)
  - Bronze, [288](#)
  - CSV, [288](#)
  - CropM, [288](#)
  - Dunum, [287](#)
  - Farm, [289](#)
  - Global, [289](#)
  - Gold, [287](#)
  - Hectare, [287](#)
  - International, [289](#)
  - KCalories, [288](#)
  - LiveM, [288](#)
  - Locality, [289](#)
  - National, [289](#)
  - Nuts1, [289](#)
  - Nuts2, [289](#)
  - Nuts3, [289](#)
  - OtherDelimited, [288](#)
  - Platinum, [287](#)
  - Regional, [289](#)
  - Silver, [288](#)
  - SquareKm, [287](#)
  - SquareMile, [287](#)
  - TAB, [288](#)
  - TDN, [288](#)
  - TradeM, [288](#)

- mad.h
  - AreaUnits, [287](#)
  - DataClass, [287](#)
  - EnergyType, [288](#)
  - FileType, [288](#)
  - MadModelInfo, [287](#)
  - MadTripleMap, [287](#)
  - ModelTheme, [288](#)
  - Nuts, [288](#)
  - Scale, [289](#)
- MadData, [11](#)
  - description, [13](#)
  - fromXml, [14](#)
  - fromXmlFile, [15](#)
  - guid, [15](#)
  - imageFile, [16](#)
  - MadData, [12](#), [13](#)
  - MadData, [12](#), [13](#)
  - name, [16](#)
  - operator=, [17](#)
  - setDescription, [17](#)
  - setGuid, [18](#)
  - setImageFile, [18](#)
  - setName, [19](#)
  - toHtml, [19](#)
  - toText, [19](#)
  - toXml, [20](#)
  - toXmlFile, [20](#)
- MadDataClassification, [21](#)
  - changeEvent, [27](#)
  - MadDataClassification, [22](#)
  - MadDataClassification, [22](#)
- MadDataClassificationCultivation, [27](#)
  - fertilisation, [30](#)
  - fromXml, [31](#)
  - fromXmlFile, [31](#)
  - guid, [32](#)
  - harvest, [32](#)
  - irrigation, [33](#)
  - MadDataClassificationCultivation, [29](#)
  - MadDataClassificationCultivation, [29](#)
  - operator=, [33](#)
  - seedDensity, [34](#)
  - setFertilisation, [35](#)
  - setGuid, [35](#)
  - setHarvest, [36](#)
  - setIrrigation, [36](#)
  - setSeedDensity, [37](#)
  - setSowing, [37](#)
  - setTillage, [37](#)
  - setVariety, [37](#)
  - setYield, [38](#)
  - sowing, [38](#)
  - tillage, [39](#)
  - toHtml, [39](#)
  - toText, [40](#)
  - toXml, [40](#)
  - toXmlFile, [41](#)
  - variety, [42](#)
  - yield, [42](#)
- MadDataClassificationInitialValues, [43](#)
  - fromXml, [45](#)
  - fromXmlFile, [46](#)
  - guid, [47](#)
  - MadDataClassificationInitialValues, [44](#), [45](#)
  - MadDataClassificationInitialValues, [44](#), [45](#)
  - nitrogenMin, [47](#)
  - operator=, [48](#)
  - setGuid, [48](#)
  - setNitrogenMin, [49](#)
  - setSoilMoisture, [49](#)
  - soilMoisture, [50](#)
  - toHtml, [50](#)
  - toText, [50](#)
  - toXml, [51](#)
  - toXmlFile, [52](#)
- MadDataClassificationPhenology, [53](#)
  - earEmergence, [56](#)
  - emergence, [57](#)
  - flowering, [57](#)
  - fromXml, [58](#)
  - fromXmlFile, [58](#)
  - guid, [59](#)
  - MadDataClassificationPhenology, [55](#)
  - MadDataClassificationPhenology, [55](#)
  - operator=, [59](#)
  - setEarEmergence, [60](#)
  - setEmergence, [60](#)
  - setFlowering, [61](#)
  - setGuid, [61](#)
  - setStemElongation, [62](#)
  - setYellowRipeness, [62](#)
  - stemElongation, [63](#)
  - toHtml, [63](#)
  - toText, [63](#)
  - toXml, [64](#)
  - toXmlFile, [65](#)
  - yellowRipeness, [65](#)
- MadDataClassificationPrevCrop, [66](#)
  - crop, [69](#)
  - fertilisation, [70](#)
  - fromXml, [70](#)
  - fromXmlFile, [71](#)
  - guid, [72](#)
  - harvestDate, [72](#)
  - irrigation, [72](#)
  - MadDataClassificationPrevCrop, [68](#)
  - MadDataClassificationPrevCrop, [68](#)
  - operator=, [73](#)
  - residueMgmt, [74](#)
  - setCrop, [75](#)
  - setFertilisation, [75](#)
  - setGuid, [75](#)
  - setHarvestDate, [76](#)
  - setIrrigation, [76](#)
  - setResidueMgmt, [77](#)

- setSowingDate, 77
- setYield, 77
- sowingDate, 78
- toHtml, 78
- toText, 79
- toXml, 79
- toXmlFile, 80
- yield, 81
- MadDataClassificationSiteData, 82
  - altitude, 84
  - fromXml, 84
  - fromXmlFile, 85
  - guid, 86
  - latitude, 86
  - longitude, 87
  - MadDataClassificationSiteData, 83
  - MadDataClassificationSiteData, 83
  - operator=, 87
  - setAltitude, 88
  - setGuid, 88
  - setLatitude, 89
  - setLongitude, 89
  - toHtml, 90
  - toText, 90
  - toXml, 91
  - toXmlFile, 92
- MadDataClassificationSoil, 93
  - bulkDensity, 96
  - carbonOrganic, 97
  - fieldCapacityMeas, 97
  - fromXml, 98
  - fromXmlFile, 98
  - guid, 99
  - hydrCondCurve, 99
  - MadDataClassificationSoil, 94
  - MadDataClassificationSoil, 94
  - nitrogenOrganic, 100
  - operator=, 100
  - pH, 102
  - pfCurve, 101
  - setBulkDensity, 102
  - setCarbonOrganic, 103
  - setFieldCapacityMeas, 103
  - setGuid, 103
  - setHydrCondCurve, 104
  - setNitrogenOrganic, 104
  - setPfCurve, 105
  - setPh, 105
  - setTexture, 106
  - setWiltingPointMeas, 106
  - texture, 106
  - toHtml, 107
  - toText, 107
  - toXml, 108
  - toXmlFile, 109
  - wiltingPointMeas, 110
- MadDataClassificationWeather, 110
  - fromXml, 114
  - fromXmlFile, 114
  - globalRadiation, 115
  - guid, 116
  - leafWetness, 116
  - MadDataClassificationWeather, 112
  - MadDataClassificationWeather, 112
  - minData, 116
  - operator=, 117
  - precipitation, 118
  - relativeHumidity, 119
  - setGlobalRadiation, 119
  - setGuid, 119
  - setLeafWetness, 120
  - setMinData, 120
  - setPrecipitation, 121
  - setRelativeHumidity, 121
  - setSoilTemp, 121
  - setSunshineHours, 122
  - setTAve, 122
  - setTMax, 122
  - setTMin, 123
  - setWindSpeed, 123
  - soilTemp, 123
  - sunshineHours, 124
  - tAve, 124
  - tMax, 125
  - tMin, 125
  - toHtml, 126
  - toText, 126
  - toXml, 127
  - toXmlFile, 128
  - windSpeed, 129
- MadDataset, 129
  - cultivation, 132
  - description, 133
  - fromXml, 133
  - fromXmlFile, 134
  - guid, 135
  - initialValues, 135
  - MadDataset, 131
  - MadDataset, 131
  - name, 135
  - operator=, 136
  - phenology, 137
  - prevCrop, 138
  - setCultivation, 138
  - setDescription, 138
  - setGuid, 138
  - setInitialValues, 139
  - setName, 139
  - setPhenology, 139
  - setPrevCrop, 140
  - setSiteData, 140
  - setSoil, 140
  - setStateVars, 140
  - setWeather, 140
  - siteData, 140
  - soil, 141

- stateVars, 141
- toHtml, 142
- toText, 142
- toXml, 143
- toXmlFile, 144
- weather, 145
- MadGuid, 145
  - guid, 147
  - MadGuid, 147
  - MadGuid, 147
  - setGuid, 147
- MadMainWindow, 148
  - changeEvent, 150
  - MadMainWindow, 149
  - MadMainWindow, 149
  - modelText, 150
  - setModelText, 150
- MadModel, 150
  - description, 153
  - fromXml, 153
  - fromXmlFile, 154
  - guid, 155
  - imageFile, 155
  - MadModel, 152
  - MadModel, 152
  - name, 156
  - operator=, 156
  - setDescription, 157
  - setGuid, 157
  - setImageFile, 158
  - setName, 158
  - toHtml, 158
  - toText, 159
  - toXml, 160
  - toXmlFile, 160
- MadModelInfo
  - mad.h, 287
- MadSVCrop, 190
  - agrBiomass, 194
  - fromXml, 194
  - fromXmlFile, 195
  - guid, 196
  - lai, 196
  - MadSVCrop, 192
  - MadSVCrop, 192
  - nInAGrBiomass, 196
  - nInOrgans, 197
  - operator=, 197
  - rootBiomass, 198
  - setAgrBiomass, 199
  - setGuid, 199
  - setLai, 200
  - setNInAGrBiomass, 200
  - setNInOrgans, 201
  - setRootBiomass, 201
  - setWeightOrgans, 202
  - toHtml, 202
  - toText, 203
  - toXml, 203
  - toXmlFile, 204
  - weightOrgans, 205
- MadSVObservations, 206
  - damages, 208
  - fromXml, 209
  - fromXmlFile, 209
  - guid, 210
  - lodging, 210
  - MadSVObservations, 207
  - MadSVObservations, 207
  - operator=, 211
  - pestsOrDiseases, 212
  - setDamages, 212
  - setGuid, 213
  - setLodging, 213
  - setPestsOrDiseases, 214
  - toHtml, 214
  - toText, 215
  - toXml, 215
  - toXmlFile, 216
- MadSVSoil, 217
  - fromXml, 220
  - fromXmlFile, 221
  - guid, 222
  - MadSVSoil, 219
  - MadSVSoil, 219
  - nMin, 222
  - nitrogenFluxBottomRoot, 222
  - operator=, 223
  - pressureHeads, 224
  - setGuid, 224
  - setNMin, 225
  - setNitrogenFluxBottomRoot, 225
  - setPressureHeads, 226
  - setSoilWaterGrav, 226
  - setSoilWaterSensorCal, 226
  - setWaterFluxBottomRoot, 227
  - soilWaterGrav, 227
  - soilWaterSensorCal, 228
  - toHtml, 228
  - toText, 229
  - toXml, 229
  - toXmlFile, 230
  - waterFluxBottomRoot, 231
- MadSVSurfaceFluxes, 231
  - ch4Loss, 235
  - et, 236
  - fromXml, 236
  - fromXmlFile, 237
  - guid, 238
  - MadSVSurfaceFluxes, 233
  - MadSVSurfaceFluxes, 233
  - n2Loss, 238
  - n2oLoss, 239
  - nh3Loss, 239
  - operator=, 240
  - setCh4Loss, 241

- setEt, [242](#)
  - setGuid, [242](#)
  - setN2Loss, [243](#)
  - setN2oLoss, [243](#)
  - setNh3Loss, [244](#)
  - toHtml, [244](#)
  - toText, [245](#)
  - toXml, [245](#)
  - toXmlFile, [246](#)
- MadSerialisable, [161](#)
  - fromXml, [163](#)
  - fromXmlFile, [164](#)
  - MadSerialisable, [163](#)
  - MadSerialisable, [163](#)
  - toXml, [165](#)
  - toXmlFile, [165](#)
- MadStateVars, [166](#)
  - cropCategories, [169](#)
  - fromXml, [170](#)
  - fromXmlFile, [170](#)
  - guid, [171](#)
  - MadStateVars, [168](#)
  - MadStateVars, [168](#)
  - observationCategories, [171](#)
  - operator=, [172](#)
  - setCropCategories, [172](#)
  - setGuid, [173](#)
  - setObservationCategories, [173](#)
  - setSoilCategories, [174](#)
  - setSurfaceFluxesCategories, [174](#)
  - soilCategories, [174](#)
  - surfaceFluxesCategories, [175](#)
  - toHtml, [175](#)
  - toText, [175](#)
  - toXml, [176](#)
  - toXmlFile, [177](#)
- MadSubCategory, [178](#)
  - depth, [181](#)
  - fromXml, [181](#)
  - fromXmlFile, [182](#)
  - guid, [183](#)
  - MadSubCategory, [180](#)
  - MadSubCategory, [180](#)
  - minData, [183](#)
  - observations, [184](#)
  - operator=, [184](#)
  - replicates, [185](#)
  - setDepth, [185](#)
  - setGuid, [186](#)
  - setMinData, [186](#)
  - setObservations, [187](#)
  - setReplicates, [187](#)
  - setWeightPoints, [187](#)
  - toHtml, [187](#)
  - toText, [187](#)
  - toXml, [188](#)
  - toXmlFile, [189](#)
  - weightPoints, [190](#)
- MadTextDisplayForm, [247](#)
  - ~MadTextDisplayForm, [248](#)
  - MadTextDisplayForm, [248](#)
  - MadTextDisplayForm, [248](#)
  - setText, [249](#)
- MadTripleMap
  - mad.h, [287](#)
- MadUtils, [249](#)
  - createTextFile, [250](#)
  - getAvailableModels, [251](#)
  - getModel, [252](#)
  - getModelOutputDir, [252](#)
  - getStandardCss, [252](#)
  - MadUtils, [250](#)
  - MadUtils, [250](#)
  - ModelMap, [250](#)
  - openGraphicFile, [253](#)
  - saveFile, [253](#)
  - sortList, [254](#)
  - uniqueList, [254](#)
  - userConversionTablesDirPath, [254](#)
  - userImagesDirPath, [255](#)
  - userModelParametersDirPath, [255](#)
  - userModelProfilesDirPath, [256](#)
  - userSettingsDirPath, [256](#)
  - xmlDecode, [257](#)
  - xmlEncode, [257](#)
- maddataclassification.cpp
  - makeString, [261](#)
- madversion.h
  - VERSION, [298](#)
- main
  - main.cpp, [300](#)
- main.cpp
  - bundleclicked, [300](#)
  - main, [300](#)
- makeString
  - maddataclassification.cpp, [261](#)
- minData
  - MadDataClassificationWeather, [116](#)
  - MadSubCategory, [183](#)
- ModelMap
  - MadUtils, [250](#)
- modelText
  - MadMainWindow, [150](#)
- ModelTheme
  - mad.h, [288](#)
- n2Loss
  - MadSVSurfaceFluxes, [238](#)
- n2oLoss
  - MadSVSurfaceFluxes, [239](#)
- nInAGrBiomass
  - MadSVCrop, [196](#)
- nInOrgans
  - MadSVCrop, [197](#)
- nMin
  - MadSVSoil, [222](#)
- name

- MadData, 16
- MadDataset, 135
- MadModel, 156
- National
  - mad.h, 289
- nh3Loss
  - MadSVSurfaceFluxes, 239
- nitrogenFluxBottomRoot
  - MadSVSoil, 222
- nitrogenMin
  - MadDataClassificationInitialValues, 47
- nitrogenOrganic
  - MadDataClassificationSoil, 100
- Nuts
  - mad.h, 288
- Nuts1
  - mad.h, 289
- Nuts2
  - mad.h, 289
- Nuts3
  - mad.h, 289
- observationCategories
  - MadStateVars, 171
- observations
  - MadSubCategory, 184
- openGraphicFile
  - MadUtils, 253
- operator=
  - MadData, 17
  - MadDataClassificationCultivation, 33
  - MadDataClassificationInitialValues, 48
  - MadDataClassificationPhenology, 59
  - MadDataClassificationPrevCrop, 73
  - MadDataClassificationSiteData, 87
  - MadDataClassificationSoil, 100
  - MadDataClassificationWeather, 117
  - MadDataset, 136
  - MadModel, 156
  - MadStateVars, 172
  - MadSubCategory, 184
  - MadSVCrop, 197
  - MadSVObservations, 211
  - MadSVSoil, 223
  - MadSVSurfaceFluxes, 240
- OtherDelimited
  - mad.h, 288
- pH
  - MadDataClassificationSoil, 102
- pestsOrDiseases
  - MadSVObservations, 212
- pfCurve
  - MadDataClassificationSoil, 101
- phenology
  - MadDataset, 137
- Platinum
  - mad.h, 287
- precipitation
  - MadDataClassificationWeather, 118
- pressureHeads
  - MadSVSoil, 224
- prevCrop
  - MadDataset, 138
- QDialog, 259
- QMainWindow, 259
- Regional
  - mad.h, 289
- relativeHumidity
  - MadDataClassificationWeather, 119
- replicates
  - MadSubCategory, 185
- residueMgmt
  - MadDataClassificationPrevCrop, 74
- rootBiomass
  - MadSVCrop, 198
- saveFile
  - MadUtils, 253
- Scale
  - mad.h, 289
- seedDensity
  - MadDataClassificationCultivation, 34
- setAgrBiomass
  - MadSVCrop, 199
- setAltitude
  - MadDataClassificationSiteData, 88
- setBulkDensity
  - MadDataClassificationSoil, 102
- setCarbonOrganic
  - MadDataClassificationSoil, 103
- setCh4Loss
  - MadSVSurfaceFluxes, 241
- setCrop
  - MadDataClassificationPrevCrop, 75
- setCropCategories
  - MadStateVars, 172
- setCultivation
  - MadDataset, 138
- setDamages
  - MadSVObservations, 212
- setDepth
  - MadSubCategory, 185
- setDescription
  - MadData, 17
  - MadDataset, 138
  - MadModel, 157
- setEarEmergence
  - MadDataClassificationPhenology, 60
- setEmergence
  - MadDataClassificationPhenology, 60
- setEt
  - MadSVSurfaceFluxes, 242
- setFertilisation
  - MadDataClassificationCultivation, 35
  - MadDataClassificationPrevCrop, 75

- setFieldCapacityMeas
  - MadDataClassificationSoil, [103](#)
- setFlowering
  - MadDataClassificationPhenology, [61](#)
- setGlobalRadiation
  - MadDataClassificationWeather, [119](#)
- setGuid
  - MadData, [18](#)
  - MadDataClassificationCultivation, [35](#)
  - MadDataClassificationInitialValues, [48](#)
  - MadDataClassificationPhenology, [61](#)
  - MadDataClassificationPrevCrop, [75](#)
  - MadDataClassificationSiteData, [88](#)
  - MadDataClassificationSoil, [103](#)
  - MadDataClassificationWeather, [119](#)
  - MadDataset, [138](#)
  - MadGuid, [147](#)
  - MadModel, [157](#)
  - MadStateVars, [173](#)
  - MadSubCategory, [186](#)
  - MadSVCrop, [199](#)
  - MadSVObservations, [213](#)
  - MadSVSoil, [224](#)
  - MadSVSurfaceFluxes, [242](#)
- setHarvest
  - MadDataClassificationCultivation, [36](#)
- setHarvestDate
  - MadDataClassificationPrevCrop, [76](#)
- setHydrCondCurve
  - MadDataClassificationSoil, [104](#)
- setImageFile
  - MadData, [18](#)
  - MadModel, [158](#)
- setInitialValues
  - MadDataset, [139](#)
- setIrrigation
  - MadDataClassificationCultivation, [36](#)
  - MadDataClassificationPrevCrop, [76](#)
- setLai
  - MadSVCrop, [200](#)
- setLatitude
  - MadDataClassificationSiteData, [89](#)
- setLeafWetness
  - MadDataClassificationWeather, [120](#)
- setLodging
  - MadSVObservations, [213](#)
- setLongitude
  - MadDataClassificationSiteData, [89](#)
- setMinData
  - MadDataClassificationWeather, [120](#)
  - MadSubCategory, [186](#)
- setModelText
  - MadMainWindow, [150](#)
- setN2Loss
  - MadSVSurfaceFluxes, [243](#)
- setN2oLoss
  - MadSVSurfaceFluxes, [243](#)
- setNInAGrBiomass
  - MadSVCrop, [200](#)
- setNInOrgans
  - MadSVCrop, [201](#)
- setNMin
  - MadSVSoil, [225](#)
- setName
  - MadData, [19](#)
  - MadDataset, [139](#)
  - MadModel, [158](#)
- setNh3Loss
  - MadSVSurfaceFluxes, [244](#)
- setNitrogenFluxBottomRoot
  - MadSVSoil, [225](#)
- setNitrogenMin
  - MadDataClassificationInitialValues, [49](#)
- setNitrogenOrganic
  - MadDataClassificationSoil, [104](#)
- setObservationCategories
  - MadStateVars, [173](#)
- setObservations
  - MadSubCategory, [187](#)
- setPestsOrDiseases
  - MadSVObservations, [214](#)
- setPfCurve
  - MadDataClassificationSoil, [105](#)
- setPh
  - MadDataClassificationSoil, [105](#)
- setPhenology
  - MadDataset, [139](#)
- setPrecipitation
  - MadDataClassificationWeather, [121](#)
- setPressureHeads
  - MadSVSoil, [226](#)
- setPrevCrop
  - MadDataset, [140](#)
- setRelativeHumidity
  - MadDataClassificationWeather, [121](#)
- setReplicates
  - MadSubCategory, [187](#)
- setResidueMgmt
  - MadDataClassificationPrevCrop, [77](#)
- setRootBiomass
  - MadSVCrop, [201](#)
- setSeedDensity
  - MadDataClassificationCultivation, [37](#)
- setSiteData
  - MadDataset, [140](#)
- setSoil
  - MadDataset, [140](#)
- setSoilCategories
  - MadStateVars, [174](#)
- setSoilMoisture
  - MadDataClassificationInitialValues, [49](#)
- setSoilTemp
  - MadDataClassificationWeather, [121](#)
- setSoilWaterGrav
  - MadSVSoil, [226](#)
- setSoilWaterSensorCal



- MadSVSoil, 226
- setSowing
  - MadDataClassificationCultivation, 37
- setSowingDate
  - MadDataClassificationPrevCrop, 77
- setStateVars
  - MadDataset, 140
- setStemElongation
  - MadDataClassificationPhenology, 62
- setSunshineHours
  - MadDataClassificationWeather, 122
- setSurfaceFluxesCategories
  - MadStateVars, 174
- setTAve
  - MadDataClassificationWeather, 122
- setTMax
  - MadDataClassificationWeather, 122
- setTMin
  - MadDataClassificationWeather, 123
- setText
  - MadTextDisplayForm, 249
- setTexture
  - MadDataClassificationSoil, 106
- setTillage
  - MadDataClassificationCultivation, 37
- setVariety
  - MadDataClassificationCultivation, 37
- setWaterFluxBottomRoot
  - MadSVSoil, 227
- setWeather
  - MadDataset, 140
- setWeightOrgans
  - MadSVCrop, 202
- setWeightPoints
  - MadSubCategory, 187
- setWiltingPointMeas
  - MadDataClassificationSoil, 106
- setWindSpeed
  - MadDataClassificationWeather, 123
- setYellowRipeness
  - MadDataClassificationPhenology, 62
- setYield
  - MadDataClassificationCultivation, 38
  - MadDataClassificationPrevCrop, 77
- Silver
  - mad.h, 288
- siteData
  - MadDataset, 140
- soil
  - MadDataset, 141
- soilCategories
  - MadStateVars, 174
- soilMoisture
  - MadDataClassificationInitialValues, 50
- soilTemp
  - MadDataClassificationWeather, 123
- soilWaterGrav
  - MadSVSoil, 227
- soilWaterSensorCal
  - MadSVSoil, 228
- sortList
  - MadUtils, 254
- sowing
  - MadDataClassificationCultivation, 38
- sowingDate
  - MadDataClassificationPrevCrop, 78
- SquareKm
  - mad.h, 287
- SquareMile
  - mad.h, 287
- stateVars
  - MadDataset, 141
- stemElongation
  - MadDataClassificationPhenology, 63
- sunshineHours
  - MadDataClassificationWeather, 124
- surfaceFluxesCategories
  - MadStateVars, 175
- TAB
  - mad.h, 288
- TDN
  - mad.h, 288
- tAve
  - MadDataClassificationWeather, 124
- tMax
  - MadDataClassificationWeather, 125
- tMin
  - MadDataClassificationWeather, 125
- texture
  - MadDataClassificationSoil, 106
- tillage
  - MadDataClassificationCultivation, 39
- toHtml
  - MadData, 19
  - MadDataClassificationCultivation, 39
  - MadDataClassificationInitialValues, 50
  - MadDataClassificationPhenology, 63
  - MadDataClassificationPrevCrop, 78
  - MadDataClassificationSiteData, 90
  - MadDataClassificationSoil, 107
  - MadDataClassificationWeather, 126
  - MadDataset, 142
  - MadModel, 158
  - MadStateVars, 175
  - MadSubCategory, 187
  - MadSVCrop, 202
  - MadSVObservations, 214
  - MadSVSoil, 228
  - MadSVSurfaceFluxes, 244
- toText
  - MadData, 19
  - MadDataClassificationCultivation, 40
  - MadDataClassificationInitialValues, 50
  - MadDataClassificationPhenology, 63
  - MadDataClassificationPrevCrop, 79
  - MadDataClassificationSiteData, 90



- MadDataClassificationSoil, [107](#)
- MadDataClassificationWeather, [126](#)
- MadDataset, [142](#)
- MadModel, [159](#)
- MadStateVars, [175](#)
- MadSubCategory, [187](#)
- MadSVCrop, [203](#)
- MadSVObservations, [215](#)
- MadSVSoil, [229](#)
- MadSVSurfaceFluxes, [245](#)
- toXml
  - MadData, [20](#)
  - MadDataClassificationCultivation, [40](#)
  - MadDataClassificationInitialValues, [51](#)
  - MadDataClassificationPhenology, [64](#)
  - MadDataClassificationPrevCrop, [79](#)
  - MadDataClassificationSiteData, [91](#)
  - MadDataClassificationSoil, [108](#)
  - MadDataClassificationWeather, [127](#)
  - MadDataset, [143](#)
  - MadModel, [160](#)
  - MadSerialisable, [165](#)
  - MadStateVars, [176](#)
  - MadSubCategory, [188](#)
  - MadSVCrop, [203](#)
  - MadSVObservations, [215](#)
  - MadSVSoil, [229](#)
  - MadSVSurfaceFluxes, [245](#)
- toXmlFile
  - MadData, [20](#)
  - MadDataClassificationCultivation, [41](#)
  - MadDataClassificationInitialValues, [52](#)
  - MadDataClassificationPhenology, [65](#)
  - MadDataClassificationPrevCrop, [80](#)
  - MadDataClassificationSiteData, [92](#)
  - MadDataClassificationSoil, [109](#)
  - MadDataClassificationWeather, [128](#)
  - MadDataset, [144](#)
  - MadModel, [160](#)
  - MadSerialisable, [165](#)
  - MadStateVars, [177](#)
  - MadSubCategory, [189](#)
  - MadSVCrop, [204](#)
  - MadSVObservations, [216](#)
  - MadSVSoil, [230](#)
  - MadSVSurfaceFluxes, [246](#)
- TradeM
  - mad.h, [288](#)
- Ui, [9](#)
- uniqueList
  - MadUtils, [254](#)
- userConversionTablesDirPath
  - MadUtils, [254](#)
- userImagesDirPath
  - MadUtils, [255](#)
- userModelParametersDirPath
  - MadUtils, [255](#)
- userModelProfilesDirPath
  - MadUtils, [256](#)
- userSettingsDirPath
  - MadUtils, [256](#)
- VERSION
  - madversion.h, [298](#)
- variety
  - MadDataClassificationCultivation, [42](#)
- waterFluxBottomRoot
  - MadSVSoil, [231](#)
- weather
  - MadDataset, [145](#)
- weightOrgans
  - MadSVCrop, [205](#)
- weightPoints
  - MadSubCategory, [190](#)
- wiltingPointMeas
  - MadDataClassificationSoil, [110](#)
- windSpeed
  - MadDataClassificationWeather, [129](#)
- xmlDecode
  - MadUtils, [257](#)
- xmlEncode
  - MadUtils, [257](#)
- yellowRipeness
  - MadDataClassificationPhenology, [65](#)
- yield
  - MadDataClassificationCultivation, [42](#)
  - MadDataClassificationPrevCrop, [81](#)