

BRAT
4.2.0-beta

Generated by Doxygen 1.8.8

Fri Sep 1 2017 20:24:01

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	1
2.1	Namespace List	1
3	Hierarchical Index	1
3.1	Class Hierarchy	2
4	Class Index	5
4.1	Class List	5
5	File Index	8
5.1	File List	8
6	Module Documentation	11
6.1	Algorithms classes	12
6.1.1	Detailed Description	14
6.1.2	Function Documentation	14
6.2	Tools	18
6.2.1	Detailed Description	28
6.2.2	Macro Definition Documentation	28
6.2.3	Typedef Documentation	28
6.2.4	Function Documentation	28
6.2.5	Variable Documentation	46
6.3	Criteria	47
6.3.1	Detailed Description	57
6.3.2	Enumeration Type Documentation	57
6.3.3	Function Documentation	57
6.3.4	Variable Documentation	63
6.4	Date conversion classes	65
6.4.1	Detailed Description	65
6.5	File services	66
6.5.1	Detailed Description	66
6.5.2	Enumeration Type Documentation	66
6.6	Parameters	67
6.6.1	Detailed Description	67
6.6.2	Function Documentation	67
6.7	Date conversion C APIs	69
6.7.1	Detailed Description	69

6.7.2	Function Documentation	69
6.8	C API for reading data	77
6.8.1	Detailed Description	77
6.8.2	Function Documentation	77
6.9	Date conversion Fortran APIs	79
6.9.1	Detailed Description	79
6.9.2	Function Documentation	79
6.10	Fortran API for reading data	92
6.10.1	Detailed Description	92
6.10.2	Function Documentation	92
7	Namespace Documentation	94
7.1	brathl Namespace Reference	94
7.1.1	Detailed Description	98
7.1.2	Typedef Documentation	98
7.1.3	Variable Documentation	99
8	Class Documentation	101
8.1	_structDateDSM Struct Reference	101
8.1.1	Detailed Description	101
8.1.2	Member Data Documentation	101
8.2	_structDateJulian Struct Reference	102
8.2.1	Detailed Description	102
8.2.2	Member Data Documentation	102
8.3	_structDateSecond Struct Reference	102
8.3.1	Detailed Description	103
8.3.2	Member Data Documentation	103
8.4	_structDateYMDHMSM Struct Reference	103
8.4.1	Detailed Description	103
8.5	brathl::CArrayDoubleArray Class Reference	103
8.5.1	Detailed Description	104
8.6	brathl::CArrayDoublePtrArray Class Reference	104
8.6.1	Detailed Description	105
8.7	brathl::CBratAlgoFilterGaussian1D Class Reference	105
8.7.1	Detailed Description	106
8.7.2	Constructor & Destructor Documentation	106
8.7.3	Member Function Documentation	106
8.8	brathl::CBratAlgoFilterGaussian2D Class Reference	107
8.8.1	Detailed Description	108
8.8.2	Constructor & Destructor Documentation	108
8.8.3	Member Function Documentation	108

8.9	bratl::CBratAlgoFilterLanczos1D Class Reference	110
8.9.1	Detailed Description	110
8.9.2	Constructor & Destructor Documentation	110
8.9.3	Member Function Documentation	111
8.10	bratl::CBratAlgoFilterLanczos2D Class Reference	112
8.10.1	Detailed Description	112
8.10.2	Constructor & Destructor Documentation	112
8.10.3	Member Function Documentation	112
8.11	bratl::CBratAlgoFilterLoess1D Class Reference	113
8.11.1	Detailed Description	114
8.11.2	Constructor & Destructor Documentation	115
8.11.3	Member Function Documentation	115
8.12	bratl::CBratAlgoFilterLoess2D Class Reference	117
8.12.1	Detailed Description	118
8.12.2	Constructor & Destructor Documentation	118
8.12.3	Member Function Documentation	118
8.13	bratl::CBratAlgoFilterMedian1D Class Reference	120
8.13.1	Detailed Description	121
8.13.2	Constructor & Destructor Documentation	121
8.13.3	Member Function Documentation	122
8.14	bratl::CBratAlgoFilterMedian2D Class Reference	123
8.14.1	Detailed Description	124
8.14.2	Constructor & Destructor Documentation	124
8.14.3	Member Function Documentation	125
8.15	bratl::CBratAlgorithmBase Class Reference	126
8.15.1	Detailed Description	129
8.15.2	Constructor & Destructor Documentation	129
8.15.3	Member Function Documentation	129
8.16	bratl::CBratAlgorithmGeosVel Class Reference	131
8.16.1	Detailed Description	133
8.16.2	Constructor & Destructor Documentation	133
8.16.3	Member Function Documentation	133
8.17	bratl::CBratAlgorithmGeosVelAtp Class Reference	133
8.17.1	Detailed Description	135
8.17.2	Constructor & Destructor Documentation	135
8.17.3	Member Function Documentation	135
8.18	bratl::CBratAlgorithmGeosVelGrid Class Reference	137
8.18.1	Detailed Description	139
8.19	bratl::CBratAlgorithmGeosVelGridU Class Reference	139
8.19.1	Detailed Description	140

8.20	bratl::CBratAlgorithmGeosVelGridV Class Reference	140
8.20.1	Detailed Description	140
8.21	bratl::CCriteria Class Reference	141
8.21.1	Detailed Description	141
8.21.2	Member Function Documentation	141
8.22	bratl::CCriteriaCycle Class Reference	142
8.22.1	Detailed Description	143
8.22.2	Constructor & Destructor Documentation	143
8.22.3	Member Function Documentation	144
8.22.4	Member Data Documentation	146
8.23	bratl::CCriteriaCycleInfo Class Reference	147
8.23.1	Detailed Description	148
8.24	bratl::CCriteriaDatetime Class Reference	148
8.24.1	Detailed Description	149
8.24.2	Constructor & Destructor Documentation	149
8.24.3	Member Function Documentation	150
8.24.4	Member Data Documentation	152
8.25	bratl::CCriteriaDatetimeInfo Class Reference	152
8.25.1	Detailed Description	153
8.26	bratl::CCriterialInfo Class Reference	153
8.26.1	Detailed Description	154
8.27	bratl::CCriteriaLatLon Class Reference	154
8.27.1	Detailed Description	156
8.27.2	Constructor & Destructor Documentation	156
8.27.3	Member Function Documentation	157
8.27.4	Member Data Documentation	160
8.28	bratl::CCriteriaLatLonInfo Class Reference	160
8.28.1	Detailed Description	161
8.29	bratl::CCriteriaPass Class Reference	162
8.29.1	Detailed Description	162
8.30	bratl::CCriteriaPassInfo Class Reference	162
8.30.1	Detailed Description	163
8.31	bratl::CCriteriaPassInt Class Reference	164
8.31.1	Detailed Description	165
8.32	bratl::CCriteriaPassIntInfo Class Reference	165
8.32.1	Detailed Description	166
8.33	bratl::CCriteriaPassString Class Reference	166
8.33.1	Detailed Description	167
8.34	bratl::CCriteriaPassStringInfo Class Reference	167
8.34.1	Detailed Description	168

8.35	brathl::CDataSet Class Reference	168
8.35.1	Detailed Description	169
8.35.2	Member Function Documentation	169
8.36	brathl::CDate Class Reference	170
8.36.1	Detailed Description	173
8.36.2	Constructor & Destructor Documentation	173
8.36.3	Member Function Documentation	173
8.36.4	Member Data Documentation	186
8.37	brathl::CDatePeriod Class Reference	187
8.37.1	Detailed Description	188
8.37.2	Constructor & Destructor Documentation	188
8.37.3	Member Function Documentation	189
8.37.4	Member Data Documentation	192
8.38	brathl::CDoubleMap Class Reference	192
8.38.1	Detailed Description	193
8.39	brathl::CDoublePtrArray Class Reference	193
8.39.1	Detailed Description	194
8.40	brathl::CDoublePtrDoubleMap Class Reference	194
8.40.1	Detailed Description	195
8.41	brathl::CExpressionValue Class Reference	195
8.41.1	Detailed Description	196
8.42	brathl::CExternalFilesAvisoGrid Class Reference	196
8.42.1	Detailed Description	197
8.42.2	Member Function Documentation	198
8.43	brathl::CExternalFilesJason2 Class Reference	198
8.43.1	Detailed Description	198
8.44	brathl::CExternalFilesNetCDF Class Reference	198
8.44.1	Detailed Description	200
8.44.2	Member Function Documentation	200
8.45	brathl::CField Class Reference	201
8.45.1	Detailed Description	204
8.45.2	Member Data Documentation	205
8.46	brathl::CFieldArray Class Reference	205
8.46.1	Detailed Description	206
8.47	brathl::CFieldBasic Class Reference	206
8.47.1	Detailed Description	207
8.48	brathl::CFieldIndexData Class Reference	207
8.48.1	Detailed Description	208
8.49	brathl::CFieldNetCdf Class Reference	208
8.49.1	Detailed Description	211

8.49.2	Member Data Documentation	211
8.50	bratl::CFieldNetCdfCF Class Reference	212
8.50.1	Detailed Description	213
8.51	bratl::CFieldNetCdfCFAAttr Class Reference	213
8.51.1	Detailed Description	215
8.52	bratl::CFieldRecord Class Reference	215
8.52.1	Detailed Description	216
8.53	bratl::CFieldSet Class Reference	216
8.53.1	Detailed Description	217
8.54	bratl::CFieldSetArrayDbf Class Reference	217
8.54.1	Detailed Description	218
8.55	bratl::CFieldSetDbf Class Reference	218
8.55.1	Detailed Description	219
8.56	bratl::CFieldSetString Class Reference	219
8.56.1	Detailed Description	220
8.57	bratl::CFile Class Reference	220
8.57.1	Detailed Description	221
8.57.2	Constructor & Destructor Documentation	221
8.57.3	Member Function Documentation	221
8.58	bratl::CFileParams Class Reference	227
8.58.1	Detailed Description	228
8.58.2	Constructor & Destructor Documentation	228
8.58.3	Member Function Documentation	228
8.58.4	Member Data Documentation	229
8.59	bratl::CProduct::CInfo Class Reference	229
8.59.1	Detailed Description	229
8.60	bratl::CInternalFiles Class Reference	230
8.60.1	Detailed Description	231
8.61	bratl::CInternalFilesYFX Class Reference	231
8.61.1	Detailed Description	232
8.62	bratl::CInternalFilesZFX Class Reference	232
8.62.1	Detailed Description	233
8.63	bratl::CIntList Class Reference	233
8.63.1	Detailed Description	233
8.64	bratl::CIntMap Class Reference	234
8.64.1	Detailed Description	234
8.65	bratl::CField::CListField Class Reference	234
8.65.1	Detailed Description	235
8.65.2	Member Function Documentation	235
8.66	bratl::CProduct::CListInfo Class Reference	235

8.66.1 Detailed Description	235
8.67 bratl::CMapParameter Class Reference	236
8.67.1 Detailed Description	236
8.68 bratl::CMapProduct Class Reference	236
8.68.1 Detailed Description	237
8.69 bratl::CObArray Class Reference	237
8.69.1 Detailed Description	238
8.70 bratl::CObDoubleMap Class Reference	238
8.70.1 Detailed Description	239
8.71 bratl::CObIntMap Class Reference	239
8.71.1 Detailed Description	240
8.72 bratl::CObList Class Reference	240
8.72.1 Detailed Description	240
8.73 bratl::CObMap Class Reference	241
8.73.1 Detailed Description	242
8.74 bratl::CObStack Class Reference	242
8.74.1 Detailed Description	242
8.75 bratl::CParameter Class Reference	242
8.75.1 Detailed Description	243
8.75.2 Constructor & Destructor Documentation	243
8.75.3 Member Function Documentation	244
8.76 bratl::CProductAop Class Reference	245
8.76.1 Detailed Description	246
8.76.2 Constructor & Destructor Documentation	246
8.77 bratl::CProductCryosat Class Reference	246
8.77.1 Detailed Description	247
8.77.2 Constructor & Destructor Documentation	247
8.78 bratl::CProductEnvisat Class Reference	247
8.78.1 Detailed Description	248
8.78.2 Constructor & Destructor Documentation	248
8.78.3 Member Function Documentation	248
8.79 bratl::CProductEnvisatNetCdf Class Reference	249
8.79.1 Detailed Description	250
8.79.2 Constructor & Destructor Documentation	250
8.80 bratl::CProductErs Class Reference	250
8.80.1 Detailed Description	251
8.80.2 Constructor & Destructor Documentation	251
8.80.3 Member Function Documentation	251
8.81 bratl::CProductErsWAP Class Reference	252
8.81.1 Detailed Description	253

8.81.2	Constructor & Destructor Documentation	253
8.81.3	Member Function Documentation	253
8.82	brathl::CProductGeosatGDR Class Reference	253
8.82.1	Detailed Description	254
8.82.2	Constructor & Destructor Documentation	254
8.83	brathl::CProductGfo Class Reference	254
8.83.1	Detailed Description	255
8.83.2	Constructor & Destructor Documentation	255
8.83.3	Member Function Documentation	256
8.84	brathl::CProductJason Class Reference	256
8.84.1	Detailed Description	257
8.84.2	Constructor & Destructor Documentation	257
8.84.3	Member Function Documentation	257
8.85	brathl::CProductJason1NetCdf Class Reference	257
8.85.1	Detailed Description	258
8.85.2	Constructor & Destructor Documentation	258
8.86	brathl::CProductJason2 Class Reference	258
8.86.1	Detailed Description	259
8.86.2	Constructor & Destructor Documentation	259
8.87	brathl::CProductList Class Reference	259
8.87.1	Detailed Description	260
8.88	brathl::CProductNetCdf Class Reference	261
8.88.1	Detailed Description	263
8.88.2	Constructor & Destructor Documentation	263
8.88.3	Member Data Documentation	264
8.89	brathl::CProductNetCdfCF Class Reference	264
8.89.1	Detailed Description	265
8.89.2	Constructor & Destructor Documentation	265
8.89.3	Member Data Documentation	265
8.90	brathl::CProductPodaac Class Reference	266
8.90.1	Detailed Description	266
8.90.2	Constructor & Destructor Documentation	267
8.91	brathl::CProductRads Class Reference	268
8.91.1	Detailed Description	268
8.91.2	Constructor & Destructor Documentation	269
8.92	brathl::CProductReaper Class Reference	269
8.92.1	Detailed Description	269
8.92.2	Constructor & Destructor Documentation	270
8.93	brathl::CProductRiverLake Class Reference	270
8.93.1	Detailed Description	271

8.93.2	Constructor & Destructor Documentation	271
8.94	brathl::CProductTopex Class Reference	271
8.94.1	Detailed Description	272
8.94.2	Constructor & Destructor Documentation	272
8.94.3	Member Function Documentation	273
8.94.4	Member Data Documentation	273
8.95	brathl::CProductTopexSDR Class Reference	273
8.95.1	Detailed Description	274
8.95.2	Constructor & Destructor Documentation	274
8.95.3	Member Function Documentation	274
8.96	brathl::CPtrMap Class Reference	275
8.96.1	Detailed Description	275
8.97	brathl::CRecord Class Reference	275
8.97.1	Detailed Description	276
8.98	brathl::CRecordSet Class Reference	276
8.98.1	Detailed Description	277
8.99	brathl::CRegisteredPass Class Reference	277
8.99.1	Detailed Description	277
8.100	brathl::CStringList Class Reference	278
8.100.1	Detailed Description	279
8.101	brathl::CStringMap Class Reference	279
8.101.1	Detailed Description	279
8.102	brathl::CTools Class Reference	279
8.102.1	Detailed Description	283
8.102.2	Member Function Documentation	283
8.103	brathl::CTreeField Class Reference	309
8.103.1	Detailed Description	309
8.104	brathl::CUIntMap Class Reference	309
8.104.1	Detailed Description	310
8.105	PyAlgo Class Reference	310
8.105.1	Detailed Description	311
8.105.2	Constructor & Destructor Documentation	311
8.105.3	Member Function Documentation	311
8.106	PythonEngine Class Reference	314
8.106.1	Detailed Description	315
9	File Documentation	315
9.1	brathl.h File Reference	315
9.1.1	Detailed Description	316
9.1.2	Macro Definition Documentation	316

9.1.3	Typedef Documentation	316
9.1.4	Enumeration Type Documentation	317
9.1.5	Variable Documentation	317
9.2	brathl_fortran.c File Reference	317
9.2.1	Detailed Description	318
9.3	brathlc.h File Reference	318
9.3.1	Detailed Description	319
9.3.2	Function Documentation	320
9.3.3	Variable Documentation	320
9.4	MapParameter.h File Reference	320
9.4.1	Detailed Description	320

1 Module Index

1.1 Modules

Here is a list of all modules:

Algorithms classes	12
Tools	18
Criteria	47
Date conversion classes	65
File services	66
Parameters	67
Date conversion C APIs	69
C API for reading data	77
Date conversion Fortran APIs	79
Fortran API for reading data	92

2 Namespace Index

2.1 Namespace List

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

brathl	94
---------------	-----------

3 Hierarchical Index

3.1 Class Hierarchy

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

_structDateDSM	101
_structDateJulian	102
_structDateSecond	102
_structDateYMDHMSM	103
bratl::CArrayDoubleArray	103
bratl::CArrayDoublePtrArray	104
bratl::CBratAlgoFilterGaussian1D	105
bratl::CBratAlgoFilterGaussian2D	107
bratl::CBratAlgoFilterLanczos1D	110
bratl::CBratAlgoFilterLanczos2D	112
bratl::CBratAlgoFilterLoess1D	113
bratl::CBratAlgoFilterLoess2D	117
bratl::CBratAlgoFilterMedian1D	120
bratl::CBratAlgoFilterMedian2D	123
bratl::CBratAlgorithmBase	126
bratl::CBratAlgorithmGeosVel	131
bratl::CBratAlgorithmGeosVelAtp	133
bratl::CBratAlgorithmGeosVelGrid	137
bratl::CBratAlgorithmGeosVelGridU	139
bratl::CBratAlgorithmGeosVelGridV	140
PyAlgo	310
bratl::CCriteria	141
bratl::CCriteriaCycle	142
bratl::CCriteriaDatetime	148
bratl::CCriteriaLatLon	154
bratl::CCriteriaPass	162
bratl::CCriteriaPassInt	164
bratl::CCriteriaPassString	166
bratl::CCriterialInfo	153
bratl::CCriteriaCycleInfo	147

brathl::CCriteriaDatetimeInfo	152
brathl::CCriteriaLatLonInfo	160
brathl::CCriteriaPassInfo	162
brathl::CCriteriaPassIntInfo	165
brathl::CCriteriaPassStringInfo	167
brathl::CDate	170
brathl::CDatePeriod	187
brathl::CDoubleMap	192
brathl::CDoublePtrArray	193
brathl::CDoublePtrDoubleMap	194
brathl::CExpressionValue	195
brathl::CExternalFilesAvisoGrid	196
brathl::CExternalFilesJason2	198
brathl::CExternalFilesNetCDF	198
brathl::CField	201
brathl::CFieldArray	205
brathl::CFieldRecord	215
brathl::CFieldBasic	206
brathl::CFieldIndexData	207
brathl::CFieldNetCdf	208
brathl::CFieldNetCdfCF	212
brathl::CFieldNetCdfCFAttr	213
brathl::CFieldSet	216
brathl::CFieldSetArrayDbl	217
brathl::CFieldSetDbl	218
brathl::CFieldSetString	219
brathl::CFile	220
brathl::CFileParams	227
brathl::CProduct::CInfo	229
brathl::CInternalFiles	230
brathl::CInternalFilesYFX	231
brathl::CInternalFilesZFX	232

brathl::CIntList	233
brathl::CIntMap	234
brathl::CMapParameter	236
brathl::CObArray	237
brathl::CDataSet	168
brathl::CObDoubleMap	238
brathl::CObIntMap	239
brathl::CObList	240
brathl::CField::CListField	234
brathl::CProduct::CListInfo	235
brathl::CObMap	241
brathl::CMapProduct	236
brathl::CRecordSet	276
brathl::CObStack	242
brathl::CParameter	242
brathl::CProductAop	245
brathl::CProductCryosat	246
brathl::CProductEnvisat	247
brathl::CProductErs	250
brathl::CProductErsWAP	252
brathl::CProductGfo	254
brathl::CProductJason	256
brathl::CProductNetCdf	261
brathl::CProductNetCdfCF	264
brathl::CProductEnvisatNetCdf	249
brathl::CProductGeosatGDR	253
brathl::CProductJason1NetCdf	257
brathl::CProductJason2	258
brathl::CProductRads	268
brathl::CProductReaper	269
brathl::CProductPodaac	266
brathl::CProductRiverLake	270

bratl::CProductTopex	271
bratl::CProductTopexSDR	273
bratl::CPtrMap	275
bratl::CRecord	275
bratl::CRegisteredPass	277
bratl::CStringList	278
bratl::CProductList	259
bratl::CStringMap	279
bratl::CTools	279
bratl::CTreeField	309
bratl::CUIntMap	309
PythonEngine	314

4 Class Index

4.1 Class List

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

_structDateDSM	101
_structDateJulian	102
_structDateSecond	102
_structDateYMDHMSM	103
bratl::CArrayDoubleArray	103
bratl::CArrayDoublePtrArray	104
bratl::CBratAlgoFilterGaussian1D	105
bratl::CBratAlgoFilterGaussian2D	107
bratl::CBratAlgoFilterLanczos1D	110
bratl::CBratAlgoFilterLanczos2D	112
bratl::CBratAlgoFilterLoess1D	113
bratl::CBratAlgoFilterLoess2D	117
bratl::CBratAlgoFilterMedian1D	120
bratl::CBratAlgoFilterMedian2D	123
bratl::CBratAlgorithmBase	126
bratl::CBratAlgorithmGeosVel	131

brathl::CBratAlgorithmGeosVelAtp	133
brathl::CBratAlgorithmGeosVelGrid	137
brathl::CBratAlgorithmGeosVelGridU	139
brathl::CBratAlgorithmGeosVelGridV	140
brathl::CCriteria	141
brathl::CCriteriaCycle	142
brathl::CCriteriaCycleInfo	147
brathl::CCriteriaDatetime	148
brathl::CCriteriaDatetimeInfo	152
brathl::CCriterialInfo	153
brathl::CCriteriaLatLon	154
brathl::CCriteriaLatLonInfo	160
brathl::CCriteriaPass	162
brathl::CCriteriaPassInfo	162
brathl::CCriteriaPassInt	164
brathl::CCriteriaPassIntInfo	165
brathl::CCriteriaPassString	166
brathl::CCriteriaPassStringInfo	167
brathl::CDataSet	168
brathl::CDate	170
brathl::CDatePeriod	187
brathl::CDoubleMap	192
brathl::CDoublePtrArray	193
brathl::CDoublePtrDoubleMap	194
brathl::CExpressionValue	195
brathl::CExternalFilesAvisoGrid	196
brathl::CExternalFilesJason2	198
brathl::CExternalFilesNetCDF	198
brathl::CField	201
brathl::CFieldArray	205
brathl::CFieldBasic	206
brathl::CFieldIndexData	207

brathl::CFieldNetCdf	208
brathl::CFieldNetCdfCF	212
brathl::CFieldNetCdfCFAttr	213
brathl::CFieldRecord	215
brathl::CFieldSet	216
brathl::CFieldSetArrayDbI	217
brathl::CFieldSetDbI	218
brathl::CFieldSetString	219
brathl::CFile	220
brathl::CFileParams	227
brathl::CProduct::CInfo	229
brathl::CInternalFiles	230
brathl::CInternalFilesYFX	231
brathl::CInternalFilesZFX	232
brathl::CIntList	233
brathl::CIntMap	234
brathl::CField::CListField	234
brathl::CProduct::CListInfo	235
brathl::CMapParameter	236
brathl::CMapProduct	236
brathl::CObArray	237
brathl::CObDoubleMap	238
brathl::CObIntMap	239
brathl::CObList	240
brathl::CObMap	241
brathl::CObStack	242
brathl::CParameter	242
brathl::CProductAop	245
brathl::CProductCryosat	246
brathl::CProductEnvisat	247
brathl::CProductEnvisatNetCdf	249
brathl::CProductErs	250

brathl::CProductErsWAP	252
brathl::CProductGeosatGDR	253
brathl::CProductGfo	254
brathl::CProductJason	256
brathl::CProductJason1NetCdf	257
brathl::CProductJason2	258
brathl::CProductList	259
brathl::CProductNetCdf	261
brathl::CProductNetCdfCF	264
brathl::CProductPodaac	266
brathl::CProductRads	268
brathl::CProductReaper	269
brathl::CProductRiverLake	270
brathl::CProductTopex	271
brathl::CProductTopexSDR	273
brathl::CPtrMap	275
brathl::CRecord	275
brathl::CRecordSet	276
brathl::CRegisteredPass	277
brathl::CStringList	278
brathl::CStringMap	279
brathl::CTools	279
brathl::CTreeField	309
brathl::CUIntMap	309
PyAlgo	
Definition of the object to hold each Python Algorithm and respective variables/methods	310
PythonEngine	
Definition of the object to hold the Python Interpreter and respective methods	314

5 File Index

5.1 File List

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

Aliases.h	??
------------------	-----------

AliasesDictionary.h	??
argtable2.h	??
BratAlgoFilter.h	??
BratAlgoFilterGaussian.h	??
BratAlgoFilterGaussian1D.h	??
BratAlgoFilterGaussian2D.h	??
BratAlgoFilterKernel.h	??
BratAlgoFilterLanczos.h	??
BratAlgoFilterLanczos1D.h	??
BratAlgoFilterLanczos2D.h	??
BratAlgoFilterLoess.h	??
BratAlgoFilterLoess1D.h	??
BratAlgoFilterLoess2D.h	??
BratAlgoFilterMedian.h	??
BratAlgoFilterMedian1D.h	??
BratAlgoFilterMedian2D.h	??
BratAlgorithmBase.h	??
BratAlgorithmGeosVel.h	??
BratAlgorithmGeosVelAtp.h	??
BratAlgorithmGeosVelGrid.h	??
BratEmbeddedPythonProcess.h	??
brathl.h	315
brathl_fortran.c	317
brathl_fortran.h	??
brathlc.h	318
BratObject.h	??
BratProcess.h	??
BratProcessExportAscii.h	??
BratProcessStats.h	??
BratProcessYFX.h	??
BratProcessZFX.Y.h	??
CallBack.h	??

Criteria.h	??
CriteriaCycle.h	??
CriteriaDatetime.h	??
CriterialInfo.h	??
CriteriaLatLon.h	??
CriteriaPass.h	??
CyclePassConverter.h	??
Date.h	??
DatePeriod.h	??
deelx.h	??
Expression.h	??
ExternalFiles.h	??
ExternalFilesATP.h	??
ExternalFilesAvisoGrid.h	??
ExternalFilesFactory.h	??
ExternalFilesJason2.h	??
ExternalFilesNetCDF.h	??
Field.h	??
File.h	??
FileParams.h	??
getopt.h	??
InternalFiles.h	??
InternalFilesFactory.h	??
InternalFilesYFX.h	??
InternalFilesZFX.Y.h	??
LatLonPoint.h	??
LatLonRect.h	??
List.h	??
MapParameter.h	320
NetCDFFiles.h	??
ObjectTree.h	??
Parameter.h	??

ParametersDictionary.h	??
pragmalocation.h	??
ProcessCommonTools.h	??
Product.h	??
ProductAop.h	??
ProductCryosat.h	??
ProductEnvisat.h	??
ProductEnvisatNetCdf.h	??
ProductErs.h	??
ProductErsWAP.h	??
ProductGeosatGDR.h	??
ProductGfo.h	??
ProductJason.h	??
ProductJason1NetCdf.h	??
ProductJason2.h	??
ProductNetCdf.h	??
ProductNetCdfCF.h	??
ProductPodaac.h	??
ProductRads.h	??
ProductReaper.h	??
ProductRiverLake.h	??
ProductTopex.h	??
ProductTopexSDR.h	??
PythonEngine.hpp	??
RunPythonAlgorithm.hpp	??
Tools.h	??
TreeField.h	??
Unit.h	??
Win32MemLeaksAccurate.h	??
Xml.h	??

6 Module Documentation

6.1 Algorithms classes

Classes

- class **bratl::CBratAlgoFilterGaussian1D**
- class **bratl::CBratAlgoFilterGaussian2D**
- class **bratl::CBratAlgoFilterLanczos1D**
- class **bratl::CBratAlgoFilterLanczos2D**
- class **bratl::CBratAlgoFilterLoess1D**
- class **bratl::CBratAlgoFilterLoess2D**
- class **bratl::CBratAlgoFilterMedian1D**
- class **bratl::CBratAlgoFilterMedian2D**
- class **bratl::CBratAlgorithmBase**
- class **bratl::CBratAlgorithmGeosVel**
- class **bratl::CBratAlgorithmGeosVelAtp**
- class **bratl::CBratAlgorithmGeosVelGrid**
- class **bratl::CBratAlgorithmGeosVelGridU**
- class **bratl::CBratAlgorithmGeosVelGridV**

Macros

- **#define AUTO_REGISTER_BASE(base) CBratAlgorithmBaseRegistration _base_registration_ ## base(new base_creator(&base_factory<base>));**

Typedefs

- **typedef std::map< std::string, CBratAlgorithmBase * > bratl::mapbratalgorithmbase**
- **typedef std::vector < CBratAlgorithmBase * > bratl::vectorbratalgorithmbase**

Functions

- **template<class T > CBratAlgorithmBase * bratl::base_factory ()**
- **bratl::CBratAlgorithmGeosVelGrid::CBratAlgorithmGeosVelGrid ()**
- **bratl::CBratAlgorithmGeosVelGrid::CBratAlgorithmGeosVelGrid (const CBratAlgorithmGeosVelGrid ©)**
- **bratl::CBratAlgorithmGeosVelGridU::CBratAlgorithmGeosVelGridU ()**
- **bratl::CBratAlgorithmGeosVelGridU::CBratAlgorithmGeosVelGridU (const CBratAlgorithmGeosVelGridU ©)**
- **bratl::CBratAlgorithmGeosVelGridV::CBratAlgorithmGeosVelGridV ()**
- **bratl::CBratAlgorithmGeosVelGridV::CBratAlgorithmGeosVelGridV (const CBratAlgorithmGeosVelGridV ©)**
- **void bratl::CBratAlgorithmGeosVelGrid::CheckEquatorLimit ()**
- **virtual void bratl::CBratAlgorithmGeosVelGrid::CheckInputParams (CVectorBratAlgorithmParam &args) override**
- **void bratl::CBratAlgorithmGeosVelGrid::CheckLatLonExpression (uint32_t index)**
- **void bratl::CBratAlgorithmGeosVelGrid::CheckProduct ()**
- **void bratl::CBratAlgorithmGeosVelGrid::CheckVarExpression (uint32_t index)**
- **double bratl::CBratAlgorithmGeosVelGrid::ComputeMean ()**
- **double bratl::CBratAlgorithmGeosVelGrid::ComputeSingle ()**
- **virtual double bratl::CBratAlgorithmGeosVelGrid::ComputeVelocity ()=0**
- **double bratl::CBratAlgorithmGeosVelGridU::ComputeVelocity () override**

- double **bratl::CBratAlgorithmGeosVelGridV::ComputeVelocity** () override
- virtual void **bratl::CBratAlgorithmGeosVelGrid::DeleteFieldNetCdf** () override
- virtual void **bratl::CBratAlgorithmGeosVelGrid::DeleteProduct** () override
- virtual void **bratl::CBratAlgorithmGeosVelGrid::Dump** (std::ostream &fOut=std::cerr) override
- virtual void **bratl::CBratAlgorithmGeosVelGridU::Dump** (std::ostream &fOut=std::cerr) override
- virtual void **bratl::CBratAlgorithmGeosVelGridV::Dump** (std::ostream &fOut=std::cerr) override
- virtual std::string **bratl::CBratAlgorithmGeosVelGridU::GetDescription** () const override
- virtual std::string **bratl::CBratAlgorithmGeosVelGridV::GetDescription** () const override
- virtual std::string **bratl::CBratAlgorithmGeosVelGrid::GetInputParamDesc** (uint32_t indexParam) const override
- virtual
 CBratAlgorithmParam::bratAlgoParamTypeVal **bratl::CBratAlgorithmGeosVelGrid::GetInputParam**↵
 Format (uint32_t indexParam) const override
- virtual std::string **bratl::CBratAlgorithmGeosVelGrid::GetInputParamUnit** (uint32_t indexParam) const override
- uint32_t **bratl::CBratAlgorithmGeosVelGrid::GetLatDimRange** (CFieldNetCdf *field)
- int32_t **bratl::CBratAlgorithmGeosVelGrid::GetLatitudeIndex** (double value)
- void **bratl::CBratAlgorithmGeosVelGrid::GetLatitudes** ()
- uint32_t **bratl::CBratAlgorithmGeosVelGrid::GetLonDimRange** (CFieldNetCdf *field)
- int32_t **bratl::CBratAlgorithmGeosVelGrid::GetLongitudeIndex** (double value)
- void **bratl::CBratAlgorithmGeosVelGrid::GetLongitudes** ()
- virtual std::string **bratl::CBratAlgorithmGeosVelGridU::GetName** () const override
- virtual std::string **bratl::CBratAlgorithmGeosVelGridV::GetName** () const override
- virtual uint32_t **bratl::CBratAlgorithmGeosVelGrid::GetNumInputParam** () const override
- virtual std::string **bratl::CBratAlgorithmGeosVelGrid::GetOutputUnit** () const override
- virtual double **bratl::CBratAlgorithmGeosVelGrid::GetParamDefaultValue** (uint32_t indexParam)
- virtual std::string **bratl::CBratAlgorithmGeosVelGrid::GetParamName** (uint32_t indexParam) const override
- void **bratl::CBratAlgorithmGeosVelGrid::GetVarCacheExpressionValue** (int32_t minIndexLat, int32_t ↵
 maxIndexLat, int32_t minIndexLon, int32_t maxIndexLon)
- double **bratl::CBratAlgorithmGeosVelGrid::GetVarExpressionValue** (int32_t indexLat, int32_t indexLon)
- double **bratl::CBratAlgorithmGeosVelGrid::GetVarExpressionValueCache** (int32_t indexLat, int32_t ↵
 indexLon)
- void **bratl::CBratAlgorithmGeosVelGrid::Init** ()
- void **bratl::CBratAlgorithmGeosVelGridU::Init** ()
- void **bratl::CBratAlgorithmGeosVelGridV::Init** ()
- virtual void **bratl::CBratAlgorithmGeosVelGrid::OpenProductFile** () override
- **CBratAlgorithmGeosVelGrid** & **bratl::CBratAlgorithmGeosVelGrid::operator=** (const **CBrat**↵
 AlgorithmGeosVelGrid ©)
- bool **bratl::CBratAlgorithmGeosVelGrid::PrepareComputeVelocity** ()
- virtual void **bratl::CBratAlgorithmGeosVelGrid::PrepareDataReading2D** (int32_t minIndexLat, int32_t ↵
 maxIndexLat, int32_t minIndexLon, int32_t maxIndexLon)
- virtual void **bratl::CBratAlgorithmGeosVelGrid::PrepareDataReading2D** (int32_t indexLat, int32_t ↵
 indexLon)
- virtual void **bratl::CBratAlgorithmGeosVelGrid::PrepareDataValues2DComplexExpression** (C↵
 ExpressionValue &exprValue)
- virtual void **bratl::CBratAlgorithmGeosVelGrid::PrepareDataValues2DComplexExpressionWithAlgo**
 (**CExpressionValue** &exprValue)
- virtual void **bratl::CBratAlgorithmGeosVelGrid::PrepareDataValues2DOneField** (**CExpressionValue**
 &exprValue)
- virtual double **bratl::CBratAlgorithmGeosVelGrid::Run** (CVectorBratAlgorithmParam &args) override
- void **bratl::CBratAlgorithmGeosVelGrid::Set** (const **CBratAlgorithmGeosVelGrid** ©)
- virtual void **bratl::CBratAlgorithmGeosVelGrid::SetBeginOfFile** () override
- virtual void **bratl::CBratAlgorithmGeosVelGrid::SetEndOfFile** () override
- virtual void **bratl::CBratAlgorithmGeosVelGrid::SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual **bratl::CBratAlgorithmGeosVelGrid::~CBratAlgorithmGeosVelGrid** ()
- virtual **bratl::CBratAlgorithmGeosVelGridU::~CBratAlgorithmGeosVelGridU** ()
- virtual **bratl::CBratAlgorithmGeosVelGridV::~CBratAlgorithmGeosVelGridV** ()

Variables

- bool **bratl::CBratAlgorithmGeosVelGrid::m_allLongitudes**
- static const uint32_t **bratl::CBratAlgorithmGeosVelGrid::m_EQUATOR_LAT_LIMIT_INDEX = 3**
- double **bratl::CBratAlgorithmGeosVelGrid::m_equatorLimit**
- CFieldNetCdf * **bratl::CBratAlgorithmGeosVelGrid::m_fieldLat**
- CFieldNetCdf * **bratl::CBratAlgorithmGeosVelGrid::m_fieldLon**
- int32_t **bratl::CBratAlgorithmGeosVelGrid::m_indexLat**
- int32_t **bratl::CBratAlgorithmGeosVelGrid::m_indexLon**
- static const uint32_t **bratl::CBratAlgorithmGeosVelGrid::m_INPUT_PARAMS = 4**
- static const uint32_t **bratl::CBratAlgorithmGeosVelGrid::m_LAT_PARAM_INDEX = 0**
- CDoubleArray **bratl::CBratAlgorithmGeosVelGrid::m_latitudes**
- static const uint32_t **bratl::CBratAlgorithmGeosVelGrid::m_LON_PARAM_INDEX = 1**
- CDoubleArray **bratl::CBratAlgorithmGeosVelGrid::m_longitudes**
- double **bratl::CBratAlgorithmGeosVelGrid::m_lonMax**
- double **bratl::CBratAlgorithmGeosVelGrid::m_lonMin**
- CExpressionValue **bratl::CBratAlgorithmGeosVelGrid::m_rawDataCache**
- static const uint32_t **bratl::CBratAlgorithmGeosVelGrid::m_VAR_PARAM_INDEX = 2**
- int32_t **bratl::CBratAlgorithmGeosVelGrid::m_varDimLatIndex**
- int32_t **bratl::CBratAlgorithmGeosVelGrid::m_varDimLonIndex**
- double **bratl::CBratAlgorithmGeosVelGrid::m_varValue**
- double **bratl::CBratAlgorithmGeosVelGrid::m_varValueE**
- double **bratl::CBratAlgorithmGeosVelGrid::m_varValueN**
- double **bratl::CBratAlgorithmGeosVelGrid::m_varValueS**
- double **bratl::CBratAlgorithmGeosVelGrid::m_varValueW**

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 **bratl::CBratAlgorithmGeosVelGrid::CBratAlgorithmGeosVelGrid ()**

Default constructor

6.1.2.2 **bratl::CBratAlgorithmGeosVelGrid::CBratAlgorithmGeosVelGrid (const CBratAlgorithmGeosVelGrid & copy)**

Copy constructor

6.1.2.3 **bratl::CBratAlgorithmGeosVelGridU::CBratAlgorithmGeosVelGridU ()**

Default constructor

6.1.2.4 **bratl::CBratAlgorithmGeosVelGridU::CBratAlgorithmGeosVelGridU (const CBratAlgorithmGeosVelGridU & copy)**

Copy constructor

6.1.2.5 **bratl::CBratAlgorithmGeosVelGridV::CBratAlgorithmGeosVelGridV ()**

Default constructor

6.1.2.6 **bratl::CBratAlgorithmGeosVelGridV::CBratAlgorithmGeosVelGridV (const CBratAlgorithmGeosVelGridV & copy)**

Copy constructor

6.1.2.7 `void brathl::CBratAlgorithmGeosVelGrid::Dump (std::ostream & fOut = std::cerr) [override],
[virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmGeosVel** (p. 133).

Reimplemented in **brathl::CBratAlgorithmGeosVelGridV** (p. 15), and **brathl::CBratAlgorithmGeosVelGridU** (p. 15).

References **brathl::CBratAlgorithmGeosVel::Dump()**, and **brathl::CFieldNetCdf::Dump()**.

Referenced by **brathl::CBratAlgorithmGeosVelGridU::Dump()**, and **brathl::CBratAlgorithmGeosVelGridV::Dump()**.

6.1.2.8 `void brathl::CBratAlgorithmGeosVelGridU::Dump (std::ostream & fOut = std::cerr) [override],
[virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmGeosVelGrid** (p. 15).

References **brathl::CBratAlgorithmGeosVelGrid::Dump()**.

6.1.2.9 `void brathl::CBratAlgorithmGeosVelGridV::Dump (std::ostream & fOut = std::cerr) [override],
[virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmGeosVelGrid** (p. 15).

References **brathl::CBratAlgorithmGeosVelGrid::Dump()**.

6.1.2.10 `virtual std::string brathl::CBratAlgorithmGeosVelGridU::GetDescription () const [inline],[override],
[virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

6.1.2.11 `virtual std::string brathl::CBratAlgorithmGeosVelGridV::GetDescription () const [inline],[override],
[virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

6.1.2.12 `virtual std::string brathl::CBratAlgorithmGeosVelGrid::GetInputParamDesc (uint32_t indexParam) const
[inline],[override],[virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References **brathl::CTools::Format()**, and **brathl::CBratAlgorithmGeosVelGrid::GetNumInputParam()**.

6.1.2.13 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::CBratAlgorithmGeosVelGrid::GetInputParamFormat (
uint32_t indexParam) const [inline],[override],[virtual]`

Gets the format of an input parameter : **CBratAlgorithmParam::T_DOUBLE** for double **CBratAlgorithmParam::T_FLOAT** for float **CBratAlgorithmParam::T_INT** for integer **CBratAlgorithmParam::T_LONG** for long integer **CBratAlgorithmParam::T_STRING** for `std::string` **CBratAlgorithmParam::T_CHAR** for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **bratl::CBratAlgorithmGeosVelGrid::GetNumInputParam()**.

6.1.2.14 `virtual std::string bratl::CBratAlgorithmGeosVelGrid::GetInputParamUnit (uint32_t indexParam) const`
[inline], [override], [virtual]

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **bratl::CBratAlgorithmGeosVelGrid::GetNumInputParam()**.

6.1.2.15 `virtual std::string bratl::CBratAlgorithmGeosVelGridU::GetName () const` [inline], [override], [virtual]

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

6.1.2.16 `virtual std::string bratl::CBratAlgorithmGeosVelGridV::GetName () const` [inline], [override], [virtual]

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

6.1.2.17 `virtual uint32_t bratl::CBratAlgorithmGeosVelGrid::GetNumInputParam () const` [inline], [override], [virtual]

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by **bratl::CBratAlgorithmGeosVelGrid::GetInputParamDesc()**, **bratl::CBratAlgorithmGeosVelGrid::GetInputParamFormat()**, and **bratl::CBratAlgorithmGeosVelGrid::GetInputParamUnit()**.

6.1.2.18 `virtual std::string bratl::CBratAlgorithmGeosVelGrid::GetOutputUnit () const` [inline], [override], [virtual]

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

6.1.2.19 `CBratAlgorithmGeosVelGrid & bratl::CBratAlgorithmGeosVelGrid::operator= (const CBratAlgorithmGeosVelGrid & copy)`

Overloads operator '='

6.1.2.20 `double bratl::CBratAlgorithmGeosVelGrid::Run (CVectorBratAlgorithmParam & args)` [override], [virtual]

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a <code>std::string</code> that indicates the format of each value of input parameters (number, <code>std::string</code>) : d for integer l for long integer f for double s for <code>std::string</code>
<i>args</i>	[in] : the values of input parameters i(a C/C++ <code>va_list</code>).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

6.1.2.21 **bratl::CBratAlgorithmGeosVelGrid::~CBratAlgorithmGeosVelGrid** () [virtual]

Destructor

6.1.2.22 **bratl::CBratAlgorithmGeosVelGridU::~CBratAlgorithmGeosVelGridU** () [virtual]

Destructor

6.1.2.23 **bratl::CBratAlgorithmGeosVelGridV::~CBratAlgorithmGeosVelGridV** () [virtual]

Destructor

6.2 Tools

Namespaces

- **brathl**

Classes

- class **brathl::CArrayDoubleArray**
- class **brathl::CArrayDoublePtrArray**
- class **brathl::CDoubleMap**
- class **brathl::CDoublePtrArray**
- class **brathl::CDoublePtrDoubleMap**
- class **brathl::CExpressionValue**
- class **brathl::CExternalFilesAvisoGrid**
- class **brathl::CExternalFilesJason2**
- class **brathl::CExternalFilesNetCDF**
- class **brathl::CInternalFiles**
- class **brathl::CInternalFilesYFX**
- class **brathl::CInternalFilesZFX**
- class **brathl::CIntList**
- class **brathl::CIntMap**
- class **brathl::CObArray**
- class **brathl::CObDoubleMap**
- class **brathl::CObIntMap**
- class **brathl::CObList**
- class **brathl::CObMap**
- class **brathl::CObStack**
- class **brathl::CPtrMap**
- class **brathl::CRegisteredPass**
- class **brathl::CStringList**
- class **brathl::CStringMap**
- class **brathl::CTools**
- class **brathl::CUIntMap**

Macros

- **#define ADD_OFFSET_ATTR** "add_offset"
- **#define AT_BEGINNING** 0xFFFFFFFFUL
- **#define AXIS_ATTR** "axis"
- **#define COMMENT_ATTR** "comment"
- **#define CONVENTIONS_ATTR** "Conventions"
- **#define DATA_SET_ATTR** "data_set"
- **#define FILE_TITLE_ATTR** "title"
- **#define FILE_TYPE_ATTR** "FileType"
- **#define FILL_VALUE_ATTR** "_FillValue"
- **#define LONG_NAME_ATTR** "long_name"
- **#define MISSION_NAME_ATTR** "mission_name"
- **#define PRODUCT_TYPE_ATTR** "product_type"
- **#define SCALE_FACTOR_ATTR** "scale_factor"
- **#define STANDARD_NAME_ATTR** "standard_name"
- **#define TITLE_ATTR** "title"
- **#define UNITS_ATTR** "units"
- **#define VALID_MAX_ATTR** "valid_max"
- **#define VALID_MIN_ATTR** "valid_min"

Typedefs

- typedef std::map< std::string, CStringArray > **bratl::maparraystring**
- typedef std::map< std::string, CObjectTreeNode * > **bratl::mapTreeNode**

Functions

- void **bratl::CArrayDoublePtrArray::AdjustValidMinMax** (double value)
- void **bratl::CArrayDoubleArray::AdjustValidMinMax** (double value)
- double * **bratl::CMatrix::At** (size_t i, size_t j)
- CExternalFiles * **bratl::BuildExistingExternalFileKind** (const std::string &path)
- CInternalFiles * **bratl::BuildExistingInternalFileKind** (const std::string &name, const CStringArray *fieldNames)
- **bratl::CArrayDoubleArray::CArrayDoubleArray** ()
Empty CDoubleArray ctor.
- **bratl::CArrayDoubleArray::CArrayDoubleArray** (const CArrayDoubleArray &a)
- **bratl::CArrayDoublePtrArray::CArrayDoublePtrArray** (bool bDelete=true)
Empty CDoubleArray ctor.
- **bratl::CArrayDoublePtrArray::CArrayDoublePtrArray** (const CArrayDoublePtrArray &a)
- **bratl::CArrayStringMap::CArrayStringMap** ()
CStringMap (p. 279) ctor.
- **bratl::CArrayStringMap::CArrayStringMap** (const CArrayStringMap &a)
- **bratl::CDoubleArrayOb::CDoubleArrayOb** (const CDoubleArrayOb &vect)
- **bratl::CDoubleMap::CDoubleMap** ()
CDoubleMap (p. 192) ctor.
- **bratl::CDoublePtrArray::CDoublePtrArray** (bool bDelete=true)
Empty CDoublePtrArray (p. 193) ctor.
- **bratl::CDoublePtrDoubleMap::CDoublePtrDoubleMap** (bool bDelete=true)
CDoublePtrDoubleMap (p. 194) ctor.
- **bratl::CDoublePtrDoubleMap::CDoublePtrDoubleMap** (const CUIntArray &matrixDims, bool bDelete=true)
- **bratl::CIntList::CIntList** ()
Empty CIntList (p. 233) ctor.
- **bratl::CIntList::CIntList** (const CIntList &list)
- **bratl::CIntMap::CIntMap** ()
CIntMap (p. 234) ctor.
- virtual CBratObject * **bratl::CDoubleArrayOb::Clone** ()
- virtual CBratObject * **bratl::CObArrayOb::Clone** ()
- **bratl::CMatrix::CMatrix** (const CMatrix &m)
- **bratl::CMatrixDouble::CMatrixDouble** (size_t nrows, size_t ncols)
- **bratl::CMatrixDouble::CMatrixDouble** (const CMatrixDouble &m)
- **bratl::CMatrixDoublePtr::CMatrixDoublePtr** (size_t nrows, size_t ncols)
- **bratl::CMatrixDoublePtr::CMatrixDoublePtr** (const CMatrixDoublePtr &m)
- **bratl::CObArray::CObArray** (bool bDelete=true)
Empty CObArray (p. 237) ctor.
- **bratl::CObArray::CObArray** (const CObArray &vect)
- **bratl::CObArrayOb::CObArrayOb** (bool bDelete=true)
- **bratl::CObArrayOb::CObArrayOb** (const CObArrayOb &vect)
- **bratl::CObDoubleMap::CObDoubleMap** (bool bDelete=true)
CObMap (p. 241) ctor.
- **bratl::CObIntMap::CObIntMap** (bool bDelete=true)

- *COBMap* (p. 241) ctor.
- **brathl::CObjectPointersArray< BRAT_OBJECT >::CObjectPointersArray** (bool del=true)
- **brathl::CObjectPointersArray< BRAT_OBJECT >::CObjectPointersArray** (const CObjectPointersArray< BRAT_OBJECT > &o)
- **brathl::COBList::COBList** (bool bDelete=true)
- *Empty COBList* (p. 240) ctor.
- **brathl::COBList::COBList** (const COBList &lst)
- **brathl::COBMap::COBMap** (bool bDelete=true)
- *COBMap* (p. 241) ctor.
- **brathl::COBMap::COBMap** (const COBMap &obMap)
- **brathl::COBStack::COBStack** (bool bDelete=true)
- *Empty COBArray* (p. 237) ctor.
- virtual bool **brathl::CStringList::Complement** (const CStringList &array, CStringList &complement) const
- **brathl::CPtrMap::CPtrMap** (bool bDelete=true)
- *CPtrMap* (p. 275) ctor.
- **brathl::CStringList::CStringList** ()
- *Empty CStringList* (p. 278) ctor.
- **brathl::CStringList::CStringList** (const CStringList &list)
- **brathl::CStringList::CStringList** (const stringlist &list)
- **brathl::CStringList::CStringList** (const CStringArray &vect)
- **brathl::CStringList::CStringList** (const std::vector< std::string > &vect)
- **brathl::CStringMap::CStringMap** ()
- *CStringMap* (p. 279) ctor.
- **brathl::CUIntMap::CUIntMap** ()
- *CUIntMap* (p. 309) ctor.
- void **brathl::CDoublePtrArray::Delete** (DoublePtr matrix)
- void **brathl::CArrayDoublePtrArray::Delete** (DoublePtr matrix)
- void **brathl::CDoublePtrDoubleMap::Delete** (DoublePtr *matrix)
- virtual void **brathl::CStringList::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CIntList::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::COBList::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CDoublePtrArray::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CArrayDoublePtrArray::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CArrayDoubleArray::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CArrayStringMap::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CDoubleArrayOb::Dump** (std::ostream &fOut=std::cerr) const
- virtual void **brathl::COBArray::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::COBArrayOb::Dump** (std::ostream &fOut=std::cerr) const
- virtual void **brathl::CStringMap::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CIntMap::Dump** (std::ostream &fOut=std::cerr) const
- *Dump fonction.*
- virtual void **brathl::CUIntMap::Dump** (std::ostream &fOut=std::cerr) const

- Dump function.*
- virtual void **brathl::CDoubleMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CObMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CObIntMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CObDoubleMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CDoublePtrDoubleMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CPtrMap::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CMatrix::Dump** (std::ostream &fOut=std::cerr) const
- Dump function.*
- virtual void **brathl::CMatrixDoublePtr::Dump** (std::ostream &fOut=std::cerr) const override
- Dump function.*
- virtual void **brathl::CMatrixDouble::Dump** (std::ostream &fOut=std::cerr) const override
- Dump function.*
- virtual void **brathl::CStringList::Erase** (const std::string &str)
- virtual void **brathl::CStringList::Erase** (CStringList::iterator it)
- bool **brathl::CObList::Erase** (CBratObject *ob)
- virtual bool **brathl::CObList::Erase** (CObList::iterator it)
- virtual bool **brathl::CDoublePtrArray::Erase** (CDoublePtrArray::iterator it)
- virtual bool **brathl::CDoublePtrArray::Erase** (int32_t index)
- virtual bool **brathl::CArrayStringMap::Erase** (CArrayStringMap::iterator it)
- virtual bool **brathl::CArrayStringMap::Erase** (const std::string &key)
- bool **brathl::CObArray::Erase** (CBratObject *ob)
- virtual bool **brathl::CObArray::Erase** (CObArray::iterator it)
- virtual bool **brathl::CObArray::Erase** (int32_t index)
- virtual bool **brathl::CStringMap::Erase** (CStringMap::iterator it)
- virtual bool **brathl::CStringMap::Erase** (const std::string &key)
- virtual bool **brathl::CIntMap::Erase** (CIntMap::iterator it)
- virtual bool **brathl::CIntMap::Erase** (const std::string &key)
- virtual bool **brathl::CUIntMap::Erase** (CUIntMap::iterator it)
- virtual bool **brathl::CUIntMap::Erase** (const std::string &key)
- virtual bool **brathl::CDoubleMap::Erase** (CDoubleMap::iterator it)
- virtual bool **brathl::CDoubleMap::Erase** (const std::string &key)
- virtual bool **brathl::CObMap::Erase** (CObMap::iterator it)
- virtual bool **brathl::CObMap::Erase** (const std::string &key)
- virtual bool **brathl::CObIntMap::Erase** (CObIntMap::iterator it)
- virtual bool **brathl::CObIntMap::Erase** (int32_t key)
- virtual bool **brathl::CObDoubleMap::Erase** (CObDoubleMap::iterator it)
- virtual bool **brathl::CObDoubleMap::Erase** (double key)
- virtual bool **brathl::CDoublePtrDoubleMap::Erase** (CDoublePtrDoubleMap::iterator it)
- virtual bool **brathl::CDoublePtrDoubleMap::Erase** (double key)
- virtual bool **brathl::CPtrMap::Erase** (CPtrMap::iterator it)
- virtual bool **brathl::CPtrMap::Erase** (const std::string &key)
- virtual bool **brathl::CStringList::Exists** (const std::string &str) const
- virtual const CStringArray * **brathl::CArrayStringMap::Exists** (const std::string &key) const
- virtual std::string **brathl::CStringMap::Exists** (const std::string &key) const
- virtual int32_t **brathl::CIntMap::Exists** (const std::string &key) const
- virtual uint32_t **brathl::CUIntMap::Exists** (const std::string &key) const

- virtual double **bratl::CDoubleMap::Exists** (const std::string &key) const
- virtual CBratObject * **bratl::CObMap::Exists** (const std::string &key) const
- virtual CBratObject * **bratl::CObIntMap::Exists** (int32_t key) const
- virtual CBratObject * **bratl::CObDoubleMap::Exists** (double key) const
- virtual DoublePtr * **bratl::CDoublePtrDoubleMap::Exists** (double key) const
- virtual void * **bratl::CPtrMap::Exists** (const std::string &key) const
- virtual bool **bratl::CStringList::ExistsNoCase** (const std::string &str) const
- virtual void **bratl::CStringList::ExtractKeys** (const std::string &str, const std::string &delim, bool bRemoveAll=true)
- virtual void **bratl::CStringList::ExtractStrings** (const std::string &str, const char delim, bool bRemoveAll=true)
- virtual void **bratl::CStringList::ExtractStrings** (const std::string &str, const std::string &delim, bool bRemoveAll=true)
- virtual int32_t **bratl::CStringList::FindIndex** (const std::string &str, bool compareNoCase=false) const
- const CArrayDoublePtrArray & **bratl::CMatrixDoublePtr::GetData** ()
- const CArrayDoubleArray & **bratl::CMatrixDouble::GetData** ()
- bool **bratl::CObList::GetDelete** ()
- bool **bratl::CDoublePtrArray::GetDelete** ()
- bool **bratl::CArrayDoublePtrArray::GetDelete** ()
- bool **bratl::CObStack::GetDelete** ()
- bool **bratl::CObArray::GetDelete** ()
- bool **bratl::CObMap::GetDelete** ()
- bool **bratl::CObIntMap::GetDelete** ()
- bool **bratl::CObDoubleMap::GetDelete** ()
- bool **bratl::CDoublePtrDoubleMap::GetDelete** ()
- virtual void **bratl::CStringMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true) const
- virtual void **bratl::CUIntMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true)
- virtual void **bratl::CObMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual void **bratl::CObMap::GetKeys** (CStringList &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual void **bratl::CObIntMap::GetKeys** (CIntArray &keys, bool bRemoveAll=true)
- virtual void **bratl::CObDoubleMap::GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- virtual void **bratl::CDoublePtrDoubleMap::GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- uint32_t **bratl::CDoublePtrDoubleMap::GetMatrixColDim** (uint32_t row)
- CStringArray * **bratl::CMatrixDoublePtr::GetMatrixDataDimIndexes** ()
- uint32_t **bratl::CDoublePtrArray::GetMatrixDim** (uint32_t row)
- size_t **bratl::CArrayDoublePtrArray::GetMatrixDim** (size_t row)
- size_t **bratl::CMatrixDoublePtr::GetMatrixDimData** (size_t row)
- CUIntArray * **bratl::CDoublePtrArray::GetMatrixDims** ()
- CUIntArray * **bratl::CArrayDoublePtrArray::GetMatrixDims** ()
- const CUIntArray * **bratl::CArrayDoublePtrArray::GetMatrixDims** () const
- CUIntArray * **bratl::CDoublePtrDoubleMap::GetMatrixDims** ()
- const CUIntArray * **bratl::CMatrixDoublePtr::GetMatrixDimsData** () const
- size_t **bratl::CDoublePtrArray::GetMatrixNumberOfDims** ()
- size_t **bratl::CArrayDoublePtrArray::GetMatrixNumberOfDims** ()
- size_t **bratl::CMatrixDoublePtr::GetMatrixNumberOfDimsData** ()
- size_t **bratl::CDoublePtrDoubleMap::GetMatrixNumberOfRows** () const
- virtual size_t **bratl::CMatrix::GetMatrixNumberOfValuesData** () const
- virtual size_t **bratl::CMatrixDoublePtr::GetMatrixNumberOfValuesData** () const override
- virtual size_t **bratl::CMatrixDouble::GetMatrixNumberOfValuesData** () const override
- void **bratl::CArrayDoublePtrArray::GetMinMaxValues** (double &min, double &max, bool recalc=true)
- void **bratl::CArrayDoubleArray::GetMinMaxValues** (double &min, double &max, bool recalc=true)
- virtual void **bratl::CMatrix::GetMinMaxValues** (double &min, double &max)=0
- virtual void **bratl::CMatrixDoublePtr::GetMinMaxValues** (double &min, double &max) override
- virtual void **bratl::CMatrixDouble::GetMinMaxValues** (double &min, double &max) override
- std::string **bratl::CMatrix::GetName** ()

- virtual size_t **brathl::CMatrix::GetNumberOfCols** () const =0
- virtual size_t **brathl::CMatrixDoublePtr::GetNumberOfCols** () const override
- virtual size_t **brathl::CMatrixDouble::GetNumberOfCols** () const override
- virtual size_t **brathl::CMatrix::GetNumberOfRows** () const =0
- virtual size_t **brathl::CMatrixDoublePtr::GetNumberOfRows** () const override
- virtual size_t **brathl::CMatrixDouble::GetNumberOfRows** () const override
- virtual size_t **brathl::CMatrix::GetNumberOfValues** () const =0
- virtual size_t **brathl::CMatrixDoublePtr::GetNumberOfValues** () const override
- virtual size_t **brathl::CMatrixDouble::GetNumberOfValues** () const override
- std::string **brathl::CMatrix::GetXName** ()
- std::string **brathl::CMatrix::GetYName** ()
- void **brathl::CArrayDoublePtrArray::Init** ()
- void **brathl::CArrayDoubleArray::Init** ()
- void **brathl::CArrayStringMap::Init** ()
- void **brathl::CArrayDoublePtrArray::InitMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- void **brathl::CArrayDoubleArray::InitMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- virtual void **brathl::CMatrix::InitMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)=0
- void **brathl::CMatrixDoublePtr::InitMatrix** (double initialValue=CTools::m_defaultValueDOUBLE) override
- void **brathl::CMatrixDouble::InitMatrix** (double initialValue=CTools::m_defaultValueDOUBLE) override
- void **brathl::CArrayDoublePtrArray::InitMatrixData** (double initialValue=CTools::m_defaultValueDOUBLE)
- void **brathl::CMatrixDoublePtr::InitMatrixDimsData** (const CUIntArray &matrixDims, double initialValue=CTools::m_defaultValueDOUBLE)
- virtual void **brathl::CStringList::Insert** (const **CStringList** &list, bool bEnd=true)
- virtual void **brathl::CStringList::Insert** (const std::string &str, bool bEnd=true)
- virtual void **brathl::CStringList::Insert** (const CStringArray &vect, bool bEnd=true)
- virtual void **brathl::CStringList::Insert** (const std::vector< std::string > &vect, bool bEnd=true)
- virtual void **brathl::CStringList::Insert** (const **stringlist** &lst, bool bEnd=true)
- virtual void **brathl::CIntList::Insert** (const **CIntList** &list, bool bEnd=true)
- virtual void **brathl::CIntList::Insert** (const int value, bool bEnd=true)
- virtual void **brathl::CObList::Insert** (const **CObList** &list, bool bEnd=true)
- virtual void **brathl::CObList::Insert** (CBratObject *ob, bool bEnd=true)
- virtual void **brathl::CDoublePtrArray::Insert** (DoublePtr ob)
- virtual CStringArray * **brathl::CArrayStringMap::Insert** (const std::string &key, const CStringArray &str, bool withExcept=true)
- virtual void **brathl::CObArray::Insert** (const **CObArray** &vect)
- virtual void **brathl::CObArray::Insert** (CBratObject *ob)
- virtual void **brathl::CObjectPointersArray< BRAT_OBJECT >::Insert** (const CObjectPointersArray< BRAT_OBJECT > &o)
- virtual std::string **brathl::CStringMap::Insert** (const std::string &key, const std::string &str, bool withExcept=true)
- virtual void **brathl::CStringMap::Insert** (const **CStringMap** &strmap, bool withExcept=true)
- virtual int32_t **brathl::CIntMap::Insert** (const std::string &key, int32_t value, bool withExcept=true)
- virtual void **brathl::CIntMap::Insert** (const **CIntMap** &m, bool bRemoveAll=true, bool withExcept=true)
- virtual void **brathl::CIntMap::Insert** (const CStringArray &keys, const CIntArray &values, bool bRemoveAll=true, bool withExcept=true)
- virtual uint32_t **brathl::CUIntMap::Insert** (const std::string &key, uint32_t value, bool withExcept=true)
- virtual void **brathl::CUIntMap::Insert** (const **CUIntMap** &m, bool bRemoveAll=true, bool withExcept=true)
- virtual void **brathl::CUIntMap::Insert** (const CStringArray &keys, uint32_t initValue, bool bRemoveAll=true, bool withExcept=true)
- virtual void **brathl::CUIntMap::Insert** (const CStringArray &keys, const CUIntArray &values, bool bRemoveAll=true, bool withExcept=true)
- virtual void **brathl::CUIntMap::Insert** (const CStringArray &keys, bool bRemoveAll=true, bool withExcept=true)
- virtual double **brathl::CDoubleMap::Insert** (const std::string &key, double value, bool withExcept=true)

- virtual CBratObject * **bratl::COBMap::Insert** (const std::string &key, CBratObject *ob, bool withExcept=true)
- virtual void **bratl::COBMap::Insert** (const COBMap &obMap, bool withExcept=true)
- virtual CBratObject * **bratl::COBIntMap::Insert** (int32_t key, CBratObject *ob, bool withExcept=true)
- virtual void **bratl::COBIntMap::Insert** (const COBIntMap &obMap, bool withExcept=true)
- virtual CBratObject * **bratl::COBDoubleMap::Insert** (double key, CBratObject *ob, bool withExcept=true)
- virtual void **bratl::COBDoubleMap::Insert** (const COBDoubleMap &obMap, bool withExcept=true)
- virtual DoublePtr * **bratl::CDoublePtrDoubleMap::Insert** (double key, DoublePtr *ob, bool withExcept=true)
- virtual DoublePtr * **bratl::CDoublePtrDoubleMap::Insert** (double key, double initialValue=CTools::m_defaultValueDOUBLE)
- virtual void * **bratl::CPtrMap::Insert** (const std::string &key, void *ptr, bool withExcept=true)
- virtual void **bratl::CPtrMap::Insert** (const CPtrMap &ptrMap, bool withExcept=true)
- virtual CDoublePtrArray::iterator **bratl::CDoublePtrArray::InsertAt** (CDoublePtrArray::iterator where, DoublePtr ob)
- virtual COBArray::iterator **bratl::COBArray::InsertAt** (COBArray::iterator where, CBratObject *ob)
- virtual void **bratl::CStringList::InsertUnique** (const std::string &str, bool bEnd=true)
- virtual void **bratl::CStringList::InsertUnique** (const CStringList &lst, bool bEnd=true)
- virtual void **bratl::CStringList::InsertUnique** (const CStringArray *vect, bool bEnd=true)
- virtual void **bratl::CStringList::InsertUnique** (const CStringArray &vect, bool bEnd=true)
- virtual void **bratl::CStringList::InsertUnique** (const std::vector< std::string > &vect, bool bEnd=true)
- virtual void **bratl::CStringList::InsertUnique** (const stringlist &lst, bool bEnd=true)
- virtual bool **bratl::CStringList::Intersect** (const CStringList &array, CStringList &intersect) const
- virtual bool **bratl::CMatrix::IsMatrixDataSet** ()
- bool **bratl::CMatrixDoublePtr::IsMatrixDataSet** () override
- virtual std::string **bratl::CStringMap::IsValue** (const std::string &value)
- DoublePtr **bratl::CDoublePtrArray::NewMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- DoublePtr **bratl::CArrayDoublePtrArray::NewMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- DoublePtr * **bratl::CDoublePtrDoubleMap::NewMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- DoublePtr **bratl::CMatrixDoublePtr::NewMatrixData** (double initialValue=CTools::m_defaultValueDOUBLE)
- virtual double * **bratl::CMatrix::operator()** (size_t i, size_t j)=0
- virtual double * **bratl::CMatrix::operator()** (size_t i, size_t j) const =0
- virtual double * **bratl::CMatrixDoublePtr::operator()** (size_t i, size_t j) override
- virtual double * **bratl::CMatrixDoublePtr::operator()** (size_t i, size_t j) const override
- virtual double * **bratl::CMatrixDouble::operator()** (size_t i, size_t j) override
- virtual double * **bratl::CMatrixDouble::operator()** (size_t i, size_t j) const override
- virtual const CStringList & **bratl::CStringList::operator=** (const CStringList &lst)
- virtual const CStringList & **bratl::CStringList::operator=** (const CStringArray &vect)
- virtual const CStringList & **bratl::CStringList::operator=** (const std::vector< std::string > &vect)
- virtual const CStringList & **bratl::CStringList::operator=** (const stringlist &lst)
- const CIntList & **bratl::CIntList::operator=** (const CIntList &lst)
- virtual const COBList & **bratl::COBList::operator=** (const COBList &lst)
- virtual const CArrayDoublePtrArray & **bratl::CArrayDoublePtrArray::operator=** (const CArrayDoublePtrArray &m)
- virtual const CArrayDoubleArray & **bratl::CArrayDoubleArray::operator=** (const CArrayDoubleArray &m)
- virtual const CArrayStringMap & **bratl::CArrayStringMap::operator=** (const CArrayStringMap &a)
- virtual const CDoubleArrayOb & **bratl::CDoubleArrayOb::operator=** (const CDoubleArrayOb &vect)
- virtual const COBArray & **bratl::COBArray::operator=** (const COBArray &lst)
- CObjectPointersArray < BRAT_OBJECT > & **bratl::CObjectPointersArray< BRAT_OBJECT >::operator=** (const CObjectPointersArray< BRAT_OBJECT > &o)

- virtual const CObArrayOb & **brathl::CObArrayOb::operator=** (const CObArrayOb &vect)
- virtual const CObMap & **brathl::CObMap::operator=** (const CObMap &obMap)
- virtual const CObIntMap & **brathl::CObIntMap::operator=** (const CObIntMap &obMap)
- virtual const CObDoubleMap & **brathl::CObDoubleMap::operator=** (const CObDoubleMap &obMap)
- const CMatrix & **brathl::CMatrix::operator=** (const CMatrix &m)
- const CMatrixDoublePtr & **brathl::CMatrixDoublePtr::operator=** (const CMatrixDoublePtr &m)
- const CMatrixDouble & **brathl::CMatrixDouble::operator=** (const CMatrixDouble &m)
- virtual int32_t **brathl::CIntMap::operator[]** (const std::string &key)
- virtual uint32_t **brathl::CUIntMap::operator[]** (const std::string &key)
- virtual double **brathl::CDoubleMap::operator[]** (const std::string &key)
- virtual CBratObject * **brathl::CObMap::operator[]** (const std::string &key)
- virtual CBratObject * **brathl::CObIntMap::operator[]** (int32_t key)
- virtual CBratObject * **brathl::CObDoubleMap::operator[]** (double key)
- virtual DoublePtr * **brathl::CDoublePtrDoubleMap::operator[]** (double key)
- virtual void * **brathl::CPtrMap::operator[]** (const std::string &key)
- virtual doubleptrarray & **brathl::CMatrixDoublePtr::operator[]** (const size_t &i)
- virtual const doubleptrarray & **brathl::CMatrixDoublePtr::operator[]** (const size_t &i) const
- virtual std::vector< double > & **brathl::CMatrixDouble::operator[]** (const size_t &i)
- virtual const std::vector< double > & **brathl::CMatrixDouble::operator[]** (const size_t &i) const
- virtual void **brathl::CObStack::Pop** ()
- virtual bool **brathl::CObList::PopBack** ()
- virtual bool **brathl::CDoublePtrArray::PopBack** ()
- virtual bool **brathl::CObArray::PopBack** ()
- virtual void **brathl::CObStack::Push** (CBratObject *ob)
- virtual void **brathl::CArrayDoublePtrArray::Remove** (doubleptrarray &vect)
- virtual void **brathl::CStringList::RemoveAll** ()
- virtual void **brathl::CIntList::RemoveAll** ()
- virtual void **brathl::CObList::RemoveAll** ()
- virtual void **brathl::CDoublePtrArray::RemoveAll** ()
- virtual void **brathl::CArrayDoublePtrArray::RemoveAll** ()
- virtual void **brathl::CArrayDoubleArray::RemoveAll** ()
- virtual void **brathl::CArrayStringMap::RemoveAll** ()
- virtual void **brathl::CObStack::RemoveAll** ()
- virtual void **brathl::CObArray::RemoveAll** ()
- virtual void **brathl::CObjectPointersArray< BRAT_OBJECT >::RemoveAll** ()
- virtual void **brathl::CStringMap::RemoveAll** ()
- virtual void **brathl::CIntMap::RemoveAll** ()
- virtual void **brathl::CUIntMap::RemoveAll** ()
- virtual void **brathl::CDoubleMap::RemoveAll** ()
- virtual void **brathl::CObMap::RemoveAll** ()
- virtual void **brathl::CObIntMap::RemoveAll** ()
- virtual void **brathl::CObDoubleMap::RemoveAll** ()
- virtual void **brathl::CDoublePtrDoubleMap::RemoveAll** ()
- virtual void **brathl::CPtrMap::RemoveAll** ()
- bool **brathl::CObMap::RenameKey** (const std::string &oldKey, const std::string &newKey)
- bool **brathl::CObIntMap::RenameKey** (int32_t oldKey, int32_t newKey)
- bool **brathl::CObDoubleMap::RenameKey** (double oldKey, double newKey)
- bool **brathl::CDoublePtrDoubleMap::RenameKey** (double oldKey, double newKey)
- virtual CDoublePtrArray::iterator **brathl::CDoublePtrArray::ReplaceAt** (CDoublePtrArray::iterator where, DoublePtr ob)
- virtual CObArray::iterator **brathl::CObArray::ReplaceAt** (CObArray::iterator where, CBratObject *ob)
- void **brathl::CArrayDoublePtrArray::ResizeRC** (size_t nrows, size_t ncols)
- void **brathl::CArrayDoubleArray::ResizeRC** (size_t nrows, size_t ncols)

- virtual void **bratl::CMatrix::ScaleDownData** (double scaleFactor, double addOffset, double default←
Value=CTools::m_defaultValueDOUBLE)=0
- virtual void **bratl::CMatrixDoublePtr::ScaleDownData** (double scaleFactor, double addOffset, double
defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **bratl::CMatrixDouble::ScaleDownData** (double scaleFactor, double addOffset, double
defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **bratl::CMatrix::ScaleUpData** (double scaleFactor, double addOffset, double defaultValue=C←
Tools::m_defaultValueDOUBLE)=0
- virtual void **bratl::CMatrixDoublePtr::ScaleUpData** (double scaleFactor, double addOffset, double
defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **bratl::CMatrixDouble::ScaleUpData** (double scaleFactor, double addOffset, double default←
Value=CTools::m_defaultValueDOUBLE) override
- void **bratl::CArrayDoublePtrArray::Set** (const CArrayDoublePtrArray &m)
- void **bratl::CArrayDoubleArray::Set** (const CArrayDoubleArray &m)
- virtual void **bratl::CArrayStringMap::Set** (const CArrayStringMap &a)
- virtual void **bratl::CMatrix::Set** (const CMatrix &m)
- virtual void **bratl::CMatrix::Set** (size_t &row, size_t &col, double *x)=0
- void **bratl::CMatrixDoublePtr::Set** (size_t &row, size_t &col, double *x) override
- void **bratl::CMatrixDoublePtr::Set** (const CMatrixDoublePtr &m)
- void **bratl::CMatrixDouble::Set** (size_t &row, size_t &col, double *x) override
- void **bratl::CMatrixDouble::Set** (const CMatrixDouble &m)
- void **bratl::CObList::SetDelete** (bool value)
- void **bratl::CDoublePtrArray::SetDelete** (bool value)
- void **bratl::CArrayDoublePtrArray::SetDelete** (bool value)
- void **bratl::CObStack::SetDelete** (bool value)
- void **bratl::CObArray::SetDelete** (bool value)
- void **bratl::CObMap::SetDelete** (bool value)
- void **bratl::CObIntMap::SetDelete** (bool value)
- void **bratl::CObDoubleMap::SetDelete** (bool value)
- void **bratl::CDoublePtrDoubleMap::SetDelete** (bool value)
- void **bratl::CMatrixDoublePtr::SetMatrixDataDimIndexes** (const CStringArray &m)
- void **bratl::CDoublePtrArray::SetMatrixDims** (const CUIntArray &matrixDims)
- void **bratl::CArrayDoublePtrArray::SetMatrixDims** (const CUIntArray &matrixDims)
- void **bratl::CDoublePtrDoubleMap::SetMatrixDims** (const CUIntArray &matrixDims)
- void **bratl::CMatrixDoublePtr::SetMatrixDimsData** (const CUIntArray &matrixDims)
- void **bratl::CMatrixDoublePtr::SetMatrixDimsData** (size_t nbValues)
- void **bratl::CMatrix::SetName** (const std::string &value)
- void **bratl::CMatrix::SetXName** (const std::string &value)
- void **bratl::CMatrix::SetYName** (const std::string &value)
- virtual void **bratl::CObMap::ToArray** (CObArray &obArray)
- virtual CBratObject * **bratl::CObStack::Top** ()
- virtual std::string **bratl::CStringList::ToString** (const std::string &delim="," , bool useBracket=true) const
- virtual **bratl::CArrayDoubleArray::~~CArrayDoubleArray** ()
- Destructor.*
- virtual **bratl::CArrayDoublePtrArray::~~CArrayDoublePtrArray** ()
- Destructor.*
- virtual **bratl::CArrayStringMap::~~CArrayStringMap** ()
- CStringMap* (p. 279) *dtor.*
- virtual **bratl::CDoubleMap::~~CDoubleMap** ()
- CDoubleMap* (p. 192) *dtor.*
- virtual **bratl::CDoublePtrArray::~~CDoublePtrArray** ()
- Destructor.*
- virtual **bratl::CDoublePtrDoubleMap::~~CDoublePtrDoubleMap** ()
- CDoublePtrDoubleMap* (p. 194) *dtor.*

- virtual **brathl::CIntList::~~CIntList ()**
Destructor.
- virtual **brathl::CIntMap::~~CIntMap ()**
CIntMap (p. 234) dtor.
- virtual **brathl::CObArray::~~CObArray ()**
Destructor.
- virtual **brathl::CObDoubleMap::~~CObDoubleMap ()**
CObMap (p. 241) dtor.
- virtual **brathl::CObIntMap::~~CObIntMap ()**
CObMap (p. 241) dtor.
- virtual **brathl::CObList::~~CObList ()**
Destructor.
- virtual **brathl::CObMap::~~CObMap ()**
CObMap (p. 241) dtor.
- virtual **brathl::CObStack::~~CObStack ()**
Destructor.
- virtual **brathl::CPtrMap::~~CPtrMap ()**
CPtrMap (p. 275) dtor.
- virtual **brathl::CStringList::~~CStringList ()**
Destructor.
- virtual **brathl::CStringMap::~~CStringMap ()**
CStringMap (p. 279) dtor.
- virtual **brathl::CUIntMap::~~CUIntMap ()**
CUIntMap (p. 309) dtor.

Variables

- const std::string **brathl::GENERIC_NETCDF_TYPE** = "Generic NetCdf"
- bool **brathl::CObList::m_bDelete**
- bool **brathl::CDoublePtrArray::m_bDelete**
- bool **brathl::CArrayDoublePtrArray::m_bDelete**
- bool **brathl::CObStack::m_bDelete**
Dump fonction.
- bool **brathl::CObArray::m_bDelete**
- bool **brathl::CObMap::m_bDelete**
- bool **brathl::CObIntMap::m_bDelete**
- bool **brathl::CObDoubleMap::m_bDelete**
- bool **brathl::CDoublePtrDoubleMap::m_bDelete**
- bool **brathl::CPtrMap::m_bDelete**
- CArrayDoublePtrArray **brathl::CMatrixDoublePtr::m_data**
- CStringArray **brathl::CMatrixDoublePtr::m_matrixDataDimIndexes**
- CUIntArray **brathl::CDoublePtrArray::m_matrixDims**
- CUIntArray **brathl::CArrayDoublePtrArray::m_matrixDims**
- CUIntArray **brathl::CDoublePtrDoubleMap::m_matrixDims**
- double **brathl::CArrayDoublePtrArray::m_maxValue**
- double **brathl::CArrayDoubleArray::m_maxValue**
- double **brathl::CArrayDoublePtrArray::m_minValue**
- double **brathl::CArrayDoubleArray::m_minValue**
- const std::string **brathl::NETCDF_CF_PRODUCT_CLASS** = "NETCDF_CF"
- const std::string **brathl::NETCDF_PRODUCT_CLASS** = "NETCDF"
- const std::string **brathl::UNKNOWN_PRODUCT_CLASS** = "UNKNOWN"
- const std::string **brathl::YFX_NETCDF_TYPE** = "Y=F(X)"
- const std::string **brathl::ZFXNETCDF_TYPE** = "Z=F(X,Y)"

6.2.1 Detailed Description

6.2.2 Macro Definition Documentation

6.2.2.1 #define FILL_VALUE_ATTR "_FillValue"

NetCDF files access.

Version

1.0

6.2.3 Typedef Documentation

6.2.3.1 typedef std::map<std::string, CStringArray> brathl::maparraystring

a set of array std::string value management classes.

Version

1.0

Creates a type name for std::map of std::string array

6.2.4 Function Documentation

6.2.4.1 CExternalFiles * brathl::BuildExistingExternalFileKind (const std::string & path)

External files access.

Version

1.0

6.2.4.2 CInternalFiles * brathl::BuildExistingInternalFileKind (const std::string & name, const CStringArray * fieldNames = NULL)

Internal files access.

Version

1.0

References brathl::CTools::Format().

6.2.4.3 brathl::CIntList::CIntList (const CIntList & list)

Creates new **CIntList** (p. 233) object from another **CStringList** (p. 278)

Parameters

<i>std::list</i>	[in] : std::list to be copied
------------------	-------------------------------

6.2.4.4 brathl::CObArray::CObArray (const CObArray & vect)

Creates new **CObArray** (p. 237) object from another **CObArray** (p. 237)

Parameters

<i>vect</i>	[in] : std::list to be copied
-------------	-------------------------------

6.2.4.5 bratl::CObList::CObList (const CObList & *lst*)

Creates new **CObList** (p. 240) object from another **CStringList** (p. 278)

Parameters

<i>lst</i>	[in] : std::list to be copied
------------	-------------------------------

6.2.4.6 bratl::CStringList::CStringList (const CStringList & *list*)

Creates new **CStringList** (p. 278) object from another **CStringList** (p. 278)

Parameters

<i>std::list</i>	[in] : std::list to be copied
------------------	-------------------------------

6.2.4.7 bool bratl::CObList::Erase (CBratObject * *ob*)

Delete an element referenced by *ob*

Returns

true if no error, otherwise false

6.2.4.8 bool bratl::CObList::Erase (CObList::iterator *it*) [virtual]

Delete an element referenced by *it*

Returns

true if no error, otherwise false

6.2.4.9 bool bratl::CDoublePtrArray::Erase (CDoublePtrArray::iterator *it*) [virtual]

Delete an element referenced by *it*

Returns

true if no error, otherwise false

Referenced by bratl::CDoublePtrArray::Erase().

6.2.4.10 bool bratl::CDoublePtrArray::Erase (int32_t *index*) [virtual]

Delete an element referenced by *index*

Returns

true if no error, otherwise false

References bratl::CDoublePtrArray::Erase().

6.2.4.11 bool bratl::CArrayStringMap::Erase (CArrayStringMap::iterator *it*) [virtual]

Delete an element referenced by *it*

Returns

true if no error, otherwise false

6.2.4.12 `bool brathl::CArrayStringMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

6.2.4.13 `bool brathl::CObArray::Erase (CBratObject * ob)`

Delete an element referenced by ob

Returns

true if no error, otherwise false

Referenced by `brathl::CObArray::Erase()`.

6.2.4.14 `bool brathl::CObArray::Erase (CObArray::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

6.2.4.15 `bool brathl::CObArray::Erase (int32_t index) [virtual]`

Delete an element referenced by index

Returns

true if no error, otherwise false

References `brathl::CObArray::Erase()`.

6.2.4.16 `bool brathl::CStringMap::Erase (CStringMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CStringMap::Erase()`.

6.2.4.17 `bool brathl::CStringMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CStringMap::Erase()`.

6.2.4.18 `bool brathl::CIntMap::Erase (CIntMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CIntMap::Erase()`.

6.2.4.19 `bool brathl::CIntMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CIntMap::Erase()`.

6.2.4.20 `bool brathl::CUIntMap::Erase (CUIntMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CUIntMap::Erase()`.

6.2.4.21 `bool brathl::CUIntMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CUIntMap::Erase()`.

6.2.4.22 `bool brathl::CDoubleMap::Erase (CDoubleMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CDoubleMap::Erase()`.

6.2.4.23 `bool brathl::CDoubleMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CDoubleMap::Erase()`.

6.2.4.24 `bool brathl::CObMap::Erase (CObMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CObMap::Erase()`, and `brathl::CDataSet::EraseFieldSet()`.

6.2.4.25 `bool brathl::CObMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CObMap::Erase()`.

6.2.4.26 `bool brathl::CObIntMap::Erase (CObIntMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CObIntMap::Erase()`.

6.2.4.27 `bool brathl::CObIntMap::Erase (int32_t key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CObIntMap::Erase()`.

6.2.4.28 `bool brathl::CObDoubleMap::Erase (CObDoubleMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CObDoubleMap::Erase()`.

6.2.4.29 `bool brathl::CObDoubleMap::Erase (double key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CObDoubleMap::Erase()`.

6.2.4.30 `bool brathl::CDoublePtrDoubleMap::Erase (CDoublePtrDoubleMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CDoublePtrDoubleMap::Erase()`.

6.2.4.31 `bool brathl::CDoublePtrDoubleMap::Erase (double key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CDoublePtrDoubleMap::Erase()`.

6.2.4.32 `bool brathl::CPtrMap::Erase (CPtrMap::iterator it) [virtual]`

Delete an element referenced by it

Returns

true if no error, otherwise false

Referenced by `brathl::CPtrMap::Erase()`.

6.2.4.33 `bool brathl::CPtrMap::Erase (const std::string & key) [virtual]`

Delete an element by its key

Returns

true if no error, otherwise false

References `brathl::CPtrMap::Erase()`.

6.2.4.34 `const CStringArray * brathl::CStringMap::Exists (const std::string & key) const [virtual]`

Tests if an element identify by 'key' already exists

Returns

a `std::string` array value corresponding to the key; if exists, otherwise empty `std::string`

6.2.4.35 `std::string brathl::CStringMap::Exists (const std::string & key) const [virtual]`

Tests if an element identify by 'key' already exists

Returns

a `std::string` value corresponding to the key; if exists, otherwise empty `std::string`

6.2.4.36 `int32_t brathl::CIntMap::Exists (const std::string & key) const [virtual]`

Tests if an element identify by 'key' already exists

Returns

a integer value corresponding to the key; if exists, otherwise default value **CTools::m_defaultValueINT32** (p. 283)

References `brathl::CTools::m_defaultValueINT32`.

Referenced by `brathl::CIntMap::operator[]()`.

6.2.4.37 `uint32_t bratl::CUIntMap::Exists (const std::string & key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a integer value corresponding to the key; if exists, otherwise default value **CTools::m_defaultValueUINT32** (p. 283)

References `bratl::CTools::m_defaultValueUINT32`.

Referenced by `bratl::CUIntMap::operator[]()`.

6.2.4.38 `double bratl::CDoubleMap::Exists (const std::string & key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a double value corresponding to the key; if exists, otherwise default value **CTools::m_defaultValueDOUBLE** (p. 283)

References `bratl::CTools::m_defaultValueDOUBLE`.

Referenced by `bratl::CDoubleMap::operator[]()`.

6.2.4.39 `CBratObject * bratl::CObMap::Exists (const std::string & key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a `CBratObject` pointer if exists, otherwise `NULL`

6.2.4.40 `CBratObject * bratl::COblntMap::Exists (int32_t key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a `CBratObject` pointer if exists, otherwise `NULL`

6.2.4.41 `CBratObject * bratl::CObDoubleMap::Exists (double key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a `CBratObject` pointer if exists, otherwise `NULL`

6.2.4.42 `DoublePtr * bratl::CDoublePtrDoubleMap::Exists (double key) const` `[virtual]`

Tests if an element identify by 'key' already exists

Returns

a `CBratObject` pointer if exists, otherwise `NULL`

6.2.4.43 `void * brathl::CPtrMap::Exists (const std::string & key) const` [virtual]

Tests if an element identify by 'key' already exists

Returns

a pointer if exists, otherwise NULL

6.2.4.44 `void brathl::CStringMap::GetKeys (CStringArray & keys, bool bRemoveAll=true) const` [virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.45 `void brathl::CUIntMap::GetKeys (CStringArray & keys, bool bRemoveAll=true)` [virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.46 `void brathl::COBMap::GetKeys (CStringArray & keys, bool bRemoveAll=true, bool bUnique=false)`
[virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.47 `void brathl::COBMap::GetKeys (CStringList & keys, bool bRemoveAll=true, bool bUnique=false)`
[virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.48 `void brathl::COblntMap::GetKeys (CIntArray & keys, bool bRemoveAll=true)` [virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.49 `void brathl::CObDoubleMap::GetKeys (CDoubleArray & keys, bool bRemoveAll=true)` [virtual]

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.50 `void bratl::CDoublePtrDoubleMap::GetKeys (CDoubleArray & keys, bool bRemoveAll = true) [virtual]`

Gets keys of the std::map

Parameters

<i>keys</i>	[out] : the keys of the std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

6.2.4.51 `CStringArray * bratl::CArrayStringMap::Insert (const std::string & key, const CStringArray & str, bool withExcept = true) [virtual]`

Inserts a std::string

Parameters

<i>key</i>	: std::map key
<i>str</i>	: std::string value

Returns

the inserted std::string value or existing std::string value if key exists

6.2.4.52 `std::string bratl::CStringMap::Insert (const std::string & key, const std::string & str, bool withExcept = true) [virtual]`

Inserts a std::string

Parameters

<i>key</i>	: std::map key
<i>str</i>	: std::string value

Returns

the inserted std::string value or existing std::string value if key exists

Referenced by bratl::CStringMap::Insert().

6.2.4.53 `void bratl::CStringMap::Insert (const CStringMap & strmap, bool withExcept = true) [virtual]`

Inserts a std::string std::map

Parameters

<i>strmap</i>	: std::map to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

the inserted std::string value or existing std::string value if key exists

References bratl::CStringMap::Insert().

6.2.4.54 `int32_t bratl::CIntMap::Insert (const std::string & key, int32_t value, bool withExcept = true) [virtual]`

Inserts an integer

Parameters

<i>key</i>	: std::map key
<i>value</i>	: int value

Returns

the inserted integer value or existing integer value if key exists

Referenced by `brathl::CIntMap::Insert()`.

6.2.4.55 `void brathl::CIntMap::Insert (const CIntMap & m, bool bRemoveAll = true, bool withExcept = true)`
`[virtual]`

Inserts a **CIntMap** (p. 234)

Parameters

<i>std::map</i>	[in]: std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

References `brathl::CIntMap::Insert()`, and `brathl::CIntMap::RemoveAll()`.

6.2.4.56 `uint32_t brathl::CUIntMap::Insert (const std::string & key, uint32_t value, bool withExcept = true)` `[virtual]`

Inserts an integer

Parameters

<i>key</i>	: std::map key
<i>value</i>	: int value

Returns

the inserted integer value or existing unsigned integer value if key exists

Referenced by `brathl::CUIntMap::Insert()`.

6.2.4.57 `void brathl::CUIntMap::Insert (const CUIntMap & m, bool bRemoveAll = true, bool withExcept = true)`
`[virtual]`

Inserts a **CUIntMap** (p. 309)

Parameters

<i>std::map</i>	[in]: std::map
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

References `brathl::CUIntMap::Insert()`, and `brathl::CUIntMap::RemoveAll()`.

6.2.4.58 `void brathl::CUIntMap::Insert (const CStringArray & keys, uint32_t initValue, bool bRemoveAll = true, bool withExcept = true)` `[virtual]`

Inserts a `CStrinArray` as keys and initial value

Parameters

<i>keys</i>	[in]: std::map keys to insert
<i>initValue</i>	[in]: value of the keys

<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys
-------------------	---

References `brathl::CUIntMap::Insert()`, and `brathl::CUIntMap::RemoveAll()`.

6.2.4.59 `void brathl::CUIntMap::Insert (const CStringArray & keys, const CUIntArray & values, bool bRemoveAll = true, bool withExcept = true) [virtual]`

Inserts a `CStrinArray` as keys and a `CUIntArray` as value

Parameters

<i>keys</i>	[in]: keys to insert
<i>values</i>	[in]: values to insert
<i>bRemoveAll</i>	[in] : if true, remove keys array element before filling the keys

References `brathl::CTools::Format()`, `brathl::CUIntMap::Insert()`, and `brathl::CUIntMap::RemoveAll()`.

6.2.4.60 `double brathl::CDoubleMap::Insert (const std::string & key, double value, bool withExcept = true) [virtual]`

Inserts an double

Parameters

<i>key</i>	: std::map key
<i>value</i>	: double value

Returns

the inserted double value or existing double value if key exists

6.2.4.61 `CBratObject * brathl::CObMap::Insert (const std::string & key, CBratObject * ob, bool withExcept = true) [virtual]`

Inserts a `CBratObject` object

Parameters

<i>key</i>	: CBratObject name (std::map key)
<i>value</i>	: CBratObject value
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

`CBratObject` object or `NULL` if error

Referenced by `brathl::CObMap::Insert()`, `brathl::CDataSet::InsertFieldSet()`, and `brathl::CObMap::RenameKey()`.

6.2.4.62 `void brathl::CObMap::Insert (const CObMap & obMap, bool withExcept = true) [virtual]`

Inserts a **CObMap** (p. 241)

Parameters

<i>obMap</i>	: CObMap (p. 241) to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

References `brathl::CObMap::Insert()`.

6.2.4.63 `CBratObject * brathl::COblntMap::Insert (int32_t key, CBratObject * ob, bool withExcept = true) [virtual]`

Inserts a `CBratObject` object

Parameters

<i>key</i>	: CBratObject name (std::map key)
<i>value</i>	: CBratObject value
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

CBratObject object or NULL if error

Referenced by brathl::COBIntMap::Insert(), and brathl::COBIntMap::RenameKey().

6.2.4.64 void brathl::COBIntMap::Insert (const COBIntMap & *obMap*, bool *withExcept* = true) [virtual]

Inserts a COBIntMap (p. 239)

Parameters

<i>obMap</i>	: COBMap (p. 241) to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

References brathl::COBIntMap::Insert().

6.2.4.65 CBratObject * brathl::COBDoubleMap::Insert (double *key*, CBratObject * *ob*, bool *withExcept* = true) [virtual]

Inserts a CBratObject object

Parameters

<i>key</i>	: CBratObject name (std::map key)
<i>value</i>	: CBratObject value
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

CBratObject object or NULL if error

Referenced by brathl::COBDoubleMap::Insert(), and brathl::COBDoubleMap::RenameKey().

6.2.4.66 void brathl::COBDoubleMap::Insert (const COBDoubleMap & *obMap*, bool *withExcept* = true) [virtual]

Inserts a COBDoubleMap (p. 238)

Parameters

<i>obMap</i>	: COBMap (p. 241) to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

References brathl::COBDoubleMap::Insert().

6.2.4.67 DoublePtr * brathl::CDBoublePtrDoubleMap::Insert (double *key*, DoublePtr * *ob*, bool *withExcept* = true) [virtual]

Inserts a DoublePtr* object

Parameters

<i>key</i>	: DoublePtr* name (std::map key)
------------	----------------------------------

<i>value</i>	: DoublePtr* value
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

DoublePtr* object or NULL if error

Referenced by brathl::CDoublePtrDoubleMap::RenameKey().

6.2.4.68 void * brathl::CPtrMap::Insert (const std::string & *key*, void * *ptr*, bool *withExcept* = true) [virtual]

Inserts a pointer

Parameters

<i>key</i>	: keymap
<i>value</i>	: pointer value
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

pointer or NULL if error

Referenced by brathl::CPtrMap::Insert().

6.2.4.69 void brathl::CPtrMap::Insert (const CPtrMap & *ptrMap*, bool *withExcept* = true) [virtual]

Inserts a **CPtrMap** (p. 275)

Parameters

<i>obMap</i>	: CPtrMap (p. 275) to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

References brathl::CPtrMap::Insert().

6.2.4.70 std::string brathl::CStringMap::IsValue (const std::string & *value*) [virtual]

Tests if an element value exists

Returns

a std::string key corresponding to the value (or the first key found, if some values are the same); if exists, otherwise empty std::string

6.2.4.71 const CStringList & brathl::CStringList::operator= (const CStringList & *lst*) [virtual]

Copy a new **CStringList** (p. 278) to the object

6.2.4.72 const CIntList & brathl::CIntList::operator= (const CIntList & *lst*)

Copy a new **CIntList** (p. 233) to the object

6.2.4.73 const CObList & brathl::CObList::operator= (const CObList & *lst*) [virtual]

Copy a new **CStringList** (p. 278) to the object

References brathl::CObList::RemoveAll().

6.2.4.74 const CObArray & brathl::CObArray::operator= (const CObArray & *lst*) [virtual]

Copy a new **CObArray** (p. 237) to the object

References brathl::CObArray::RemoveAll().

6.2.4.75 `int32_t brathl::CIntMap::operator[] (const std::string & key)` [virtual]

`operator[]` redefinition. Searches an integer value identify by '*key*'.

Parameters

<i>key</i>	: std::string keyword
------------	-----------------------

Returns

the interger value if found, default value **CTools::m_defaultValueINT32** (p. 283) if not found

References bratl::CIntMap::Exists().

6.2.4.76 `uint32_t bratl::CUIntMap::operator[] (const std::string & key) [virtual]`

operator[] redefinition. Searches an integer value identifiy by 'key'.

Parameters

<i>key</i>	: std::string keyword
------------	-----------------------

Returns

the interger value if found, default value **CTools::m_defaultValueUINT32** (p. 283) if not found

References bratl::CUIntMap::Exists().

6.2.4.77 `double bratl::CDoubleMap::operator[] (const std::string & key) [virtual]`

operator[] redefinition. Searches an integer value identifiy by 'key'.

Parameters

<i>key</i>	: std::string keyword
------------	-----------------------

Returns

the double value if found, default value **CTools::m_defaultValueDOUBLE** (p. 283) if not found

References bratl::CDoubleMap::Exists().

6.2.4.78 `CBratObject * bratl::CObMap::operator[] (const std::string & key) [virtual]`

operator[] redefinition. Searches a CBratObject object identifiy by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object CBratObject *o = myMap[key] -> use Exists method instead ;

Parameters

<i>key</i>	: CBratObject keyword
------------	-----------------------

Returns

a pointer to the CBratObject object if found, NULL if not found

6.2.4.79 `CBratObject * bratl::COblntMap::operator[] (int32_t key) [virtual]`

operator[] redefinition. Searches a CBratObject object identifiy by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object CBratObject *o = myMap[key] -> use Exists method instead ;

Parameters

<i>key</i>	: CBratObject keyword
------------	-----------------------

Returns

a pointer to the CBratObject object if found, NULL if not found

6.2.4.80 CBratObject * brathl::CObDoubleMap::operator[] (double key) [virtual]

operator[] redefinition. Searches a CBratObject object identify by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object CBratObject *o = myMap[key] → use Exists method instead ;

Parameters

<i>key</i>	: CBratObject keyword
------------	-----------------------

Returns

a pointer to the CBratObject object if found, NULL if not found

6.2.4.81 DoublePtr * brathl::CDoublePtrDoubleMap::operator[] (double key) [virtual]

operator[] redefinition. Searches a CBratObject object identify by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object CBratObject *o = myMap[key] → use Exists method instead ;

Parameters

<i>key</i>	: CBratObject keyword
------------	-----------------------

Returns

a pointer to the CBratObject object if found, NULL if not found

6.2.4.82 void * brathl::CPtrMap::operator[] (const std::string & key) [virtual]

operator[] redefinition. Searches a CBratObject object identify by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object void *p = myMap[key] → use Exists method instead ;

Parameters

<i>key</i>	: CBratObject keyword
------------	-----------------------

Returns

a pointer to the pointer if found, NULL if not found

6.2.4.83 void brathl::CObList::RemoveAll () [virtual]

Remove all elements and clear the std::list

Reimplemented in **brathl::CField::CListField** (p. 235).

Referenced by brathl::CObList::operator=(), brathl::CField::CListField::RemoveAll(), and brathl::CObList::~CObList().

6.2.4.84 void brathl::CDoublePtrArray::RemoveAll() [virtual]

Remove all elements and clear the std::list

Referenced by brathl::CDoublePtrArray::~CDoublePtrArray().

6.2.4.85 void brathl::CArrayDoublePtrArray::RemoveAll() [virtual]

Remove all elements and clear the std::list

Referenced by brathl::CArrayDoublePtrArray::~CArrayDoublePtrArray().

6.2.4.86 void brathl::CArrayDoubleArray::RemoveAll() [virtual]

Remove all elements and clear the std::list

Referenced by brathl::CArrayDoubleArray::~CArrayDoubleArray().

6.2.4.87 void brathl::CArrayStringMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

6.2.4.88 void brathl::CObStack::RemoveAll() [virtual]

Remove all elements and clear the std::list

References brathl::CObStack::m_bDelete.

Referenced by brathl::CObStack::~CObStack().

6.2.4.89 void brathl::CObArray::RemoveAll() [virtual]

Remove all elements and clear the std::list

Reimplemented in **brathl::CDataSet** (p. 170).

Referenced by brathl::CObArray::operator=(), brathl::CDataSet::RemoveAll(), and brathl::CObArray::~CObArray().

6.2.4.90 void brathl::CStringMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CStringMap::~CStringMap().

6.2.4.91 void brathl::CIntMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CIntMap::Insert(), and brathl::CIntMap::~CIntMap().

6.2.4.92 void brathl::CUIntMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CUIntMap::Insert(), and brathl::CUIntMap::~CUIntMap().

6.2.4.93 void brathl::CDoubleMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CDoubleMap::~CDoubleMap().

6.2.4.94 void brathl::CObMap::RemoveAll() [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CDataSet::RemoveAll(), and brathl::CObMap::~CObMap().

6.2.4.95 void brathl::COblntMap::RemoveAll () [virtual]

Remove all elements and clear the std::map

Referenced by brathl::COblntMap::~~COblntMap().

6.2.4.96 void brathl::CObDoubleMap::RemoveAll () [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CObDoubleMap::~~CObDoubleMap().

6.2.4.97 void brathl::CDoublePtrDoubleMap::RemoveAll () [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CDoublePtrDoubleMap::~~CDoublePtrDoubleMap().

6.2.4.98 void brathl::CPtrMap::RemoveAll () [virtual]

Remove all elements and clear the std::map

Referenced by brathl::CPtrMap::~~CPtrMap().

6.2.4.99 bool brathl::CObMap::RenameKey (const std::string & *oldKey*, const std::string & *newKey*)

Rename a key

Parameters

<i>oldKey</i>	: old key
<i>newKey</i>	: new key

Returns

true if key is renamed, otherwise false

References brathl::CObMap::Insert().

6.2.4.100 bool brathl::COblntMap::RenameKey (int32_t *oldKey*, int32_t *newKey*)

Rename a key

Parameters

<i>oldKey</i>	: old key
<i>newKey</i>	: new key

Returns

true if key is renamed, otherwise false

References brathl::COblntMap::Insert().

6.2.4.101 bool brathl::CObDoubleMap::RenameKey (double *oldKey*, double *newKey*)

Rename a key

Parameters

<i>oldKey</i>	: old key
<i>newKey</i>	: new key

Returns

true if key is renamed, otherwise false

References bratl::CObDoubleMap::Insert().

6.2.4.102 `bool bratl::CDoublePtrDoubleMap::RenameKey (double oldKey, double newKey)`

Rename a key

Parameters

<i>oldKey</i>	: old key
<i>newKey</i>	: new key

Returns

true if key is renamed, otherwise false

References bratl::CDoublePtrDoubleMap::Insert().

6.2.4.103 `void bratl::CArrayStringMap::Set (const CArrayStringMap & a) [virtual]`

Inserts a std::string std::map

Parameters

<i>strmap</i>	: std::map to insert
<i>withExcept</i>	: true for exception handling, flse otherwise

Returns

the inserted std::string value or existing std::string value if key exists

6.2.5 Variable Documentation

6.2.5.1 `const std::string bratl::UNKNOWN_PRODUCT_CLASS = "UNKNOWN"`

External files access.

Version

1.0

6.3 Criteria

Classes

- class **brathl::CCriteria**
- class **brathl::CCriteriaCycle**
- class **brathl::CCriteriaCycleInfo**
- class **brathl::CCriteriaDatetime**
- class **brathl::CCriteriaDatetimeInfo**
- class **brathl::CCriterialInfo**
- class **brathl::CCriteriaLatLon**
- class **brathl::CCriteriaLatLonInfo**
- class **brathl::CCriteriaPass**
- class **brathl::CCriteriaPassInfo**
- class **brathl::CCriteriaPassInt**
- class **brathl::CCriteriaPassIntInfo**
- class **brathl::CCriteriaPassString**
- class **brathl::CCriteriaPassStringInfo**
- class **brathl::CDataSet**
- class **brathl::CField**
- class **brathl::CFieldArray**
- class **brathl::CFieldBasic**
- class **brathl::CFieldIndexData**
- class **brathl::CFieldNetCdf**
- class **brathl::CFieldNetCdfCF**
- class **brathl::CFieldNetCdfCFAttr**
- class **brathl::CFieldRecord**
- class **brathl::CFieldSet**
- class **brathl::CFieldSetArrayDbl**
- class **brathl::CFieldSetDbl**
- class **brathl::CFieldSetString**
- class **brathl::CProduct::CInfo**
- class **brathl::CField::CListField**
- class **brathl::CProduct::CListInfo**
- class **brathl::CMapProduct**
- class **brathl::CProductAop**
- class **brathl::CProductCryosat**
- class **brathl::CProductEnvisat**
- class **brathl::CProductEnvisatNetCdf**
- class **brathl::CProductErs**
- class **brathl::CProductErsWAP**
- class **brathl::CProductGeosatGDR**
- class **brathl::CProductGfo**
- class **brathl::CProductJason**
- class **brathl::CProductJason1NetCdf**
- class **brathl::CProductJason2**
- class **brathl::CProductList**
- class **brathl::CProductNetCdf**
- class **brathl::CProductNetCdfCF**
- class **brathl::CProductPodaac**
- class **brathl::CProductRads**
- class **brathl::CProductReaper**
- class **brathl::CProductRiverLake**
- class **brathl::CProductTopex**
- class **brathl::CProductTopexSDR**
- class **brathl::CRecord**
- class **brathl::CRecordSet**
- class **brathl::CTreeField**

Enumerations

- enum **bratl::CCriteria::CriteriaKind** {
bratl::CCriteria::UNKNOWN, **bratl::CCriteria::LATLON**, **bratl::CCriteria::DATETIME**, **bratl::C**↵
Criteria::PASS,
bratl::CCriteria::CYCLE }

Functions

- void **bratl::CProduct::AddCriteria** (bool force=false)
- void **bratl::CProduct::AddCriteria** (CCriteria *criteria, bool erase=true)
- void **bratl::CProduct::AddCriteria** (CProduct *product)
- void **bratl::CMapProduct::AddCriteriaToProducts** ()
- virtual void **bratl::CProduct::AddInternalHighResolutionFieldCalculation** ()
- CInfo * **bratl::CProduct::CListInfo::AddNew** ()
- virtual void **bratl::CProduct::AddOffset** (double value, CField *field=NULL)
- bool **bratl::CProduct::AddRecordNameToField** (const CExpression &expr, const std::string &dataSet↵
Name, CExpression &exprOut, std::string &errorMsg)
- bool **bratl::CProduct::AddRecordNameToField** (const std::string &in, const std::string &dataSetName,
std::string &out, std::string &errorMsg)
- bool **bratl::CProduct::AddRecordNameToField** (const std::string &in, const std::string &dataSetName,
const CStringArray &fieldsIn, std::string &out, std::string &errorMsg)
- bool **bratl::CProduct::AddRecordNameToField** (CProductAliases *productAliases, std::string &errorMsg)
- virtual void **bratl::CProduct::AddSameFieldName** (const std::string &fieldNameToSearch, CStringArray
&arrayFieldsAdded)
- void **bratl::CCriteriaPassInt::Adjust** ()
- virtual bool **bratl::CProduct::ApplyCriteria** (CStringList &filteredFileList, CProgressInterface *pi, const
std::string &log_file="")
- virtual bool **bratl::CProduct::ApplyCriteriaCycle** (CCriterialInfo *criterialInfo)
- virtual bool **bratl::CProduct::ApplyCriteriaDatetime** (CCriterialInfo *criterialInfo)
- virtual bool **bratl::CProduct::ApplyCriteriaLatLon** (CCriterialInfo *criterialInfo)
- virtual bool **bratl::CProduct::ApplyCriteriaPass** (CCriterialInfo *criterialInfo)
- virtual bool **bratl::CProduct::ApplyCriteriaPassInt** (CCriterialInfo *criterialInfo)
- virtual bool **bratl::CProduct::ApplyCriteriaPassString** (CCriterialInfo *criterialInfo)
- CInfo * **bratl::CProduct::CListInfo::Back** (bool withExcept=true)
- void **bratl::CProduct::BuildCriteriaFieldsToRead** (CRecordDataMap &listRecord)
- **bratl::CCriteriaPass::CCriteriaPass** ()
Empty *CCriteriaPass* (p. 162) ctor.
- **bratl::CCriteriaPassInt::CCriteriaPassInt** ()
Empty *CCriteriaPassInt* (p. 164) ctor.
- **bratl::CCriteriaPassInt::CCriteriaPassInt** (CCriteriaPassInt &c)
- **bratl::CCriteriaPassInt::CCriteriaPassInt** (CCriteriaPassInt *c)
- **bratl::CCriteriaPassInt::CCriteriaPassInt** (int32_t from, int32_t to)
- **bratl::CCriteriaPassInt::CCriteriaPassInt** (const std::string &from, const std::string &to)
- **bratl::CCriteriaPassInt::CCriteriaPassInt** (const CStringArray &array)
- **bratl::CCriteriaPassString::CCriteriaPassString** ()
Empty *CCriteriaPassString* (p. 166) ctor.
- **bratl::CCriteriaPassString::CCriteriaPassString** (CCriteriaPassString &c)
- **bratl::CCriteriaPassString::CCriteriaPassString** (CCriteriaPassString *c)
- **bratl::CCriteriaPassString::CCriteriaPassString** (const std::string &passes, const std::string &delimiter=C↵
CriteriaPassString::m_delimiter)
- **bratl::CCriteriaPassString::CCriteriaPassString** (const CStringArray &array)
- static bool **bratl::CProduct::CheckAliases** (const std::string &fileName, CStringArray &errors)
- bool **bratl::CProduct::CheckAliases** (CStringArray &errors)

- virtual void **brathl::CProduct::CheckConsistencyHighResolutionField** (CFieldSetArrayDbI *fieldSet↵
ArrayDbI)
- bool **brathl::CProduct::CheckFieldNames** (const CExpression &expr, const std::string &dataSetName, C↵
StringArray &fieldNamesNotFound)
- bool **brathl::CProduct::CheckFieldNames** (const CExpression &expr, CStringArray &fieldNamesNotFound)
- bool **brathl::CProduct::CheckFieldNames** (const CStringArray *fieldNames, const std::string &dataSet↵
Name, CStringArray &fieldNamesNotFound)
- void **brathl::CProduct::CheckFields** (bool convertDate=false)
- bool **brathl::CProductList::CheckFile** (const stringlist::iterator &it, bool netcdf_check)
- virtual void **brathl::CProduct::CheckFileOpened** ()
- bool **brathl::CProductList::CheckFiles** (bool onlyFirstFile=false, bool onlyFirstNetcdf=false)
- virtual CProduct * **brathl::CProduct::Clone** ()
- virtual bool **brathl::CProduct::Close** ()
- **brathl::CMapProduct::CMapProduct** ()
CIntMap (p. 234) ctor.
- static void **brathl::CProduct::Codalnit** ()
- static void **brathl::CProduct::CodaRelease** ()
- static CProduct * **brathl::CProduct::Construct** (const CProductList &fileNameList)
- static CProduct * **brathl::CProduct::Construct** (CProductList &fileNameList, bool check_only_first_↵
file=false)
- static CProduct * **brathl::CProduct::Construct** (const CStringArray &fileNameArray, bool check_only_first_↵
_file=false)
- static CProduct * **brathl::CProduct::Construct** (const std::string &fileName)
- void **brathl::CProduct::ConvertDate** (CDoubleArray &vect)
- **brathl::CProduct::CProduct** (const std::string &fileName)
- **brathl::CProduct::CProduct** (const CStringList &fileNameList, bool check_only_first_file)
- **brathl::CProductGeneric::CProductGeneric** ()
Empty CProductGeneric ctor.
- **brathl::CProductGeneric::CProductGeneric** (const std::string &fileName)
- **brathl::CProductGeneric::CProductGeneric** (const CStringList &fileNameList, bool check_only_first_file)
- **brathl::CProductList::CProductList** ()
Empty CProductList (p. 259) ctor.
- **brathl::CProductList::CProductList** (const CProductList &o)
- **brathl::CProductList::CProductList** (const std::string &fileName)
- **brathl::CProductList::CProductList** (const CStringList &fileNameList)
- **brathl::CProductList::CProductList** (const CStringArray &fileNameArray)
- void **brathl::CProduct::CreateFieldIndexData** ()
- void **brathl::CProduct::CreateFieldIndexes** (CFieldArray *field)
- void **brathl::CProduct::CreateLogFile** (const std::string &log_file, uint32_t mode=CFile::modeWrite|CFile↵
::typeText)
- std::string **brathl::CProduct::DatasetRecordsNumberToString** (const CIntMap &datasetRecordsNumber)
- void **brathl::CProduct::DeleteLogFile** ()
- virtual void **brathl::CCriteriaPass::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **brathl::CProductList::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **brathl::CCriteriaPassString::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **brathl::CCriteriaPassInt::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **brathl::CProduct::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **brathl::CMapProduct::Dump** (std::ostream &fOut=std::cerr)
- void **brathl::CProduct::DumpDictionary** (std::ostream &fOut=std::cout)

- void **brathl::CProduct::DumpDictionary** (const std::string &outputFileName)
- virtual void **brathl::CProduct::EndApplyCriteriaStats** (const **CStringList** &filteredFileList)
- void **brathl::CProduct::ExpandFieldsArray** ()
- virtual void **brathl::CProduct::ExtractDatasetNamesFromFields** (const **CStringList** &listFields, **CStringList** &datasetNames)
- static void **brathl::CCriteriaPassString::ExtractPass** (const std::string &passes, CStringArray &arrayPass, const std::string &delimiter=CCriteriaPassString::m_delimiter)
- static void **brathl::CCriteriaPassString::ExtractPass** (const CStringArray &array, CStringArray &arrayPass)
- virtual void **brathl::CProduct::FillDescription** ()
- void **brathl::CProduct::FillListFields** (const std::string &key)
- **CField** * **brathl::CProduct::FindFieldByInternalName** (const std::string &internalFieldName, bool withExcept=true)
- **CField** * **brathl::CProduct::FindFieldByName** (const std::string &fieldName, const std::string &dataSetName, bool withExcept=true, std::string *errorMsg=NULL, bool showTrace=true)
- **CField** * **brathl::CProduct::FindFieldByName** (const std::string &fieldName, bool withExcept=true, std::string *errorMsg=NULL, bool showTrace=true)
- virtual bool **brathl::CProduct::FindParentToRead** (**CField** *fromField, **CObList** *parentFieldList)
- **CInfo** * **brathl::CProduct::CListInfo::Front** (bool withExcept=true)
- const **CProductAlias** * **brathl::CProduct::GetAlias** (const std::string &key)
- const **CProductAliases** * **brathl::CProduct::GetAliases** ()
- const **CStringMap** * **brathl::CProduct::GetAliasesAsString** () const
- static const **CStringMap** * **brathl::CProduct::GetAliasesAsString** (const **CProduct** *product)
- std::string **brathl::CProduct::GetAliasExpandedValue** (const std::string &key)
- void **brathl::CProduct::GetAliasKeys** (CStringArray &keys)
- std::string **brathl::CCriteriaPassString::GetAsText** (const std::string &delimiter=CCriteriaPassString::m_delimiter)
- std::string **brathl::CCriteriaPassInt::GetAsText** (const std::string &delimiter=CCriteriaPassInt::m_delimiter)
- bool **brathl::CProduct::GetCreateVirtualField** ()
- static **CCriteriaPass** * **brathl::CCriteriaPass::GetCriteria** (**CBratObject** *ob, bool withExcept=true)
- static **CCriteriaPassString** * **brathl::CCriteriaPassString::GetCriteria** (**CBratObject** *ob, bool withExcept=true)
- static **CCriteriaPassInt** * **brathl::CCriteriaPassInt::GetCriteria** (**CBratObject** *ob, bool withExcept=true)
- **CCriteria** * **brathl::CProduct::GetCriteria** (**CCriterialInfo** *criterialInfo)
- virtual std::string **brathl::CProduct::GetCurrentFileName** ()
- virtual int32_t **brathl::CProduct::GetCurrentRecordNumber** ()
- **CCriteriaCycle** * **brathl::CProduct::GetCycleCriteria** ()
- **CCriteriaCycleInfo** * **brathl::CProduct::GetCycleCriterialInfo** ()
- **CStringArray** * **brathl::CProduct::GetDataDictionaryFieldNames** (bool forceReload=false)
- **CStringArray** * **brathl::CProduct::GetDataDictionaryFieldNamesWithDatasetName** (bool forceReload=false)
- **CDataSet** * **brathl::CProduct::GetDataSet** ()
- std::string **brathl::CProduct::GetDataSetNameToRead** ()
- virtual bool **brathl::CProduct::GetDateMinMax** (**CDatePeriod** &datePeriodMinMax, **CProgressInterface** *pi=NULLptr)
- virtual bool **brathl::CProduct::GetDateMinMax** (**CDate** &dateMin, **CDate** &dateMax)
- **CCriteriaDatetime** * **brathl::CProduct::GetDatetimeCriteria** ()
- **CCriteriaDatetimeInfo** * **brathl::CProduct::GetDatetimeCriterialInfo** ()
- const std::string & **brathl::CProduct::GetDescription** ()
- bool **brathl::CProduct::GetDisableTrace** ()
- bool **brathl::CProduct::GetExpandArray** ()
- std::string **brathl::CProduct::GetFieldSpecificUnit** (const std::string &key)
- **CStringMap** * **brathl::CProduct::GetFieldSpecificUnits** ()
- **CStringArray** * **brathl::CProduct::GetFieldToTranspose** ()
- double **brathl::CProduct::GetForceLatMaxCriteriaValue** ()
- double **brathl::CProduct::GetForceLatMinCriteriaValue** ()

- virtual bool **bratl::CProduct::GetForceReadDataOneByOne** ()
- int32_t **bratl::CCriteriaPassInt::GetFrom** ()
- int_t **bratl::CProduct::GetIndexProcessedFile** ()
- bool **bratl::CProduct::GetInfoArray** ()
- bool **bratl::CProduct::GetInfoRecord** (int32_t nbDims=1, const long dim[]=DEFAULT_DIM)
- bool **bratl::CProduct::GetInfoSpecial** (int32_t nbDims=1, const long dim[]=DEFAULT_DIM)
- static **CMapProduct** & **bratl::CMapProduct::GetInstance** ()
- virtual const std::string & **bratl::CProduct::GetLabel** () const
- virtual std::string **bratl::CProduct::GetLabelForCyclePass** () const
- virtual std::string **bratl::CProduct::GetLatitudeFieldName** ()
- **CCriteriaLatLon** * **bratl::CProduct::GetLatLonCriteria** ()
- **CCriteriaLatLonInfo** * **bratl::CProduct::GetLatLonCriteriaInfo** ()
- virtual bool **bratl::CProduct::GetLatLonMinMax** (double &latMin, double &lonMin, double &latMax, double &lonMax, CProgressInterface *pi=nullptr)
- virtual bool **bratl::CProduct::GetLatLonMinMax** (CLatLonRect &latlonRectMinMax, CProgressInterface *pi=nullptr)
- **CStringList** * **bratl::CProduct::GetListFieldOrigin** ()
- virtual std::string **bratl::CProduct::GetLongitudeFieldName** ()
- virtual void **bratl::CProduct::GetMinMaxNumberOfRecords** (int32_t &min, int32_t &max, **CIntMap** *datasetRecordsNumber=NULL, int32_t minThreshold=-1)
- void **bratl::CProduct::GetNamesCaseSensitive** (const CStringArray &fieldsIn, CStringArray &fieldsOut, NoCaseSensitive, CStringArray &fieldsOutCaseSensitive, bool forceReload=false)
- virtual int32_t **bratl::CProduct::GetNumberOfRecords** ()
- virtual int32_t **bratl::CProduct::GetNumberOfRecords** (const std::string &dataSetName)
- virtual void **bratl::CProduct::GetNumberOfRecords** (const **CStringList** &datasetNames, **CIntMap** &datasetRecordsNumber)
- virtual double **bratl::CProduct::GetOffset** ()
- **CCriteriaPass** * **bratl::CProduct::GetPassCriteria** ()
- **CCriteriaPassInfo** * **bratl::CProduct::GetPassCriteriaInfo** ()
- CStringArray * **bratl::CCriteriaPassString::GetPasses** ()
- **CCriteriaPassInt** * **bratl::CProduct::GetPassIntCriteria** ()
- **CCriteriaPassIntInfo** * **bratl::CProduct::GetPassIntCriteriaInfo** ()
- **CCriteriaPassString** * **bratl::CProduct::GetPassStringCriteria** ()
- **CCriteriaPassStringInfo** * **bratl::CProduct::GetPassStringCriteriaInfo** ()
- int32_t **bratl::CProduct::GetPerformBoundaryChecks** ()
- int32_t **bratl::CProduct::GetPerformConversions** ()
- const std::string & **bratl::CProduct::GetProductClass** () const
- std::string **bratl::CProduct::GetProductClassAndType** ()
- void **bratl::CMapProduct::GetProductKeysWithCriteria** (CStringArray &keys)
- **CProductList** & **bratl::CProduct::GetProductList** ()
- const std::string & **bratl::CProduct::GetProductType** () const
- std::string **bratl::CProduct::GetRecordFieldName** ()
- virtual void **bratl::CProduct::GetRecords** (CStringArray &array)
- static int_t **bratl::CProduct::GetRefCount** ()
- **bratl_refDate** **bratl::CProduct::GetRefDate** () const
- **CDate** **bratl::CProduct::GetRefDateAsDate** ()
- void **bratl::CProduct::GetRootType** ()
- uint32_t **bratl::CProduct::GetSkippedRecordCount** ()
- int32_t **bratl::CCriteriaPassInt::GetTo** ()
- **CTreeField** * **bratl::CProduct::GetTreeField** ()
- std::string **bratl::CProduct::GetTypeDesc** ()
- std::string **bratl::CProduct::GetTypeDesc** (coda_Type *type)
- std::string **bratl::CProduct::GetTypeName** ()
- std::string **bratl::CProduct::GetTypeUnit** ()

- virtual bool **brathl::CProduct::GetValueMinMax** (CExpression &expr, const std::string &recordName, double &valueMin, double &valueMax, const CUnit &unit, CProgressInterface *pi=nullptr)
- static void **brathl::CProduct::GroupAliases** (const CProduct *product, const **CStringMap** *formulaAliases, **CStringMap** &allAliases)
- void **brathl::CProduct::HandleBratError** (const std::string &str="", int32_t errClass=BRATHL_LOGIC_ERROR)
- virtual bool **brathl::CProduct::HasAliases** ()
- virtual bool **brathl::CProduct::HasCompatibleDims** (const std::string &value, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const std::string &value, const std::string &dataSetName, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CExpression &expr, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CExpression &expr, const std::string &dataSetName, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CStringArray *fieldNames, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CStringArray *fieldNames, const std::string &dataSetName, std::string &msg, bool useVirtualDims, CUIntArray *commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCriterialInfo** ()
- bool **brathl::CProduct::HasCycleCriteria** ()
- bool **brathl::CProduct::HasCycleCriterialInfo** ()
- bool **brathl::CProduct::HasDatetimeCriteria** ()
- bool **brathl::CProduct::HasDatetimeCriterialInfo** ()
- bool **brathl::CProduct::HasEqualDims** (const std::string &value, std::string &msg)
- bool **brathl::CProduct::HasEqualDims** (const std::string &value, const std::string &dataSetName, std::string &msg)
- bool **brathl::CProduct::HasEqualDims** (const CExpression &expr, std::string &msg)
- bool **brathl::CProduct::HasEqualDims** (const CExpression &expr, const std::string &dataSetName, std::string &msg)
- bool **brathl::CProduct::HasEqualDims** (const CStringArray *fieldNames, std::string &msg)
- bool **brathl::CProduct::HasEqualDims** (const CStringArray *fieldNames, const std::string &dataSetName, std::string &msg)
- bool **brathl::CProduct::HasEqualsNumberOfRecord** (const CIntMap &datasetRecordsNumber)
- virtual bool **brathl::CProduct::HasHighResolutionFieldCalculation** ()
- bool **brathl::CProduct::HasLatLonCriteria** ()
- bool **brathl::CProduct::HasLatLonCriterialInfo** ()
- bool **brathl::CProduct::HasPassCriteria** ()
- bool **brathl::CProduct::HasPassCriterialInfo** ()
- bool **brathl::CProduct::HasPassIntCriteria** ()
- bool **brathl::CProduct::HasPassIntCriterialInfo** ()
- bool **brathl::CProduct::HasPassStringCriteria** ()
- bool **brathl::CProduct::HasPassStringCriterialInfo** ()
- void **brathl::CCriteriaPass::Init** ()
- void **brathl::CCriteriaPassString::Init** ()
- void **brathl::CCriteriaPassInt::Init** ()
- void **brathl::CMapProduct::Init** ()
- virtual void **brathl::CProduct::InitApplyCriteriaStats** ()
- virtual void **brathl::CProduct::InitCriterialInfo** ()
- virtual void **brathl::CProduct::InitDateRef** ()=0
- virtual void **brathl::CProductGeneric::InitDateRef** ()
- virtual void **brathl::CProduct::InitInternalFieldName** (const std::string &dataSetName, **CStringList** &listField, bool convertDate=false)
- virtual void **brathl::CProduct::InitInternalFieldName** (**CStringList** &listField, bool convertDate=false)
- virtual void **brathl::CProduct::InitInternalFieldName** (const std::string &field, bool convertDate=false)

- virtual void **brathl::CProduct::InitInternalFieldNamesForCombinedVariable** (CStringList &listField, const std::string &record)
- void **brathl::CProduct::InsertRecord** (int32_t pos)
- void **brathl::CProduct::InsertRecord** (CDataSet &dataSet, int32_t pos)
- bool **brathl::CCriteriaPassString::Intersect** (const std::string &passes, CStringArray &intersect)
- bool **brathl::CCriteriaPassString::Intersect** (CStringArray &passes, CStringArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (CStringArray &array, CStringArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (CStringArray &array, CIntArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (CIntArray &array, CStringArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (CIntArray &array, CIntArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (int32_t from, int32_t to, CStringArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (int32_t from, int32_t to, CIntArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (double otherFrom, double otherTo, CIntArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (const std::string &from, const std::string &to, CIntArray &intersect)
- bool **brathl::CCriteriaPassInt::Intersect** (const std::string &from, const std::string &to, CStringArray &intersect)
- bool **brathl::CProductList::IsATP** () const
- virtual bool **brathl::CCriteriaPass::IsDefaultValue** ()=0
- bool **brathl::CCriteriaPassString::IsDefaultValue** ()
- bool **brathl::CCriteriaPassInt::IsDefaultValue** ()
- bool **brathl::CProductList::IsGenericNetCdf** () const
- bool **brathl::CProductList::IsHdfOrNetcdfCodaFormat** ()
- static bool **brathl::CProductList::IsHdfOrNetcdfCodaFormat** (coda_format format)
- virtual bool **brathl::CProduct::IsHighResolutionField** (CField *)
- bool **brathl::CProductList::IsJason2** () const
- virtual bool **brathl::CProduct::IsLatitudeFieldName** (const std::string &name) const
- virtual bool **brathl::CProduct::IsLongitudeFieldName** (const std::string &name) const
- bool **brathl::CProduct::IsNetCdf** ()
- bool **brathl::CProductList::IsNetCdfCFProduct** () const
- bool **brathl::CProduct::IsNetCdfCFProduct** ()
- bool **brathl::CProductList::IsNetCdfOrNetCdfCFProduct** () const
- bool **brathl::CProduct::IsNetCdfOrNetCdfCFProduct** ()
- bool **brathl::CProductList::IsNetCdfProduct** () const
- bool **brathl::CProduct::IsNetCdfProduct** ()
- virtual bool **brathl::CProduct::IsOpened** ()
- virtual bool **brathl::CProduct::IsOpened** (const std::string &fileName)
- bool **brathl::CProductList::IsSameProduct** (const std::string &productClass, const std::string &productType)
- bool **brathl::CProduct::IsSameProduct** (const CProductList &fileList)
- bool **brathl::CProduct::IsSameProduct** (const std::string &productClass, const std::string &productType)
- bool **brathl::CProduct::IsSetCycleCriteria** ()
- bool **brathl::CProduct::IsSetDatetimeCriteria** ()
- bool **brathl::CProduct::IsSetLatLonCriteria** ()
- bool **brathl::CProduct::IsSetPassCriteria** ()
- bool **brathl::CProduct::IsSetPassIntCriteria** ()
- bool **brathl::CProduct::IsSetPassStringCriteria** ()
- bool **brathl::CProductList::IsYFX** () const
- bool **brathl::CProductList::IsZFX** () const
- virtual void **brathl::CProduct::LoadAliases** ()
- virtual void **brathl::CProduct::LoadFieldsInfo** ()
- bool **brathl::CProduct::LoadTransposeFieldsValue** (CStringArray &fieldsToTranspose)
- template<typename T >
void **brathl::CProduct::Log** (const T n, bool bCrLf)
- void **brathl::CProduct::Log** (const char *str, bool bCrLf)
- void **brathl::CProduct::Log** (const std::string &str, bool bCrLf)

- void **brathl::CProduct::Log** (const bool n, bool bCrLf)
- void **brathl::CProduct::Log** (const **CStringList** &l, bool bCrLf)
- void **brathl::CProduct::LogSelectionResult** (const std::string &fileName, bool result)
- virtual std::string **brathl::CProduct::MakeInternalDataSetName** (const std::string &dataSetName)
- virtual std::string **brathl::CProduct::MakeInternalFieldName** (const std::string &dataSetName, const std::string &field)
- virtual std::string **brathl::CProduct::MakeInternalFieldName** (const std::string &field)
- virtual std::string **brathl::CProduct::MakeInternalNameByAddingRoot** (const std::string &name)
- virtual bool **brathl::CProduct::Open** (const std::string &fileName, const std::string &dataSetName, **CStringList** &listFieldToRead, bool convertDate=false)
- virtual bool **brathl::CProduct::Open** (const std::string &fileName, const std::string &dataSetName)
- virtual bool **brathl::CProduct::Open** (const std::string &fileName)
- virtual bool **brathl::CProduct::Open** ()
- **CProductList** & **brathl::CProductList::operator=** (const **CProductList** &lst)
- const **CCriteriaPassString** & **brathl::CCriteriaPassString::operator=** (**CCriteriaPassString** &c)
- const **CCriteriaPassInt** & **brathl::CCriteriaPassInt::operator=** (**CCriteriaPassInt** &c)
- **CInfo** * **brathl::CProduct::CListInfo::PrevBack** (bool withExcept=true)
- void **brathl::CProduct::ProcessHighResolution** ()
- virtual void **brathl::CProduct::ProcessHighResolutionWithFieldCalculation** ()
- virtual void **brathl::CProduct::ProcessHighResolutionWithoutFieldCalculation** ()
- virtual void **brathl::CProduct::Put** (**CDataSet** *dataSet, **CFieldSetDbI** *fieldSetDbI, uint32_t repeat, uint32_t insertRecordAt=0)
- virtual void **brathl::CProduct::Put** (**CDataSet** *dataSet, **CFieldSetArrayDbI** *fieldSetArrayDbI, uint32_t repeat, uint32_t insertRecordAt=0)
- virtual void **brathl::CProduct::Put** (**CDataSet** *dataSet, **CFieldSetDbI** *fieldSetDbI)
- virtual void **brathl::CProduct::PutFlat** (**CDataSet** *dataSet, **CFieldSetArrayDbI** *fieldSetArrayDbI, uint32_t insertRecordAt=0)
- virtual void **brathl::CProduct::PutFlatHighResolution** (**CDataSet** *dataSet, **CFieldSetArrayDbI** *fieldSetArrayDbI)
- virtual void **brathl::CProduct::ReadBratFieldRecord** (const std::string &key, int32_t iRecord)
- virtual void **brathl::CProduct::ReadBratFieldRecord** (**CField::CListField::iterator** it)
- virtual void **brathl::CProduct::ReadBratFieldRecord** (**CField::CListField::iterator** it, bool &skipRecord)
- virtual void **brathl::CProduct::ReadBratRecord** (const std::string &dataSetName, const std::string &field, int32_t iRecord)
- virtual void **brathl::CProduct::ReadBratRecord** (const std::string &dataSetName, **CStringList** &listField, int32_t iRecord)
- virtual void **brathl::CProduct::ReadBratRecord** (int32_t iRecord)
- static int32_t **brathl::CProduct::ReadData** (int32_t nbFiles, char **fileNames, const char *recordName, const char *selection, int32_t nbData, char **dataExpressions, char **units, double **results, int32_t sizes[], size_t *actualSize, int ignoreOutOfRange, int statistics, double defaultValue, **CStringMap** *fieldSpecificUnit=NULL)
- static void **brathl::CProduct::ReadDataForOneMeasure** (**CDataSet** *dataSet, const std::string &recordName, **CExpression** &Select, std::vector< **CExpression** > &Expressions, const std::vector< **CUnit** > &WantedUnits, double **results, int32_t *sizes, size_t *actualSize, int ignoreOutOfRange, int statistics, **CProduct** *product=NULL)
- void **brathl::CProduct::RemoveCriteria** ()
- void **brathl::CMapProduct::RemoveCriteriaFromProducts** ()
- void **brathl::CProduct::RemoveUnusedFields** ()
- void **brathl::CProduct::ReplaceNamesCaseSensitive** (const **CExpression** &exprIn, const **CStringArray** &fieldsIn, **CExpression** &exprOut, bool forceReload=false)
- void **brathl::CProduct::ReplaceNamesCaseSensitive** (const std::string &in, const **CStringArray** &fieldsIn, std::string &out, bool forceReload=false)
- void **brathl::CProduct::ReplaceNamesCaseSensitive** (const std::string &in, std::string &out, bool forceReload=false)
- void **brathl::CProduct::ReplaceNamesCaseSensitive** (const **CExpression** &exprIn, std::string &out, bool forceReload=false)

- virtual void **brathl::CProduct::Rewind** ()
- virtual void **brathl::CProduct::RewindEnd** ()
- virtual void **brathl::CProduct::RewindInit** ()
- virtual void **brathl::CProduct::RewindProcess** ()
- void **brathl::CCriteriaPassString::Set** (const std::string &passes, const std::string &delimiter=CCriteriaPassString::m_delimiter)
- void **brathl::CCriteriaPassString::Set** (const CStringArray &array)
- void **brathl::CCriteriaPassString::Set** (CCriteriaPassString &c)
- void **brathl::CCriteriaPassInt::Set** (CCriteriaPassInt &c)
- void **brathl::CCriteriaPassInt::Set** (int32_t from, int32_t to)
- void **brathl::CCriteriaPassInt::Set** (const std::string &from, const std::string &to)
- void **brathl::CCriteriaPassInt::Set** (const CStringArray &array)
- void **brathl::CProduct::SetCreateVirtualField** (bool value)
- void **brathl::CProduct::SetCursor** (CField *field, bool &skipRecord)
- void **brathl::CProduct::SetDataSetNameToRead** (const std::string &value)
- virtual void **brathl::CCriteriaPass::SetDefaultValue** ()=0
- void **brathl::CCriteriaPassString::SetDefaultValue** ()
- void **brathl::CCriteriaPassInt::SetDefaultValue** ()
- void **brathl::CProduct::SetDescription** (const std::string &value)
- void **brathl::CProduct::SetDisableTrace** (bool value)
- void **brathl::CProduct::SetDynInfo** ()
- void **brathl::CProduct::SetExpandArray** (bool value)
- void **brathl::CProduct::SetFieldSpecificUnit** (const std::string &key, const std::string &value)
- virtual void **brathl::CProduct::SetFieldSpecificUnit** (CField *field)
- void **brathl::CProduct::SetFieldSpecificUnits** (const CStringMap &fieldSpecificUnit)
- virtual void **brathl::CProduct::SetForceReadDataOneByOne** (bool)
- void **brathl::CCriteriaPassInt::SetFrom** (int32_t from)
- void **brathl::CCriteriaPassInt::SetFrom** (const std::string &from)
- void **brathl::CCriteriaPassInt::SetFromText** (const std::string &values, const std::string &delimiter=CCriteriaPassInt::m_delimiter)
- virtual void **brathl::CProduct::SetHighResolution** (CField *field)
- void **brathl::CProduct::SetIndex** (CField *field)
- void **brathl::CProduct::SetListFieldOrigin** (const CStringList &listFieldOrigin)
- void **brathl::CProduct::SetListFieldToRead** (CStringList &listFieldToRead, bool convertDate=false)
- void **brathl::CProduct::SetNativeType** (CField *field)
- virtual void **brathl::CProduct::SetOffset** (double value)
- void **brathl::CProduct::SetPerformBoundaryChecks** (bool performBoundaryChecks)
- void **brathl::CProduct::SetPerformConversions** (bool performConversions)
- void **brathl::CProduct::SetSpecialType** (CField *field)
- void **brathl::CCriteriaPassInt::SetTo** (int32_t to)
- void **brathl::CCriteriaPassInt::SetTo** (const std::string &to)
- void **brathl::CProduct::SetTypeClass** (CField *field)
- void **brathl::CProduct::SetUnion** (CField *field)
- bool **brathl::CProduct::TraverseData** ()
- bool **brathl::CProduct::TraverseRecord** (int32_t indexFields)
- virtual **brathl::CCriteriaPass::~CCriteriaPass** ()
- *Destructor.*
- virtual **brathl::CCriteriaPassInt::~CCriteriaPassInt** ()
- *Destructor.*
- virtual **brathl::CCriteriaPassString::~CCriteriaPassString** ()
- *Destructor.*
- virtual **brathl::CMapProduct::~CMapProduct** ()
- *CIntMap (p. 234) dtor.*
- virtual **brathl::CProductGeneric::~CProductGeneric** ()
- *Destructor.*
- virtual **brathl::CProductList::~CProductList** ()
- *Destructor.*

Variables

- static const uint32_t **brathl::CProduct::COUNT_INDEX** = 0
- const long **brathl::DEFAULT_DIM** [] = {1}
- CStringArray **brathl::CProduct::m_arrayLatitudeFieldName**
- CStringArray **brathl::CProduct::m_arrayLongitudeFieldName**
- static coda_array_ordering **brathl::CProduct::m_arrayOrdering** = coda_array_ordering_c
- uint32_t **brathl::CProduct::m_countForTrace**
- bool **brathl::CProduct::m_createVirtualField**
- COblntMap **brathl::CProduct::m_criterialInfoMap**
- COblntMap **brathl::CProduct::m_criteriaMap**
- int32_t **brathl::CProduct::m_currentRecord**
- coda_ProductFile * **brathl::CProduct::m_currFile**
- std::string **brathl::CProduct::m_currFileName**
- coda_Cursor **brathl::CProduct::m_cursor**
- CStringArray **brathl::CProduct::m_dataDictionaryFieldNames**
- CStringArray **brathl::CProduct::m_dataDictionaryFieldNamesWithDatasetName**
- CDataSet **brathl::CProduct::m_dataSet**
- std::string **brathl::CProduct::m_dataSetNameToRead**
- CDate **brathl::CProduct::m_dateProcessBegin**
- CDate **brathl::CProduct::m_dateProcessEnd**
- static const std::string **brathl::CCriteriaPassString::m_delimiter** = ","
- static const std::string **brathl::CCriteriaPassInt::m_delimiter** = " "
- double **brathl::CProduct::m_deltaTimeHighResolution**
- std::string **brathl::CProduct::m_description**
- bool **brathl::CProduct::m_disableTrace**
- bool **brathl::CProduct::m_expandArray**
- std::string **brathl::CProduct::CInfo::m_fieldName**
- CStringMap **brathl::CProduct::m_fieldNameEquivalence**
- bool **brathl::CProduct::m_fieldsHaveDefaultValue**
- CStringMap **brathl::CProduct::m_fieldSpecificUnit**
- CStringList **brathl::CProduct::m_fieldsToProcess**
- CStringArray **brathl::CProduct::m_fieldsToTranspose**
- CProductList **brathl::CProduct::m_fileList**
- double **brathl::CProduct::m_forceLatMaxCriteriaValue**
- double **brathl::CProduct::m_forceLatMinCriteriaValue**
- int32_t **brathl::CCriteriaPassInt::m_from**
- bool **brathl::CProduct::m_hasHighResolutionFieldToProcess**
- int32_t **brathl::CProduct::CInfo::m_index**
- int_t **brathl::CProduct::m_indexProcessedFile**
- int32_t **brathl::CProduct::CInfo::m_isUnion**
- std::string **brathl::CProduct::m_latitudeFieldName**
- CStringList **brathl::CProduct::m_listFieldExpandArray**
- CStringList **brathl::CProduct::m_listFieldOrigin**
- CField::CListField **brathl::CProduct::m_listFields**
- CListInfo **brathl::CProduct::m_listInfo**
- CStringList **brathl::CProduct::m_listInternalFieldName**
- CFile * **brathl::CProduct::m_logFile**
- std::string **brathl::CProduct::m_longitudeFieldName**
- int32_t **brathl::CProduct::m_nbRecords**
- uint32_t **brathl::CProduct::m_nSkippedRecord**
- uint32_t **brathl::CProduct::m_numHighResolutionMeasure**
- double **brathl::CProduct::m_offset**
- CStringArray **brathl::CCriteriaPassString::m_passes**
- double **brathl::CProduct::m_previousLatitude**

- double **bratl::CProduct::m_previousLongitude**
- double **bratl::CProduct::m_previousTimeStamp**
- std::string **bratl::CProductList::m_productClass**
- coda_format **bratl::CProductList::m_productFormat**
- std::string **bratl::CProductList::m_productType**
- size_t **bratl::CProduct::m_recordCount**
- **bratl_refDate** **bratl::CProduct::m_refDate**
- int32_t **bratl::CProduct::m_refPoint**
- int32_t **bratl::CCriteriaPassInt::m_to**
- uint32_t **bratl::CProduct::m_traceProcessRecordRatio**
- static const char * **bratl::CProduct::m_transposeFieldValuesFileName** = "bratl_transposefieldvalues.↵
txt"
- **CtreeField** **bratl::CProduct::m_tree**
- static const std::string **bratl::CProduct::m_treeRootName** = "Root"
- coda_Type * **bratl::CProduct::CInfo::m_type**
- coda_type_class **bratl::CProduct::CInfo::m_type_class**
- static const uint32_t **bratl::CProduct::MAX_INDEX** = 4
- std::string **bratl::CProductList::mCodaProductClass**
- std::string **bratl::CProductList::mCodaProductType**
- static const uint32_t **bratl::CProduct::MEAN_INDEX** = 1
- static const uint32_t **bratl::CProduct::MIN_INDEX** = 3
- std::string **bratl::CProduct::mLabel**
- static const int32_t **bratl::CProduct::NUMBER_OF_STATISTICS** = 5
- static const uint32_t **bratl::CProduct::STDDEV_INDEX** = 2

6.3.1 Detailed Description

6.3.2 Enumeration Type Documentation

6.3.2.1 enum **bratl::CCriteria::CriteriaKind**

Kind of criteria enumeration.

Enumerator

- UNKNOWN** not set
- LATLON** geographical latitude/longitude area
- DATETIME** date/time
- PASS** Pass
- CYCLE** Cycle

6.3.3 Function Documentation

6.3.3.1 **bratl::CCriteriaPassInt::CCriteriaPassInt** (int32_t *from*, int32_t *to*)

Constructor.

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

6.3.3.2 **bratl::CCriteriaPassInt::CCriteriaPassInt** (const std::string & *from*, const std::string & *to*)

Constructor.

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

6.3.3.3 bratl::CCriteriaPassInt::CCriteriaPassInt (const CStringArray & array)

Constructor from a array that contains start pass as std::string, end pass as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

6.3.3.4 bratl::CCriteriaPassString::CCriteriaPassString (const std::string & passes, const std::string & delimiter = CCriteriaPassString::m_delimiter)

Constructor from a std::string that contains passes delimited by a comma)

Parameters

<i>passes</i>	passes to set
---------------	---------------

References bratl::CCriteriaPassString::Set().

6.3.3.5 bratl::CCriteriaPassString::CCriteriaPassString (const CStringArray & array)

Constructor from a array that contains passes

Parameters

<i>array</i>	start and end dates
--------------	---------------------

References bratl::CCriteriaPassString::Set().

6.3.3.6 bratl::CProduct::CProduct (const std::string & fileName) [protected]

Creates new CProduct object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

6.3.3.7 bratl::CProduct::CProduct (const CStringList & fileNameList, bool check_only_first_file) [protected]

Creates new CProduct object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

6.3.3.8 bratl::CProductGeneric::CProductGeneric (const std::string & fileName) [inline]

Creates new CProdCProductGenericuct object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

6.3.3.9 bratl::CProductGeneric::CProductGeneric (const CStringList & fileNameList, bool check_only_first_file) [inline]

Creates new CProductGeneric object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

6.3.3.10 `brathl::CProductList::CProductList (const CProductList & o) [inline]`

Creates new **CProductList** (p. 259) object from another one

Parameters

<i>o</i>	[in] : productList object to be copied
----------	--

6.3.3.11 `brathl::CProductList::CProductList (const std::string & fileName)`

Creates new **CProductList** (p. 259) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

6.3.3.12 `brathl::CProductList::CProductList (const CStringList & fileNameList)`

Creates new CProduct object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

6.3.3.13 `brathl::CProductList::CProductList (const CStringArray & fileNameArray)`

Creates new CProduct object

Parameters

<i>fileNameArray</i>	[in] : array of file to be connected
----------------------	--------------------------------------

6.3.3.14 `bool brathl::CCriteriaPassString::Intersect (const std::string & passes, CStringArray & intersect)`

Creates the intersection of these passes with the given onee

Parameters

<i>passes</i>	intersect with this
<i>intersect</i>	intersection passes

Returns

true, or false if there is no intersection

6.3.3.15 `bool brathl::CCriteriaPassString::Intersect (CStringArray & passes, CStringArray & intersect)`

Creates the intersection of these passes with the given onee

Parameters

<i>passes</i>	intersect with this
<i>intersect</i>	intersection passes

Returns

true, or false if there is no intersection

References `brathl::CCriteriaPassString::m_passes`.

6.3.3.16 `bool brathl::CCriteriaPassInt::Intersect (CStringArray & array, CStringArray & intersect)`

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

Referenced by `brathl::CCriteriaPassInt::Intersect()`.

6.3.3.17 `bool brathl::CCriteriaPassInt::Intersect (CStringArray & array, CIntArray & intersect)`

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References `brathl::CCriteriaPassInt::Intersect()`.

6.3.3.18 `bool brathl::CCriteriaPassInt::Intersect (CIntArray & array, CStringArray & intersect)`

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References `brathl::CCriteriaPassInt::Intersect()`.

6.3.3.19 `bool brathl::CCriteriaPassInt::Intersect (CIntArray & array, CIntArray & intersect)`

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References `brathl::CCriteriaPassInt::Intersect()`.

6.3.3.20 `virtual bool brathl::CCriteriaPass::IsDefaultValue () [pure virtual]`

Tests whether date period have been initialized or not

Returns

true if not initialized

Implements **brathl::CCriteria** (p. 141).

Implemented in **brathl::CCriteriaPassInt** (p. 61), and **brathl::CCriteriaPassString** (p. 61).

6.3.3.21 `bool brathl::CCriteriaPassString::IsDefaultValue () [virtual]`

Tests whether passes have been initialized or not

Returns

true if not initialized

Implements **brathl::CCriteriaPass** (p. 60).

References `brathl::CCriteriaPassString::m_passes`.

6.3.3.22 `bool brathl::CCriteriaPassInt::IsDefaultValue () [virtual]`

Tests whether the pass have been initialized or not

Returns

true if not initialized

Implements **brathl::CCriteriaPass** (p. 60).

References `brathl::CCriteriaPassInt::m_from`, and `brathl::CCriteriaPassInt::m_to`.

6.3.3.23 `virtual bool brathl::CProduct::IsHighResolutionField (CField *) [inline], [virtual]`

Determines if a field object is a 'high resolution' array data see classes derived from CProduct.

6.3.3.24 `CProductList & brathl::CProductList::operator= (const CProductList & lst)`

Creates new **CProductList** (p. 259) object from another one

Parameters

<i>o</i>	[in] : productList object to be copied
----------	--

6.3.3.25 `void brathl::CCriteriaPassString::Set (const std::string & passes, const std::string & delimiter = CCriteriaPassString::m_delimiter)`

Sets one or more passes from a std::string (delimited by a comma)

Parameters

<i>passes</i>	passes to set
---------------	---------------

References `brathl::CCriteriaPassString::m_passes`.

Referenced by `brathl::CCriteriaPassString::CCriteriaPassString()`.

6.3.3.26 `void brathl::CCriteriaPassString::Set (const CStringArray & array)`

Sets passes from a array

Parameters

<i>array</i>	array of passes
--------------	-----------------

References `brathl::CCriteriaPassString::m_passes`.

6.3.3.27 `void brathl::CCriteriaPassInt::Set (int32_t from, int32_t to)`

Sets date period from start and end pass

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

References `brathl::CCriteriaPassInt::SetFrom()`, and `brathl::CCriteriaPassInt::SetTo()`.

6.3.3.28 `void brathl::CCriteriaPassInt::Set (const std::string & from, const std::string & to)`

Sets date period from start and end pass

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

References `brathl::CTools::StrToInt32()`.

6.3.3.29 `void brathl::CCriteriaPassInt::Set (const CStringArray & array)`

Sets a date period from a array that contains start pass as `std::string`, end pass as `std::string`

Parameters

<i>array</i>	start and end dates
--------------	---------------------

6.3.3.30 `virtual void brathl::CCriteriaPass::SetDefaultValue () [pure virtual]`

Sets internal value to the default value (uninitialized)

Implements **`brathl::CCriteria`** (p. 142).

Implemented in **`brathl::CCriteriaPassInt`** (p. 62), and **`brathl::CCriteriaPassString`** (p. 62).

6.3.3.31 `void brathl::CCriteriaPassString::SetDefaultValue () [virtual]`

Sets internal value to the default value (uninitialized)

Implements **`brathl::CCriteriaPass`** (p. 62).

References `brathl::CCriteriaPassString::m_passes`.

6.3.3.32 `void brathl::CCriteriaPassInt::SetDefaultValue () [virtual]`

Sets internal value to the default value (uninitialized)

Implements **`brathl::CCriteriaPass`** (p. 62).

References `brathl::CCriteriaPassInt::m_from`, and `brathl::CCriteriaPassInt::m_to`.

6.3.3.33 `void brathl::CCriteriaPassInt::SetFrom (int32_t from)`

Sets start pass

Parameters

<i>to</i>	start pass
-----------	------------

References bratl::CCriteriaPassInt::m_from.

Referenced by bratl::CCriteriaPassInt::Set().

6.3.3.34 void bratl::CCriteriaPassInt::SetFrom (const std::string & from)

Sets start pass

Parameters

<i>to</i>	start pass
-----------	------------

References bratl::CCriteriaPassInt::m_from, and bratl::CTools::StrToInt32().

6.3.3.35 void bratl::CCriteriaPassInt::SetTo (int32_t to)

Sets end pass

Parameters

<i>to</i>	end pass
-----------	----------

References bratl::CCriteriaPassInt::m_to.

Referenced by bratl::CCriteriaPassInt::Set().

6.3.3.36 void bratl::CCriteriaPassInt::SetTo (const std::string & to)

Sets end pass

Parameters

<i>to</i>	end pass
-----------	----------

References bratl::CCriteriaPassInt::m_to, and bratl::CTools::StrToInt32().

6.3.4 Variable Documentation

6.3.4.1 const long bratl::DEFAULT_DIM[] = {1}

Product management class.

Version

1.0

6.3.4.2 int32_t bratl::CCriteriaPassInt::m_from [protected]

start pass

Referenced by bratl::CCriteriaPassInt::Dump(), bratl::CCriteriaPassInt::IsDefaultValue(), bratl::CCriteriaPassInt::SetDefaultValue(), and bratl::CCriteriaPassInt::SetFrom().

6.3.4.3 int32_t bratl::CProduct::m_nbRecords [protected]

Number of records to read

6.3.4.4 CStringArray bratl::CCriteriaPassString::m_passes [protected]

Date period

Referenced by `brathl::CCriteriaPassString::Dump()`, `brathl::CCriteriaPassString::Intersect()`, `brathl::CCriteriaPassString::IsDefaultValue()`, `brathl::CCriteriaPassString::Set()`, and `brathl::CCriteriaPassString::SetDefaultValue()`.

6.3.4.5 `int32_t brathl::CCriteriaPassInt::m_to` `[protected]`

end pass

Referenced by `brathl::CCriteriaPassInt::Dump()`, `brathl::CCriteriaPassInt::IsDefaultValue()`, `brathl::CCriteriaPassInt::SetDefaultValue()`, and `brathl::CCriteriaPassInt::SetTo()`.

6.4 Date conversion classes

Classes

- class **brathl::CDate**
- class **brathl::CDatePeriod**

6.4.1 Detailed Description

6.5 File services

Classes

- class **bratl::CFile**

Enumerations

- enum **bratl::CFile::openFlags** {
bratl::CFile::modeRead = 0x0001, **bratl::CFile::modeWrite** = 0x0002, **bratl::CFile::modeAppend** = 0x0004, **bratl::CFile::modeReadWrite** = 0x0008,
bratl::CFile::modeRWCreate = 0x0010, **bratl::CFile::modeReadAppend** = 0x0020, **bratl::CFile::typeText** = 0x4000, **bratl::CFile::typeBinary** = static_cast<int32_t>(0x8000) }

6.5.1 Detailed Description

6.5.2 Enumeration Type Documentation

6.5.2.1 enum **bratl::CFile::openFlags**

File access mode enumeration: Flags can be combined by using the bitwise-OR (|) operator

Enumerator

modeRead Opens for reading. If the file does not exist or cannot be found, open fails.

modeWrite Opens an empty file for writing. If the given file exists, its contents are destroyed.

modeAppend Opens for writing at the end of the file (appending) without removing the EOF marker before writing new data to the file; creates the file first if it doesn't exist.

modeReadWrite Opens for both reading and writing. (The file must exist.)

modeRWCreate Opens an empty file for both reading and writing. If the given file exists, its contents are destroyed.

modeReadAppend Opens for reading and appending; the appending operation includes the removal of the EOF marker before new data is written to the file and the EOF marker is restored after writing is complete; creates the file first if it doesn't exist.

typeText Open in text (translated) mode.

typeBinary Open in binary (untranslated) mode.

6.6 Parameters

Classes

- class **brathl::CFileParams**
- class **brathl::CMapParameter**
- class **brathl::CParameter**

Functions

- **brathl::CMapParameter::CMapParameter ()**
CMapParameter (p. 236) ctor.
- virtual void **brathl::CMapParameter::Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- bool **brathl::CMapParameter::Erase** (CMapParameter::iterator iteratorParameter)
- bool **brathl::CMapParameter::Erase** (const std::string &key)
- **CParameter * brathl::CMapParameter::Exists** (const std::string &key)
- **CParameter * brathl::CMapParameter::Insert** (const std::string &key, const std::string &value)
- **CParameter * brathl::CMapParameter::operator[]** (const std::string key)
- void **brathl::CMapParameter::RemoveAll** ()
- virtual **brathl::CMapParameter::~~CMapParameter** ()
CMapParameter (p. 236) dtor.

6.6.1 Detailed Description

6.6.2 Function Documentation

6.6.2.1 bool brathl::CMapParameter::Erase (CMapParameter::iterator iteratorParameter)

Delete an element referenced by iteratorMnemo

Returns

true if no error, otherwise false

Referenced by brathl::CMapParameter::Erase().

6.6.2.2 bool brathl::CMapParameter::Erase (const std::string & key)

Delete an element by its key

Returns

true if no error, otherwise false

References brathl::CMapParameter::Erase().

6.6.2.3 CParameter * brathl::CMapParameter::Exists (const std::string & key)

Tests if an element identify by 'key' already exists

Returns

a **CParameter** (p. 242) pointer if exists, otherwise NULL

Referenced by brathl::CFileParams::CheckCount().

6.6.2.4 CParameter * brathl::CMapParameter::Insert (const std::string & key, const std::string & value)

Inserts a **CParameter** (p. 242) object

Parameters

<i>key</i>	: parameter name (std::map key)
<i>value</i>	: parameter value

Returns

CParameter (p. 242) object or NULL if error

References bratl::CParameter::AddValue().

6.6.2.5 CParameter * bratl::CMapParameter::operator[] (const std::string key)

operator[] redefinition. Searches a **CParameter** (p. 242) object identify by 'key'. DON'T USE this syntax if you are not sure the key exists, there's a bug in STL, after calling 'record = m_recordSetMap[recordSetName]', if key not existed and the std::map is empty then the key exists in the std::map and points to a NULL object **CParameter** (p. 242) *p = m_mapParam[key] -> use Exists method instead ;

Parameters

<i>key</i>	: parameter keyword
------------	---------------------

Returns

a pointer to th **CParameter** (p. 242) object if found, NULL if not found

6.6.2.6 void bratl::CMapParameter::RemoveAll ()

Remove all elements and clear the std::map

Referenced by bratl::CFileParams::Load(), and bratl::CMapParameter::~~CMapParameter().

6.7 Date conversion C APIs

Functions

- LIBRATHL_API int32_t **brathl_Cycle2YMDHMSM** (**brathl_mission** mission, int32_t cycle, int32_t pass, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_DayOfYear** (**brathl_DateYMDHMSM** *dateYMDHMSM, uint32_t *dayOfYear)
- LIBRATHL_API int32_t **brathl_DiffDSM** (**brathl_DateDSM** *dateDSM1, **brathl_DateDSM** *dateDSM2, double *diff)
- LIBRATHL_API int32_t **brathl_DiffJulian** (**brathl_DateJulian** *dateJulian1, **brathl_DateJulian** *dateJulian2, double *diff)
- LIBRATHL_API int32_t **brathl_DiffYMDHMSM** (**brathl_DateYMDHMSM** *dateYMDHMSM1, **brathl_DateYMDHMSM** *dateYMDHMSM2, double *diff)
- LIBRATHL_API int32_t **brathl_DSM2Julian** (**brathl_DateDSM** *dateDSM, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_DSM2Seconds** (**brathl_DateDSM** *dateDSM, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)
- LIBRATHL_API int32_t **brathl_DSM2YMDHMSM** (**brathl_DateDSM** *dateDSM, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_Julian2DSM** (**brathl_DateJulian** *dateJulian, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_Julian2Seconds** (**brathl_DateJulian** *dateJulian, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)
- LIBRATHL_API int32_t **brathl_Julian2YMDHMSM** (**brathl_DateJulian** *dateJulian, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_NowYMDHMSM** (**brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_Seconds2DSM** (**brathl_DateSecond** *dateSeconds, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_Seconds2Julian** (**brathl_DateSecond** *dateSeconds, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_Seconds2YMDHMSM** (**brathl_DateSecond** *dateSeconds, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Cycle** (**brathl_mission** mission, **brathl_DateYMDHMSM** *dateYMDHMSM, int32_t *cycle, int32_t *pass)
- LIBRATHL_API int32_t **brathl_YMDHMSM2DSM** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Julian** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Seconds** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)

6.7.1 Detailed Description

6.7.2 Function Documentation

6.7.2.1 LIBRATHL_API int32_t brathl_Cycle2YMDHMSM (brathl_mission mission, int32_t cycle, int32_t pass, brathl_DateYMDHMSM * dateYMDHMSM)

Converts a cyle/pass into a date

Parameters

in	mission	: mission type (see brathl_mission (p.317))
----	---------	--

in	<i>cycle</i>	: number of cycle to convert
in	<i>pass</i>	: number of pass in the cycle to convert
out	<i>dateYMDHMSM</i>	: date corresponding to the cycle/pass

Returns

#BRATHL_SUCCESS or error code (see Cycle_date_error_codes)

References brathl_errno, and brathl::CDate::Convert2YMDHMSM().

Referenced by FTN_NAME().

6.7.2.2 LIBRATHL_API int32_t brathl_DayOfYear (brathl_DateYMDHMSM * dateYMDHMSM, uint32_t * dayOfYear)

Retrieves the day of year of a date

Parameters

in	<i>dateYMDHMSM</i>	: date
out	<i>dayOfYear</i>	: day of year of the date parameter

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::DayOfYear(), and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.3 LIBRATHL_API int32_t brathl_DiffDSM (brathl_DateDSM * dateDSM1, brathl_DateDSM * dateDSM2, double * diff)

Computes the difference between two dates (date1 - date2) the result is expressed in a decimal number of seconds

Parameters

in	<i>dateDSM1</i>	: date1
in	<i>dateDSM2</i>	: date2
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.4 LIBRATHL_API int32_t brathl_DiffJulian (brathl_DateJulian * dateJulian1, brathl_DateJulian * dateJulian2, double * diff)

Computes the difference between two dates (date1 - date2) the result is expressed in a decimal number of seconds

Parameters

in	<i>dateJulian1</i>	: date1
in	<i>dateJulian2</i>	: date2
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.5 LIBRATHL_API int32_t brathl_DiffYMDHMSM (brathl_DateYMDHMSM * dateYMDHMSM1, brathl_DateYMDHMSM * dateYMDHMSM2, double * diff)

Computes the difference between two dates (date1 - date2) the result is expressed in a decimal number of seconds

Parameters

in	<i>dateYMDHMSM1</i>	: date1
in	<i>dateYMDHMSM2</i>	: date2
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.6 LIBRATHL_API int32_t brathl_DSM2Julian (brathl_DateDSM * dateDSM, brathl_refDate refDate, brathl_DateJulian * dateJulian)

Converts a days-seconds-microseconds date into a decimal julian date, according to refDate parameter

Parameters

in	<i>dateDSM</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateJulian</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DecimalJulian(), _structDateJulian::julian, _structDateJulian::refDate, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.7 LIBRATHL_API int32_t brathl_DSM2Seconds (brathl_DateDSM * dateDSM, brathl_refDate refDate, brathl_DateSecond * dateSeconds)

Converts a date in days-seconds-microseconds into a seconds, according to refDate parameter

Parameters

in	<i>dateDSM</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateSeconds</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2Second(), _structDateSecond::nbSeconds, _structDateSecond::refDate, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.8 LIBRATHL_API int32_t brathl_DSM2YMDHMSM (brathl_DateDSM * dateDSM, brathl_DateYMDHMSM * dateYMDHMSM)

Converts a days-seconds-microseconds date into a year, month, day, hour, minute, second, microsecond date

Parameters

in	<i>dateDSM</i>	: date to convert
out	<i>dateYMDHMSM</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2YMDHMSM(), and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.9 LIBRATHL_API int32_t brathl_Julian2DSM (brathl_DateJulian * dateJulian, brathl_refDate refDate, brathl_DateDSM * dateDSM)

Converts a decimal julian date into a days-seconds-microseconds date, according to refDate parameter

Parameters

in	<i>dateJulian</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateDSM</i>	: result of conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, _structDateDSM::refDate, _structDateDSM::seconds, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.10 LIBRATHL_API int32_t brathl_Julian2Seconds (brathl_DateJulian * dateJulian, brathl_refDate refDate, brathl_DateSecond * dateSeconds)

Converts a decimal julian date into seconds, according to refDate parameter

Parameters

in	<i>dateJulian</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateSeconds</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2Second(), _structDateSecond::nbSeconds, _structDateSecond::refDate, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.11 `LIBRATHL_API int32_t brathl_Julian2YMDHMSM (brathl_DateJulian * dateJulian, brathl_DateYMDHMSM * dateYMDHMSM)`

Converts a decimal julian date into a year, month, day, hour, minute, second, microsecond date

Parameters

in	<i>dateJulian</i>	: date to convert
out	<i>dateYMDHMSM</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2YMDHMSM(), and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.12 `LIBRATHL_API int32_t brathl_NowYMDHMSM (brathl_DateYMDHMSM * dateYMDHMSM)`

Gets the current date/time,

Parameters

out	<i>dateYMDHMSM</i>	: current date/time
-----	--------------------	---------------------

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2YMDHMSM(), and brathl::CDate::SetDateNow().

Referenced by FTN_NAME().

6.7.2.13 `LIBRATHL_API int32_t brathl_Seconds2DSM (brathl_DateSecond * dateSeconds, brathl_refDate refDate, brathl_DateDSM * dateDSM)`

Converts seconds into a days-seconds-microseconds date, according to refDate parameter

Parameters

in	<i>dateSeconds</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateDSM</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, ↔ _structDateDSM::refDate, _structDateDSM::seconds, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.14 `LIBRATHL_API int32_t brathl_Seconds2Julian (brathl_DateSecond * dateSeconds, brathl_refDate refDate, brathl_DateJulian * dateJulian)`

Converts seconds into a decimal julian date, according to refDate parameter

Parameters

in	<i>dateSeconds</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateJulian</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DecimalJulian(), _structDateJulian::julian, _structDateJulian::refDate, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.15 LIBRATHL_API int32_t brathl_Seconds2YMDHMSM (brathl_DateSecond * dateSeconds, brathl_DateYMDHMSM * dateYMDHMSM)

Converts seconds into a year, month, day, hour, minute, second, microsecond date

Parameters

in	<i>dateSeconds</i>	: date to convert
out	<i>dateYMDHMSM</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2YMDHMSM(), and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.16 LIBRATHL_API int32_t brathl_YMDHMSM2Cycle (brathl_mission mission, brathl_DateYMDHMSM * dateYMDHMSM, int32_t * cycle, int32_t * pass)

Converts a date into a cycle/pass

Parameters

in	<i>mission</i>	: mission type (see brathl_mission (p. 317))
in	<i>dateYMDHMSM</i>	: date to convert
out	<i>cycle</i>	: number of cycle
out	<i>pass</i>	: number of pass in the cycle

Returns

#BRATHL_SUCCESS or error code (see Cycle_date_error_codes)

References brathl_errno, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.17 LIBRATHL_API int32_t brathl_YMDHMSM2DSM (brathl_DateYMDHMSM * dateYMDHMSM, brathl_refDate refDate, brathl_DateDSM * dateDSM)

Converts a year, month, day, hour, minute, second, microsecond date into a days-seconds-microseconds date, according to refDate parameter

Parameters

in	<i>dateYMDHMSM</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateDSM</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, ↵
_structDateDSM::refDate, _structDateDSM::seconds, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.18 LIBRATHL_API int32_t brathl_YMDHMSM2Julian (brathl_DateYMDHMSM * *dateYMDHMSM*, brathl_refDate
refDate, brathl_DateJulian * *dateJulian*)

Converts a year, month, day, hour, minute, second, microsecond date into a decimal julian date, according to refDate parameter

Parameters

in	<i>dateYMDHMSM</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateJulian</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2DecimalJulian(), _structDateJulian::julian, _structDateJulian::ref↵
Date, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.7.2.19 LIBRATHL_API int32_t brathl_YMDHMSM2Seconds (brathl_DateYMDHMSM * *dateYMDHMSM*, brathl_refDate
refDate, brathl_DateSecond * *dateSeconds*)

Converts a year, month, day, hour, minute, second, microsecond date into seconds, according to refDate parameter

Parameters

in	<i>dateYMDHMSM</i>	: date to convert
in	<i>refDate</i>	: date reference conversion
out	<i>dateSeconds</i>	: result of the conversion

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_errno, brathl::CDate::Convert2Second(), _structDateSecond::nbSeconds, _structDateSecond↵
::refDate, and brathl::CDate::SetDate().

Referenced by FTN_NAME().

6.8 C API for reading data

Functions

- LIBRATHL_API void **brathl_LoadAliasesDictionary** ()
- LIBRATHL_API int32_t **brathl_ReadData** (int32_t nbFiles, char **fileNames, const char *recordName, const char *selection, int32_t nbData, char **dataExpressions, char **units, double **results, int32_t sizes[], size_t *actualSize, int ignoreOutOfRange, int statistics, double defaultValue)
- LIBRATHL_API void **brathl_RegisterAlgorithms** ()

6.8.1 Detailed Description

6.8.2 Function Documentation

6.8.2.1 LIBRATHL_API int32_t brathl_ReadData (int32_t nbFiles, char ** fileNames, const char * recordName, const char * selection, int32_t nbData, char ** dataExpressions, char ** units, double ** results, int32_t sizes[], size_t * actualSize, int ignoreOutOfRange, int statistics, double defaultValue)

Read data from a set of files Each measure for a data is a scalar value (a single number)

Parameters

in	<i>nbFiles</i>	: Number of files in file name list This is the usable size of #fileNames
in	<i>fileNames</i>	: File name list Must contain at least #nbFiles entries. If an entry is NULL or points to an empty string, the entry is ignored.
in	<i>selection</i>	: Expression involving data fields which has to be true to select returned data if NULL or empty string no selection is done (all data is selected)
in	<i>nbData</i>	: Number of expression used to retrieve data
in	<i>dataExpressions</i>	: Expression applied to data fields to build the wanted value Must contain at least #nbData entries. If an entry is NULL or points to an empty string, the data returned are always default values.
in	<i>units</i>	: Wanted unit for each expression Must be NULL or contain at least #nbData entries. If NULL, no unit conversion is done. If an entry is NULL or points to an empty string, no unit conversion is applied to the data of the corresponding expression. When a unit conversion has to be applied, the result of the expression is considered to be the base unit (SI). For example if the wanted unit is gram/l, the unit of the expression is supposed to be kilogram/m3 (internally all data are converted to base unit of the actual fields unit which is coherent with the above assumption).
	<i>results</i>	[in/out] : Data read Must be a vector of at least #nbData pointers (entries) to values to read. If NULL, nothing is returned in results and sizes MUST be NULL (otherwise this is an error). An entry can be NULL, see #sizes for the actual behaviour.
	<i>sizes</i>	[in/out] : Number of allocated values in a #results entry. Must be a vector of at least #nbData integers. If NULL, results MUST also be NULL (otherwise this is an error). If a value is 0, nothing is returned. If a value is > 0, the corresponding entry in results must not be NULL and must have been allocated to be able to store as much float values as indicated. If a value is < 0, and the corresponding entry in results is NULL, the entry will be allocated with enough space to store the result and sizes modified to reflect the size of allocated data (may be more than actual used ones). If a value is < 0, and the corresponding entry in results is not NULL, this is an error.

out	<i>actualSize</i>	: Number of actual data needed to store result. It cannot be NULL. The actual number of values in the corresponding entry of #results are returned in this number (all entries need the same amount of result). If #result is NULL, the number of values which would be needed for each entry is returned.
in	<i>ignoreOutOfRange</i>	: Skip excess data. 0=false, other = true If true, #actualSize can be greater than any positive value of #sizes, if there is too much value to store they are ignored. If false, it generates an error. Has no effect on #sizes entries which are <= 0 (or if it is NULL).
in	<i>statistics</i>	<p>: returns statistics on data instead of data themselves 0=false, other = true If statistics is true, ignoreOutOfRange must be false. And sizes must be <=0 or >=5. The returned values for each expression are:</p> <ul style="list-style-type: none"> • Count of <i>valid</i> data taken into account. Invalid data are those which are equal to the default/missing value • Mean of the valid data. • Standard deviation of the valid data • Minimum value of the valid data • Maximum value of the valid data <p>In this case actualSize always returns 5</p>
in	<i>defaultValue</i>	: value to use for default/missing values This is the value you want to indicate that a value is missing or invalid.

Returns

#BRATHL_SUCCESS or error code

References brathl_errno.

Referenced by FTN_NAME().

6.9 Date conversion Fortran APIs

Functions

- void **FTN_NAME** (brathlf_setrefuser1, BRATHLF_SETREFUSER1)
- void **FTN_NAME** (brathlf_setrefuser2, BRATHLF_SETREFUSER2)
- INTEGER4 **FTN_NAME** (brathlf_geterrno, BRATHLF_GETERRNO)
- void **FTN_NAME** (brathlf_errno2string, BRATHLF_ERRNO2STRING)
- INTEGER4 **FTN_NAME** (brathlf_seconds2dsm, BRATHLF_SECONDS2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2seconds, BRATHLF_DSM2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_julian2dsm, BRATHLF_JULIAN2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2julian, BRATHLF_DSM2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2dsm, BRATHLF_YMDHMSM2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2ymdhmsm, BRATHLF_DSM2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_seconds2julian, BRATHLF_SECONDS2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_julian2seconds, BRATHLF_JULIAN2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_seconds2ymdhmsm, BRATHLF_SECONDS2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2seconds, BRATHLF_YMDHMSM2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_julian2ymdhmsm, BRATHLF_JULIAN2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2julian, BRATHLF_YMDHMSM2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_nowymdhmsm, BRATHLF_NOWYMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_dayofyear, BRATHLF_DAYOFYEAR)
- INTEGER4 **FTN_NAME** (brathlf_diffymdhmsm, BRATHLF_DIFFYMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_diffdsm, BRATHLF_DIFFDSM)
- INTEGER4 **FTN_NAME** (brathlf_diffjulian, BRATHLF_DIFFJULIAN)
- INTEGER4 **FTN_NAME** (brathlf_cycle2ymdhmsm, BRATHLF_CYCLE2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2cycle, BRATHLF_YMDHMSM2CYCLE)

6.9.1 Detailed Description

6.9.2 Function Documentation

6.9.2.1 void FTN_NAME (brathlf_setrefuser1 , BRATHLF_SETREFUSER1)

Initializes the date reference user1 from a string See also **brathl_refDate** (p. 317)

Fortran specification

```
SUBROUTINE brathlf_SetRefUser1(dateRefUser)
  CHARACTER*(*) dateRefUser
```

Parameters

in	<i>dateRefUser</i>	: date string (format: YYYY-MM-DD HH:MM:SS:MS)
----	--------------------	--

References BRATHL_REF_DATE_USER_LEN, and brathl_refDateUser1.

6.9.2.2 void FTN_NAME (brathlf_setrefuser2 , BRATHLF_SETREFUSER2)

Initializes the date reference user2 from a string See also **brathl_refDate** (p. 317)

Fortran specification

```
SUBROUTINE brathlf_SetRefUser2(dateRefUser)
  CHARACTER*(*) dateRefUser
```


Parameters

<i>dateRefUser</i>	: date string (format: YYYY-MM-DD HH:MM:SS:MS)
--------------------	--

References BRATHL_REF_DATE_USER_LEN, and brathl_refDateUser2.

6.9.2.3 INTEGER4 FTN_NAME (brathlf_geterrno , BRATHLF_GETERRNO)

returns **brathl_errno** (p. 320)

Fortran specification

```
INTEGER*4 FUNCTION brathlf_GetErrno()
```

Returns

Last registered error code

References brathl_errno.

6.9.2.4 void FTN_NAME (brathlf_errno2string , BRATHLF_ERRNO2STRING)

Retrieve a string with the error description

With a few exceptions almost all BRATHL functions return an integer that indicate whether the function was able to perform its operations successfully. The return value will be 0 on success and < 0 otherwise. The result is also save in the global variable #brat_errno In case you get a negative value.

```
\par Fortran specification
SUBROUTINE brathlf_errno2string(err, str)<BR>
  INTEGER*4 err
  CHARACTER*(*) str
```

Parameters

in	<i>err</i>	: error code
out	<i>str</i>	: string error description

References brathl_Errno2String().

6.9.2.5 INTEGER4 FTN_NAME (brathlf_seconds2dsm , BRATHLF_SECONDS2DSM)

Converts seconds into a days-seconds-microseconds date, according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_Seconds2DSM(iRefDateSrc,iSeconds,iRefDateDest,oDays,oSeconds,oMu↵
Seconds)
INTEGER*4 iRefDateSrc REAL*8 iSeconds INTEGER*4 iRefDateDest INTEGER*4 oDays INTEGER*4 o↵
Seconds INTEGER*4 oMuSeconds
```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iSeconds</i>	: date to convert
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oDays</i>	: numbers of days
out	<i>oSeconds</i>	: number of seconds

out	<i>oMuSeconds</i>	: numbers of microseconds
-----	-------------------	---------------------------

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Seconds2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, _structDateSecond↵
::nbSeconds, _structDateSecond::refDate, and _structDateDSM::seconds.

6.9.2.6 INTEGER4 FTN_NAME (brathlf_dsm2seconds , BRATHLF_DSM2SECONDS)

Converts a date in days-seconds-microseconds into a seconds, according to refDate parameter

Fortran specification

```

INTEGER*4 FUNCTION brathlf_DSM2Seconds(iRefDateSrc,iDays,iSeconds,iMuSeconds,iRefDateDest,o↵
Seconds)
INTEGER*4 iRefDateSrc INTEGER*4 iDays INTEGER*4 iSeconds INTEGER*4 iMuSeconds INTEGER*4
iRefDateDest REAL*8 oSeconds

```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iDays</i>	: numbers of days
in	<i>iSeconds</i>	: number of seconds
in	<i>iMuSeconds</i>	: numbers of microseconds
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oSeconds</i>	: date to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DSM2Seconds(), _structDateDSM::days, _structDateDSM::muSeconds, _structDateSecond↵
::nbSeconds, _structDateDSM::refDate, and _structDateDSM::seconds.

6.9.2.7 INTEGER4 FTN_NAME (brathlf_julian2dsm , BRATHLF_JULIAN2DSM)

Converts a decimal julian date into a days-seconds-microseconds date, according to refDate parameter

Fortran specification

```

INTEGER*4 FUNCTION brathlf_Julian2DSM(iRefDateSrc,iJulian,iRefDateDest,oDays,oSeconds,oMu↵
Seconds)
INTEGER*4 iRefDateSrc REAL*8 iJulian INTEGER*4 iRefDateDest INTEGER*4 oDays INTEGER*4 o↵
Seconds INTEGER*4 oMuSeconds

```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iJulian</i>	: decimal julian date
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oDays</i>	: numbers of days
out	<i>oSeconds</i>	: number of seconds

out	<i>oMuSeconds</i>	: numbers of microseconds
-----	-------------------	---------------------------

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Julian2DSM(), _structDateDSM::days, _structDateJulian::julian, _structDateDSM::muSeconds, _structDateJulian::refDate, and _structDateDSM::seconds.

6.9.2.8 INTEGER4 FTN_NAME (brathlf_dsm2julian , BRATHLF_DSM2JULIAN)

Converts a days-seconds-microseconds date into a decimal julian date, according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_DSM2Julian(iRefDateSrc,iDays,iSeconds,iMuSeconds,iRefDateDest,oJulian)
INTEGER*4 iRefDateSrc INTEGER*4 iDays INTEGER*4 iSeconds INTEGER*4 iMuSeconds INTEGER*4
iRefDateDest REAL*8 oJulian
```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iDays</i>	: numbers of days
in	<i>iSeconds</i>	: number of seconds
in	<i>iMuSeconds</i>	: numbers of microseconds
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oJulian</i>	: date to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DSM2Julian(), _structDateDSM::days, _structDateJulian::julian, _structDateDSM::muSeconds, _structDateDSM::refDate, and _structDateDSM::seconds.

6.9.2.9 INTEGER4 FTN_NAME (brathlf_ymdhmsm2dsm , BRATHLF_YMDHMSM2DSM)

Converts a year, month, day, hour, minute, second, microsecond date into a days-seconds-microseconds date, according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_YMDHMSM2DSM(iYear,iMonth,iDay,iHour,iMinute,iSecond,iMuSecond,iRefDateDest,oDays,oSeconds,oMuSeconds)
```

```
INTEGER*4 iYear, INTEGER*4 iMonth, INTEGER*4 iDay, INTEGER*4 iHour, INTEGER*4 iMinute, INTEGER*4 iSecond, INTEGER*4 iMuSecond, INTEGER*4 iRefDateDest, INTEGER*4 oDays, INTEGER*4 oSeconds,
INTEGER*4 oMuSeconds
```

Parameters

in	<i>iYear</i>	: year
in	<i>iMonth</i>	: month
in	<i>iDay</i>	: day
in	<i>iHour</i>	: hour

in	<i>iMinute</i>	: minute
in	<i>iSecond</i>	: second
in	<i>iMuSecond</i>	: micro-second
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oDays</i>	: numbers of days
out	<i>oSeconds</i>	: number of seconds
out	<i>oMuSeconds</i>	: numbers of microseconds

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_YMDHMSM2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, and _structDateDSM::seconds.

6.9.2.10 INTEGER4 FTN_NAME (brathlf_dsm2ymdhmsm , BRATHLF_DSM2YMDHMSM)

Converts a days-seconds-microseconds date into a year, month, day, hour, minute, second, microsecond date according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_DSM2MDHMSM(iRefDateSrc,iDays,iSeconds,iMuSeconds,oYear,oMonth,oDay,oHour,oMinute,oSecond,oMuSecond)
```

INTEGER*4 iRefDateSrc, INTEGER*4 iDays, INTEGER*4 iSeconds, INTEGER*4 iMuSeconds INTEGER*4 oYear, INTEGER*4 oMonth, INTEGER*4 oDay, INTEGER*4 oHour, INTEGER*4 oMinute, INTEGER*4 oSecond, INTEGER*4 oMuSecond,

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iDays</i>	: numbers of days
in	<i>iSeconds</i>	: number of seconds
in	<i>iMuSeconds</i>	: numbers of microseconds
out	<i>oYear</i>	: year
out	<i>oMonth</i>	: month
out	<i>oDay</i>	: day
out	<i>oHour</i>	: hour convert
out	<i>oMinute</i>	: minute to convert
out	<i>oSecond</i>	: second to convert
out	<i>oMuSecond</i>	: micro-second to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DSM2YMDHMSM(), _structDateDSM::days, _structDateDSM::muSeconds, _structDateDSM::refDate, and _structDateDSM::seconds.

6.9.2.11 INTEGER4 FTN_NAME (brathlf_seconds2julian , BRATHLF_SECONDS2JULIAN)

Converts seconds into a decimal julian date, according to refDate parameter INTEGER*4 FUNCTION brathlf_Seconds2Julian(iRefDateSrc,iSeconds,iRefDateDest,oJulian)

Fortran specification

```
INTEGER*4 iRefDateSrc, INTEGER*4 iSeconds, INTEGER*4 iRefDateDest REAL*8 oJulian,
```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iSeconds</i>	: number of seconds
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oJulian</i>	: julian date

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Seconds2Julian(), _structDateJulian::julian, _structDateSecond::nbSeconds, and _structDateSecond::refDate.

6.9.2.12 INTEGER4 FTN_NAME (brathlf_julian2seconds , BRATHLF_JULIAN2SECONDS)

Converts a decimal julian date into seconds, according to refDate parameter
 INTEGER*4 FUNCTION brathlf_Seconds2Julian(iRefDateSrc,iJulian,iRefDateDest,oSeconds)

Fortran specification

INTEGER*4 iRefDateSrc, REAL*8 iJulian, INTEGER*4 iRefDateDest INTEGER*4 oSeconds,

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iJulian</i>	: julian date
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oSeconds</i>	: number of seconds

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Julian2Seconds(), _structDateJulian::julian, _structDateSecond::nbSeconds, and _structDateJulian::refDate.

6.9.2.13 INTEGER4 FTN_NAME (brathlf_seconds2ymdhmsm , BRATHLF_SECONDS2YMDHMSM)

Converts seconds into a year, month, day, hour, minute, second, microsecond date, according to refDate parameter
 INTEGER*4 FUNCTION brathlf_Seconds2YMDHMSM(iRefDateSrc,iSeconds,oYear,oMonth,oDay,oHour,oMinute,oSecond,oMuSecond)

Fortran specification

INTEGER*4 iRefDateSrc, INTEGER*4 iSeconds, INTEGER*4 iRefDateDest INTEGER*4 oYear, INTEGER*4 oMonth, INTEGER*4 oDay, INTEGER*4 oHour, INTEGER*4 oMinute, INTEGER*4 oSecond, INTEGER*4 oMuSecond,

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iSeconds</i>	: number of seconds

out	<i>oYear</i>	: year
out	<i>oMonth</i>	: month
out	<i>oDay</i>	: day
out	<i>oHour</i>	: hour convert
out	<i>oMinute</i>	: minute to convert
out	<i>oSecond</i>	: second to convert
out	<i>oMuSecond</i>	: micro-second to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Seconds2YMDHMSM(), _structDateSecond::nbSeconds, and _structDateSecond::refDate.

6.9.2.14 INTEGER4 FTN_NAME (brathlf_ymdhmsm2seconds , BRATHLF_YMDHMSM2SECONDS)

Converts a year, month, day, hour, minute, second, microsecond date into seconds, according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_YMDHMSM2DSM(iYear,iMonth,iDay,iHour,iMinute,iSecond,iMuSecond,iRefDateDest,oSeconds)
```

INTEGER*4 iYear, INTEGER*4 iMonth, INTEGER*4 iDay, INTEGER*4 iHour, INTEGER*4 iMinute, INTEGER*4 iSecond, INTEGER*4 iMuSecond, INTEGER*4 iRefDateDest, INTEGER*4 oSeconds,

Parameters

in	<i>iYear</i>	: year
in	<i>iMonth</i>	: month
in	<i>iDay</i>	: day
in	<i>iHour</i>	: hour
in	<i>iMinute</i>	: minute
in	<i>iSecond</i>	: second
in	<i>iMuSecond</i>	: micro-second
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oSeconds</i>	: number of seconds

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_YMDHMSM2Seconds(), and _structDateSecond::nbSeconds.

6.9.2.15 INTEGER4 FTN_NAME (brathlf_julian2ymdhmsm , BRATHLF_JULIAN2YMDHMSM)

Converts julian date into into a year, month, day, hour, minute, second, microsecond date, according to refDate parameter
 INTEGER*4 FUNCTION brathl_Julian2YMDHMSM(iRefDateSrc,iJulian,oYear,oMonth,oDay,oHour,oMinute,oSecond,oMuSecond)

Fortran specification

```
INTEGER*4 iRefDateSrc, REAL*8 iJulian, INTEGER*4 oYear, INTEGER*4 oMonth, INTEGER*4 oDay, INTEGER*4 oHour, INTEGER*4 oMinute, INTEGER*4 oSecond, INTEGER*4 oMuSecond,
```

Parameters

in	<i>iRefDateSrc</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iJulian</i>	: julian date
out	<i>oYear</i>	: year
out	<i>oMonth</i>	: month
out	<i>oDay</i>	: day
out	<i>oHour</i>	: hour convert
out	<i>oMinute</i>	: minute to convert
out	<i>oSecond</i>	: second to convert
out	<i>oMuSecond</i>	: micro-second to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Julian2YMDHMSM(), _structDateJulian::julian, and _structDateJulian::refDate.

6.9.2.16 INTEGER4 FTN_NAME (brathlf_ymdhmsm2julian , BRATHLF_YMDHMSM2JULIAN)

Converts a year, month, day, hour, minute, second, microsecond date into a decimal julian date, according to refDate parameter

Fortran specification

```
INTEGER*4 FUNCTION brathlf_YMDHMSM2Julian(iYear,iMonth,iDay,iHour,iMinute,iSecond,iMuSecond,i←
RefDateDest,oJulian)
```

INTEGER*4 iYear, INTEGER*4 iMonth, INTEGER*4 iDay, INTEGER*4 iHour, INTEGER*4 iMinute, INTEGER*4 iSecond, INTEGER*4 iMuSecond, INTEGER*4 iRefDateDest, REAL*8 oJulian,

Parameters

in	<i>iYear</i>	: year
in	<i>iMonth</i>	: month
in	<i>iDay</i>	: day
in	<i>iHour</i>	: hour
in	<i>iMinute</i>	: minute
in	<i>iSecond</i>	: second
in	<i>iMuSecond</i>	: micro-second
in	<i>iRefDateDest</i>	: date reference conversion (see brathl_refDate (p. 317))
out	<i>oJulian</i>	: julian date

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_YMDHMSM2Julian(), and _structDateJulian::julian.

6.9.2.17 INTEGER4 FTN_NAME (brathlf_nowymdhmsm , BRATHLF_NOWYMDHMSM)

Gets the current date/time,

Fortran specification

```
INTEGER*4 FUNCTION brathlf_NowYMDHMSM(oYear,oMonth,oDay,oHour,oMinute,oSecond,oMuSecond)
```

INTEGER*4 oYear, INTEGER*4 oMonth, INTEGER*4 oDay, INTEGER*4 oHour, INTEGER*4 oMinute, INTEG←
ER*4 oSecond, INTEGER*4 oMuSecond,

Parameters

out	<i>oYear</i>	: year
out	<i>oMonth</i>	: month
out	<i>oDay</i>	: day
out	<i>oHour</i>	: hour convert
out	<i>oMinute</i>	: minute to convert
out	<i>oSecond</i>	: second to convert
out	<i>oMuSecond</i>	: micro-second to convert

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_NowYMDHMSM().

6.9.2.18 INTEGER4 FTN_NAME (brathlf_dayofyear , BRATHLF_DAYOFYEAR)

Retrieves the day of year of a date

Fortran specification

```
INTEGER*4 FUNCTION brathlf_DayOfYear(iYear,iMonth,iDay,iHour,iMinute,iSecond,iMuSecond,iRefDate←
Dest,oSeconds)
```

INTEGER*4 iYear, INTEGER*4 iMonth, INTEGER*4 iDay, INTEGER*4 iHour, INTEGER*4 iMinute, INTEGER*4 iSecond, INTEGER*4 iMuSecond, INTEGER*4 oQuant,

Parameters

in	<i>iYear</i>	: year
in	<i>iMonth</i>	: month
in	<i>iDay</i>	: day
in	<i>iHour</i>	: hour
in	<i>iMinute</i>	: minute
in	<i>iSecond</i>	: second
in	<i>iMuSecond</i>	: micro-second
out	<i>oQuant</i>	: day of year

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DayOfYear().

6.9.2.19 INTEGER4 FTN_NAME (brathlf_diffymdhmsm , BRATHLF_DIFFYMDHMSM)

Computes the difference between two dates (date1 - date2). The result is expressed in a decimal number of seconds

Fortran specification

```
INTEGER*4 FUNCTION brathl_DiffYMDHMSM(iYear1,iMonth1,iDay1,iHour1,iMinute1,iSecond1,iMu←
Second1,iYear2,iMonth2,iDay2,iHour2,iMinute2,iSecond2,iMuSecond2)
```

INTEGER*4 iYear1, INTEGER*4 iMonth1, INTEGER*4 iDay1, INTEGER*4 iHour1, INTEGER*4 iMinute1, IN←
TEGER*4 iSecond1, INTEGER*4 iMuSecond1, INTEGER*4 iYear2, INTEGER*4 iMonth2, INTEGER*4 iDay2,
INTEGER*4 iHour2, INTEGER*4 iMinute2, INTEGER*4 iSecond2, INTEGER*4 iMuSecond2, REAL*8 diff

Parameters

in	<i>iYear1</i>	: year
in	<i>iMonth1</i>	: month
in	<i>iDay1</i>	: day
in	<i>iHour1</i>	: hour
in	<i>iMinute1</i>	: minute
in	<i>iSecond1</i>	: second
in	<i>iMuSecond1</i>	: micro-second
in	<i>iYear2</i>	: year
in	<i>iMonth2</i>	: month
in	<i>iDay2</i>	: day
in	<i>iHour2</i>	: hour
in	<i>iMinute2</i>	: minute
in	<i>iSecond2</i>	: second
in	<i>iMuSecond2</i>	: micro-second
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DiffYMDHMSM().

6.9.2.20 INTEGER4 FTN_NAME (brathlf_diffdsm , BRATHLF_DIFFDSM)

Computes the difference between two dates (date1 - date2). The result is expressed in a decimal number of seconds

Fortran specification

```
INTEGER*4 FUNCTION brathl_DiffDSM(iRefDate1,iDays1,iSeconds1,iMuSeconds1,iRefDate2,iDays2,i←
Seconds2,iMuSeconds2,diff)
```

INTEGER*4 iRefDate1, INTEGER*4 iDays1, INTEGER*4 iSeconds1, INTEGER*4 iMuSeconds1, INTEGER*4 iRefDate2, INTEGER*4 iDays2, INTEGER*4 iSeconds2, INTEGER*4 iMuSeconds2 REAL*8 diff

Parameters

in	<i>iRefDate1</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iDays1</i>	: numbers of days
in	<i>iSeconds1</i>	: number of seconds
in	<i>iMuSeconds1</i>	: numbers of microseconds
in	<i>iRefDate2</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iDays2</i>	: numbers of days
in	<i>iSeconds2</i>	: number of seconds
in	<i>iMuSeconds2</i>	: numbers of microseconds
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DiffDSM(), _structDateDSM::days, _structDateDSM::muSeconds, _structDateDSM::refDate, and _structDateDSM::seconds.

6.9.2.21 INTEGER4 FTN_NAME (brathlf_diffjulian , BRATHLF_DIFFJULIAN)

Computes the difference between two dates (date1 - date2). The result is expressed in a decimal number of seconds

Fortran specification

```
INTEGER*4 FUNCTION brathl_DiffJulian(iRefDate1,iJulian1,iRefDate2,iJulian2,diff)
```

INTEGER*4 iRefDate1, REAL*8 iJulian1, INTEGER*4 iRefDate2, REAL*8 iJulian2, REAL*8 diff

Parameters

in	<i>iRefDate1</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iJulian1</i>	: julian date
in	<i>iRefDate2</i>	: source date reference (see brathl_refDate (p. 317))
in	<i>iJulian2</i>	: julian date
out	<i>diff</i>	: difference in seconds (date1 - date2)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_DiffJulian(), _structDateJulian::julian, and _structDateJulian::refDate.

6.9.2.22 INTEGER4 FTN_NAME (brathlf_cycle2ymdhmsm , BRATHLF_CYCLE2YMDHMSM)

Converts a cycle/pass into a date

Fortran specification

```
INTEGER*4 FUNCTION brathl_Cycle2YMDHMSM(iMission,iCycle,iPass,oYear,oMonth,oDay,oHour,oMinute,oSecond,oMuSecond)
```

INTEGER*4 iMission, INTEGER*4 iCycle, INTEGER*4 iPass, INTEGER*4 oYear, INTEGER*4 oMonth, INTEGER*4 oDay, INTEGER*4 oHour, INTEGER*4 oMinute, INTEGER*4 oSecond, INTEGER*4 oMuSecond,

Parameters

in	<i>iMission</i>	: mission type (see brathl_mission (p. 317))
in	<i>iCycle</i>	: number of cycle to convert
in	<i>iPass</i>	: number of pass in the cycle to convert
out	<i>oYear</i>	: year
out	<i>oMonth</i>	: month
out	<i>oDay</i>	: day
out	<i>oHour</i>	: hour
out	<i>oMinute</i>	: minute
out	<i>oSecond</i>	: second
out	<i>oMuSecond</i>	: micro-second

Returns

#BRATHL_SUCCESS or error code (see Cycle_date_error_codes)

References brathl_Cycle2YMDHMSM().

6.9.2.23 INTEGER4 FTN_NAME (brathlf_ymdhmsm2cycle , BRATHLF_YMDHMSM2CYCLE)

Converts a date into a cycle/pass

Fortran specification

```
INTEGER*4 FUNCTION brathl_YMDHMSM2Cycle(iMission,iYear,iMonth,iDay,iHour,iMinute,iSecond,iMuSecond,oCycle,oPass)
```

INTEGER*4 iMission, INTEGER*4 iYear, INTEGER*4 iMonth, INTEGER*4 iDay, INTEGER*4 iHour, INTEGER*4 iMinute, INTEGER*4 iSecond, INTEGER*4 iMuSecond, INTEGER*4 oCycle, INTEGER*4 oPass,

Parameters

in	<i>iMission</i>	: mission type (see brathl_mission (p.317))
in	<i>iYear</i>	: year
in	<i>iMonth</i>	: month
in	<i>iDay</i>	: day
in	<i>iHour</i>	: hour
in	<i>iMinute</i>	: minute
in	<i>iSecond</i>	: second
in	<i>iMuSecond</i>	: micro-second
out	<i>oCycle</i>	: number of cycle to convert
out	<i>oPass</i>	: number of pass in the cycle to convert

Returns

#BRATHL_SUCCESS or error code (see Cycle_date_error_codes)

References brathl_YMDHMSM2Cycle().

6.10 Fortran API for reading data

Functions

- INTEGER4 **FTN_NAME** (brathlf_readdata, BRATHLF_READDATA)

6.10.1 Detailed Description

6.10.2 Function Documentation

6.10.2.1 INTEGER4 FTN_NAME (brathlf_readdata , BRATHLF_READDATA)

Read data from a set of files Each measure for a data is a scalar value (a single number)

Fortran specification

```

INTEGER*4 FUNCTION brathlf_ReadData(nbFiles,
$ fileNames,
$ recordName,
$ selection,
$ nbData,
$ dataExpressions,
$ units,
$ results,
$ size,
$ actualSize,
$ ignoreOutOfRange,
$ statistics,
$ defaultValue,
INTEGER*4 nbFiles
CHARACTER*(*) fileNames(nbFiles) CHARACTER*(*) recordName CHARACTER*(*) selection INT←
EGER*4 nbData CHARACTER*(*) dataExpressions(nbData) CHARACTER*(*) units(nbData) REAL*8
results(size,nbData) INTEGER*4 size INTEGER*4 actualSize INTEGER*4 ignoreOutOfRange INTEGE←
R*4 statistics REAL*8 defaultValue

```

Parameters

in	<i>nbFiles</i>	: Number of files in file name list This is the usable size of #fileNames
in	<i>fileNames</i>	: File name list Must contain at least #nbFiles entries. If an entry is an empty string, the entry is ignored.
in	<i>selection</i>	: Expression involving data fields which has to be true to select returned data if it is an empty string no selection is done (all data is selected)
in	<i>nbData</i>	: Number of expressions used to retrieve data
in	<i>dataExpressions</i>	: Expression applied to data fields to build the wanted value Must contain at least #nbData entries. If an entry is an empty string, the data returned are always default values.
in	<i>units</i>	: Wanted unit for each expression Must contain at least #nbData entries. If an entry is an empty string, no unit conversion is applied to the data of the corresponding expression. When a unit conversion has to be applied, the result of the expression is considered to be the base unit (SI). For example if the wanted unit is gram/l, the unit of the expression is supposed to be kilogram/m3 (internally all data are converted to base unit of the actual fields unit which is coherent with the above assumption).

out	<i>results</i>	: Data read Must be a matrix of at least #nbData entries of size values. Must be declared as <code>real*8 results[size, N]</code> where $N \geq \text{\#nbData}$
in	<i>size</i>	: Number of data in each #results entry. Must be ≥ 0 . If 0, nothing is returned in results and results can be anything.
out	<i>actualSize</i>	: Number of actual data needed to store result. The actual number of values in the corresponding entry of #results are returned in this number. If size is 0 this is the number of values that would have been returned.
in	<i>ignoreOutOfRange</i>	: Skip excess data. 0=false, other = true If true, #actualSize can be greater than any positive value of #size, if there is too much value to store they are ignored. If false, it generates an error.
in	<i>statistics</i>	<p>: returns statistics on data instead of data themselves 0=false, other = true If statistics is true, ignoreOutOfRange must be false. And sizes must be ≤ 0 or ≥ 5. The returned values for each expression are:</p> <ul style="list-style-type: none"> • Count of <i>valid</i> data taken into account. Invalid data are those which are equal to the default/missing value • Mean of the valid data. • Standard deviation of the valid data • Minimum value of the valid data • Maximum value of the valid data <p>In this case actualSize always returns 5</p>
in	<i>defaultValue</i>	: value to use for default/missing values This is the value you want to indicate that a value is missing or invalid.

Returns

#BRATHL_SUCCESS or error code

References `brathl_ReadData()`.

7 Namespace Documentation

7.1 brathl Namespace Reference

Classes

- class **CArrayDoubleArray**
- class **CArrayDoublePtrArray**
- class **CBratAlgoFilterGaussian1D**
- class **CBratAlgoFilterGaussian2D**
- class **CBratAlgoFilterLanczos1D**
- class **CBratAlgoFilterLanczos2D**
- class **CBratAlgoFilterLoess1D**
- class **CBratAlgoFilterLoess2D**
- class **CBratAlgoFilterMedian1D**
- class **CBratAlgoFilterMedian2D**
- class **CBratAlgorithmBase**
- class **CBratAlgorithmGeosVel**
- class **CBratAlgorithmGeosVelAtp**
- class **CBratAlgorithmGeosVelGrid**
- class **CBratAlgorithmGeosVelGridU**
- class **CBratAlgorithmGeosVelGridV**
- class **CCriteria**
- class **CCriteriaCycle**
- class **CCriteriaCycleInfo**
- class **CCriteriaDatetime**
- class **CCriteriaDatetimestrInfo**
- class **CCriterialInfo**
- class **CCriteriaLatLon**
- class **CCriteriaLatLonInfo**
- class **CCriteriaPass**
- class **CCriteriaPassInfo**
- class **CCriteriaPassInt**
- class **CCriteriaPassIntInfo**
- class **CCriteriaPassString**
- class **CCriteriaPassStringInfo**
- class **CDataSet**
- class **CDate**
- class **CDatePeriod**
- class **CDoubleMap**
- class **CDoublePtrArray**
- class **CDoublePtrDoubleMap**
- class **CExpressionValue**
- class **CExternalFilesAvisoGrid**
- class **CExternalFilesJason2**
- class **CExternalFilesNetCDF**
- class **CField**
- class **CFieldArray**
- class **CFieldBasic**
- class **CFieldIndexData**
- class **CFieldNetCdf**
- class **CFieldNetCdfCF**
- class **CFieldNetCdfCFAttr**
- class **CFieldRecord**

- class **CFieldSet**
- class **CFieldSetArrayDbI**
- class **CFieldSetDbI**
- class **CFieldSetString**
- class **CFile**
- class **CFileParams**
- class **CInternalFiles**
- class **CInternalFilesYFX**
- class **CInternalFilesZFX**
- class **CIntList**
- class **CIntMap**
- class **CMapParameter**
- class **CMapProduct**
- class **CObArray**
- class **CObDoubleMap**
- class **CObIntMap**
- class **CObList**
- class **CObMap**
- class **CObStack**
- class **CParameter**
- class **CProductAop**
- class **CProductCryosat**
- class **CProductEnvisat**
- class **CProductEnvisatNetCdf**
- class **CProductErs**
- class **CProductErsWAP**
- class **CProductGeosatGDR**
- class **CProductGfo**
- class **CProductJason**
- class **CProductJason1NetCdf**
- class **CProductJason2**
- class **CProductList**
- class **CProductNetCdf**
- class **CProductNetCdfCF**
- class **CProductPodaac**
- class **CProductRads**
- class **CProductReaper**
- class **CProductRiverLake**
- class **CProductTopex**
- class **CProductTopexSDR**
- class **CPtrMap**
- class **CRecord**
- class **CRecordSet**
- class **CRegisteredPass**
- class **CStringList**
- class **CStringMap**
- class **CTools**
- class **CTreeField**
- class **CUIntMap**

Typedefs

- typedef std::bitset< 32 > **bitSet32**
- typedef double * **DoublePtr**
- typedef std::vector< DoublePtr > **doubleptrarray**
- typedef double **ExpressionCallableFunction1** (double)
- typedef double **ExpressionCallableFunction2** (double, double)
- typedef double **ExpressionCallableFunction3** (double, double, double)
- typedef double **ExpressionCallableFunctionAlgoN** (const char *, CVectorBratAlgorithmParam &arg)
- typedef double **ExpressionCallableFunctionBratAlgoBaseN** (CBratAlgorithmBase *algo, CVectorBratAlgorithmParam &arg)
- typedef double **ExpressionCallableFunctionStrToFlt1** (const char *)
- typedef const char * **ExpressionCallableFunctionStrToStr1** (const char *)
- typedef CUIntArray **ExpressionValueDimensions**
- typedef CDoubleArray **ExpressionValueValues**
- typedef std::list< int32_t > **intlist**
- typedef std::map< std::string, CParameter * > **map_parameter**
- typedef std::map< std::string, CStringArray > **maparraystring**
- typedef std::map< std::string, CBratAlgorithmBase * > **mapbratalgorithmbase**
- typedef std::map< std::string, double > **mapdouble**
- typedef std::map< double, DoublePtr * > **mapdoubledoubleptr**
- typedef std::map< double, CBratObject * > **mapdoubleobject**
- typedef std::map< std::string, int32_t > **mapint**
- typedef std::map< int32_t, CBratObject * > **mapintobject**
- typedef std::map< std::string, CBratObject * > **mapobject**
- typedef std::map< std::string, void * > **mapptr**
- typedef std::map< std::string, std::string > **mapstring**
- typedef std::map< std::string, CObjectTreeNode * > **mapTreeNode**
- typedef std::map< std::string, uint32_t > **mapuint**
- typedef std::numeric_limits< char > **numeric_limits_char**
- typedef std::vector< CBratObject * > **obarray**
- typedef std::list< CBratObject * > **oblist**
- typedef std::stack< CBratObject * > **obstack**
- typedef std::vector< std::string > **stringarray**
- typedef std::list< std::string > **stringlist**
- typedef std::vector< CBratAlgorithmBase * > **vectorbratalgorithmbase**

Enumerations

- enum **brathl_global_constants** {
 EARTH_ROTATION = 0, **LIGHT_SPEED**, **EARTH_GRAVITY**, **EARTH_RADIUS**,
 ELLIPSOID_PARAM }
- enum **ExpressionValueType** { **CharacterType**, **FloatType** }
- enum **FunctionCategory** {
 MathTrigo, **Statistical**, **Logical**, **Relational**,
 Constant, **BitwiseOp**, **DateTime**, **Algorithm**,
 Geographical }
- enum **NetCDFVarKind** {
 Unknown, **X**, **Y**, **Z**,
 T, **Latitude**, **Longitude**, **Data** }

Functions

- template<class T >
 CBratAlgorithmBase * **base_factory** ()
- CExternalFiles * **BuildExistingExternalFileKind** (const std::string &path)
- CInternalFiles * **BuildExistingInternalFileKind** (const std::string &name, const CStringArray *fieldNames)
- static void **CommentHnd** (void *userData, const char *data)
- static void **DefaultHnd** (void *userData, const char *s, int len)
- static void **EndElementHnd** (void *userData, const char *name)
- static double **GetGlobalConstant** (brathl_global_constants constantValue)
- static void **StartCdataHnd** (void *userData)
- static void **StartElementHnd** (void *userData, const char *name, const char **atts)
- static void **TextHnd** (void *userData, const char *s, int len)

Variables

- static const DefCharFunction1 **CharFonctions1** []
- static const DefConstant **Constants** []
- const std::string **CRYOSAT_MPH** = "mph"
- const std::string **CRYOSAT_SPH** = "sph"
- const long **DEFAULT_DIM** [] = {1}
- const char * **DUMP_FORMAT_DOUBLE** = "%.15g"
- const std::string **ENVISAT_MPH** = "mph"
- const std::string **ENVISAT_SPH** = "sph"
- const std::string **ERS_HEADER** = "header"
- static const DefFunction1 **Fonctions1** []
- static const DefFunction2 **Fonctions2** []
- static const DefFunction3 **Fonctions3** []
- static const DefFunctionAlgoN **FonctionsAlgoN** []
- static const
 DefFunctionBratAlgoBaseN **FonctionsBratAlgoBaseN** []
- const std::string **FORMAT_FLOAT_LATLON** = "%-#.5g"
- const std::string **FORMAT_INT_CYCLE** = "%d"
- const std::string **FORMAT_INT_PASS** = "%d"
- const std::string **GDR** = "GDR"
- const std::string **GDR_TITLE** = "standard dataset"
- const std::string **GENERIC_NETCDF_TYPE** = "Generic NetCdf"
- const std::string **GFO_HEADER** = "header"
- const std::string **JASON_HEADER** = "header"
- const int32_t **MAX_NUM_DIMS** = CODA_MAX_NUM_DIMS
- const std::string **NC_BYTE_NAME** = "signed 1 byte integer"

- const std::string **NC_CHAR_NAME** = "ASCII character"
- const std::string **NC_DOUBLE_NAME** = "double precision floating point number"
- const std::string **NC_FLOAT_NAME** = "single precision floating point number"
- const std::string **NC_INT64_NAME** = "signed 8 byte integer"
- const std::string **NC_INT_NAME** = "signed 4 byte integer"
- const std::string **NC_NAT_NAME** = "Not A Type"
- const std::string **NC_SHORT_NAME** = "signed 2 byte integer"
- const std::string **NC_STRING_NAME** = "array of strings"
- const std::string **NC_UBYTE_NAME** = "unsigned 1 byte integer"
- const std::string **NC_UINT64_NAME** = "unsigned 8 byte integer"
- const std::string **NC_UINT_NAME** = "unsigned 4 byte integer"
- const std::string **NC_USHORT_NAME** = "unsigned 2 byte integer"
- static const double **NcFillByte** = NC_FILL_BYTE
- static const double **NcFillChar** = NC_FILL_CHAR
- static const double **NcFillDouble** = NC_FILL_DOUBLE
- static const double **NcFillFloat** = NC_FILL_FLOAT
- static const double **NcFillInt** = NC_FILL_INT
- static const double **NcFillInt64** = (double)NC_FILL_INT64
- static const double **NcFillShort** = NC_FILL_SHORT
- static const double **NcFillString** = (double)(ptrdiff_t)NC_FILL_STRING
- static const double **NcFillUByte** = NC_FILL_UBYTE
- static const double **NcFillUInt** = NC_FILL_UINT
- static const double **NcFillUInt64** = (double)NC_FILL_UINT64
- static const double **NcFillUShort** = NC_FILL_USHORT
- const std::string **NETCDF_CF_PRODUCT_CLASS** = "NETCDF_CF"
- const std::string **NETCDF_PRODUCT_CLASS** = "NETCDF"
- const std::string **PODAAC_HEADER** = "header"
- const std::string **SGDR** = "SGDR"
- const std::string **SGDR_TITLE** = "expertise dataset"
- const std::string **SSHA** = "SSHA"
- const std::string **SSHA_TITLE** = "reduced dataset"
- const std::string **UNKNOWN_PRODUCT_CLASS** = "UNKNOWN"
- const std::string **YFX_NETCDF_TYPE** = "Y=F(X)"
- const std::string **ZFGY_NETCDF_TYPE** = "Z=F(X,Y)"

7.1.1 Detailed Description

object base class

Version

1.0

7.1.2 Typedef Documentation

7.1.2.1 typedef std::vector<DoublePtr> brathl::doubleptrarray

Creates a type name for double pointer array

7.1.2.2 typedef std::list<int32_t> brathl::intlist

Creates a type name for int std::list

7.1.2.3 typedef std::map<std::string, CParameter*> brathl::map_parameter

Creates a type name for std::map parameter base class

7.1.2.4 `typedef std::map<std::string, double> bratl::mapdouble`

Creates a type name for `std::map` pointer base class

7.1.2.5 `typedef std::map<double, DoublePtr*> bratl::mapdoubledoubleptr`

Creates a type name for `std::map` pointer base class

7.1.2.6 `typedef std::map<double, CBratObject*> bratl::mapdoubleobject`

Creates a type name for `std::map` object base class

7.1.2.7 `typedef std::map<std::string, int32_t> bratl::mapint`

Creates a type name for `std::map` int base class

7.1.2.8 `typedef std::map<int32_t, CBratObject*> bratl::mapintobject`

Creates a type name for `std::map` object base class

7.1.2.9 `typedef std::map<std::string, CBratObject*> bratl::mapobject`

Creates a type name for `std::map` object base class

7.1.2.10 `typedef std::map<std::string, void*> bratl::mapptr`

Creates a type name for `std::map` pointer base class

7.1.2.11 `typedef std::map<std::string, std::string> bratl::mapstring`

Creates a type name for `std::map` object base class

7.1.2.12 `typedef std::map<std::string, uint32_t> bratl::mapuint`

Creates a type name for `std::map` unsigned int base class

7.1.2.13 `typedef std::vector<CBratObject*> bratl::obarray`

Creates a type name for object array

7.1.2.14 `typedef std::list<CBratObject*> bratl::oblist`

Creates a type name for object `std::list`

7.1.2.15 `typedef std::stack<CBratObject*> bratl::obstack`

Creates a type name for object `std::stack`

7.1.2.16 `typedef std::vector<std::string> bratl::stringarray`

Creates a type name for `std::string` array

7.1.2.17 `typedef std::list<std::string> bratl::stringlist`

Creates a type name for `std::string` `std::list`

7.1.3 Variable Documentation

7.1.3.1 `const DefCharFunction1 bratl::CharFonctions1[]` `[static]`

Initial value:

```
=
{
    DefCharFunction1("to_date",      "Translates a std::string value into a date value"
        "\nAllowed format are:"
        "\n\n YYYY-MM-DD HH:MM:SS.MS std::string."
        "\n For instance:"
        "\n '1995-12-05 12:02:10.1230'"
        "\n '1995-12-05 12:02:10'"
        "\n '1995-12-05'"
        "\n\n a julian std::string: format:positive 'Days Seconds"
        "\n\n Seconds must be strictly less 86400 and Microseconds must be"
        "\n For instance:"
        "\n '2530 230 4569'"
        "\n\n a julian std::string: format:positive decimal julian day"
        "\n For instance:"
        "\n '850.2536985'"
        "\n\nFor julian std::string, it can contain its reference date at"
        "the end by specifying @YYYY where YYYY is the reference year"
        "that's must be one of 1950, 1958, 1985, 1990, 2000"
        "\nThe reference year YYYY stands for YYYY-01-01 00:00:00.0"
        "\nIf no reference date is specified the default reference date"
        "(1950) is used."
        "\n For instance:"
        "\n '2530 230 4569@2000'"
        "\n '850.2536985@1990'"
        "\n '850.2536985@1950' is equal to '850.2536985'"
        "\n\nDates prior to 1950-01-01 00:00:00.0 are invalid",
        NULL, CDate::CvDate, DateTime),
}
```

7.1.3.2 const DefConstant brathl::Constants[] [static]

Initial value:

```
=
{
    DefConstant("PI",      "PI value",      M_PI),
    DefConstant("PI2",     "PI/2 value",     M_PI_2),
    DefConstant("PI4",     "PI/4 value",     M_PI_4),
    DefConstant("DV",      "Default value",  CTools::m_defaultValueDOUBLE),
    DefConstant("dv",      "Default value",  CTools::m_defaultValueDOUBLE)
}
```

7.1.3.3 const DefFunction2 brathl::Fonctions2[] [static]

Initial value:

```
=
{
    DefFunction2("max",      "Calculates the larger of two values",
        CTools::Max,      Statistical),
    DefFunction2("min",      "Calculates the smaller of two values",
        CTools::Min,      Statistical),
    DefFunction2("mod",      "Calculates the floating-point remainder",
        CTools::Mod),
    DefFunction2("deg_normalize", "Normalizes longitude (degree)",
        CTools::NormalizeLongitude, Geographical),
    DefFunction2("rnd",      "Calculates the rounded value with a decimal precision (e.g"
        "round(20.23645, 3) ==> 20.236)",
        CTools::Rnd,      MathTrigo, false),
}
```

7.1.3.4 const DefFunction3 brathl::Fonctions3[] [static]

Initial value:

```
=
{
    DefFunction3("is_bounded",      "Checks if a value x is included between two value"
        "(min/max). \nis_bounded(min, x, max)",
        CTools::IsBounded,      Relational),
}
```

```

    DefFunction3("is_bounded_strict",          "Checks if a value x is stricly included between two value
(min/max). \nis_bounded_strict(min, x, max)",
                CTools::IsBoundedStrict,    Relational),
}

```

7.1.3.5 `const DefFunctionAlgoN brathl::FonctionsAlgoN[]` `[static]`

Initial value:

```

=
{
    DefFunctionAlgoN("exec",          "Execute an algorithm with variable arguments",
                    NULL,    Algorithm),
}

```

7.1.3.6 `const DefFunctionBratAlgoBaseN brathl::FonctionsBratAlgoBaseN[]` `[static]`

Initial value:

```

=
{
    DefFunctionBratAlgoBaseN(
        CBratAlgorithmBase::ExecInternal),
}

```

8 Class Documentation

8.1 `_structDateDSM` Struct Reference

```
#include <brathl.h>
```

Public Attributes

- `int32_t days`
- `int32_t muSeconds`
- `brathl_refDate refDate`
- `int32_t seconds`

8.1.1 Detailed Description

Day/seconds/microseconds date structure

8.1.2 Member Data Documentation

8.1.2.1 `int32_t _structDateDSM::days`

numbers of days

Referenced by `brathl_Julian2DSM()`, `brathl_Seconds2DSM()`, `brathl_YMDHMSM2DSM()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

8.1.2.2 `int32_t _structDateDSM::muSeconds`

numbers of microseconds

Referenced by `brathl_Julian2DSM()`, `brathl_Seconds2DSM()`, `brathl_YMDHMSM2DSM()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

8.1.2.3 `brathl_refDate _structDateDSM::refDate`

date reference (see `brathl_refDate` (p. 317))

Referenced by `brathl_Julian2DSM()`, `brathl_Seconds2DSM()`, `brathl_YMDHMSM2DSM()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

8.1.2.4 `int32_t _structDateDSM::seconds`

numbers of seconds

Referenced by `brathl_Julian2DSM()`, `brathl_Seconds2DSM()`, `brathl_YMDHMSM2DSM()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

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

- `brathl.h`

8.2 `_structDateJulian` Struct Reference

```
#include <brathl.h>
```

Public Attributes

- double `julian`
- `brathl_refDate refDate`

8.2.1 Detailed Description

Decimal julian date structure

8.2.2 Member Data Documentation

8.2.2.1 `double _structDateJulian::julian`

decimal julian day

Referenced by `brathl_DSM2Julian()`, `brathl_Seconds2Julian()`, `brathl_YMDHMSM2Julian()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

8.2.2.2 `brathl_refDate _structDateJulian::refDate`

date reference (see `brathl_refDate` (p. 317))

Referenced by `brathl_DSM2Julian()`, `brathl_Seconds2Julian()`, `brathl_YMDHMSM2Julian()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

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

- `brathl.h`

8.3 `_structDateSecond` Struct Reference

```
#include <brathl.h>
```

Public Attributes

- double **nbSeconds**
- **brathl_refDate** refDate

8.3.1 Detailed Description

Decimal seconds date structure

8.3.2 Member Data Documentation

8.3.2.1 double _structDateSecond::nbSeconds

numbers of seconds/microseconds

Referenced by `brathl_DSM2Seconds()`, `brathl_Julian2Seconds()`, `brathl_YMDHMSM2Seconds()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

8.3.2.2 brathl_refDate _structDateSecond::refDate

date reference (see **brathl_refDate** (p. 317))

Referenced by `brathl_DSM2Seconds()`, `brathl_Julian2Seconds()`, `brathl_YMDHMSM2Seconds()`, `FTN_NAME()`, and `brathl::CDate::SetDate()`.

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

- **brathl.h**

8.4 _structDateYMDHMSM Struct Reference

```
#include <brathl.h>
```

Public Attributes

- uint32_t **day**
- uint32_t **hour**
- uint32_t **minute**
- uint32_t **month**
- uint32_t **muSecond**
- uint32_t **second**
- uint32_t **year**

8.4.1 Detailed Description

YYYY-MM-DD HH:MN:SS:MS date structure

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

- **brathl.h**

8.5 brathl::CArrayDoubleArray Class Reference

```
#include <List.h>
```

Inherits `vector< std::vector< double > >`.

Public Member Functions

- **CArrayDoubleArray** ()
Empty CDoubleArray ctor.
- **CArrayDoubleArray** (const **CArrayDoubleArray** &a)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- void **GetMinMaxValues** (double &min, double &max, bool recalc=true)
- void **InitMatrix** (double initialValue=**CTools::m_defaultValueDOUBLE**)
- virtual const **CArrayDoubleArray** & **operator=** (const **CArrayDoubleArray** &m)
- virtual void **RemoveAll** ()
- void **ResizeRC** (size_t nrows, size_t ncols)
- void **Set** (const **CArrayDoubleArray** &m)
- virtual ~**CArrayDoubleArray** ()
Destructor.

Protected Member Functions

- void **AdjustValidMinMax** (double value)
- void **Init** ()

Protected Attributes

- double **m_maxValue**
- double **m_minValue**

8.5.1 Detailed Description

An array (std::vector) of std::vector of double

Version

1.0

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

- List.h
- List.cpp

8.6 brathl::CArrayDoublePtrArray Class Reference

```
#include <List.h>
```

Inherits vector< std::vector< double * > >.

Public Member Functions

- **CArrayDoublePtrArray** (bool bDelete=true)
Empty CDoubleArray ctor.
- **CArrayDoublePtrArray** (const **CArrayDoublePtrArray** &a)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- bool **GetDelete** ()

- `size_t GetMatrixDim (size_t row)`
- `CUIntArray * GetMatrixDims ()`
- `const CUIntArray * GetMatrixDims () const`
- `size_t GetMatrixNumberOfDims ()`
- `void GetMinMaxValues (double &min, double &max, bool recalc=true)`
- `void InitMatrix (double initialValue=CTools::m_defaultValueDOUBLE)`
- `void InitMatrixData (double initialValue=CTools::m_defaultValueDOUBLE)`
- `DoublePtr NewMatrix (double initialValue=CTools::m_defaultValueDOUBLE)`
- `virtual const CArrayDoublePtrArray & operator= (const CArrayDoublePtrArray &m)`
- `virtual void Remove (doubleptrarray &vect)`
- `virtual void RemoveAll ()`
- `void ResizeRC (size_t nrows, size_t ncols)`
- `void Set (const CArrayDoublePtrArray &m)`
- `void SetDelete (bool value)`
- `void SetMatrixDims (const CUIntArray &matrixDims)`
- `virtual ~CArrayDoublePtrArray ()`

Destructor.

Protected Member Functions

- `void AdjustValidMinMax (double value)`
- `void Delete (DoublePtr matrix)`
- `void Init ()`

Protected Attributes

- `bool m_bDelete`
- `CUIntArray m_matrixDims`
- `double m_maxValue`
- `double m_minValue`

8.6.1 Detailed Description

An array (`std::vector`) of `std::vector` of double pointer

Version

1.0

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

- `List.h`
- `List.cpp`

8.7 brathl::CBratAlgoFilterGaussian1D Class Reference

```
#include <BratAlgoFilterGaussian1D.h>
```

Inherits `brathl::CBratAlgoFilterGaussian`.

Public Member Functions

- **CBratAlgoFilterGaussian1D** ()
- **CBratAlgoFilterGaussian1D** (const **CBratAlgoFilterGaussian1D** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- **CBratAlgoFilterGaussian1D** & **operator=** (const **CBratAlgoFilterGaussian1D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual ~**CBratAlgoFilterGaussian1D** ()

Protected Member Functions

- virtual void **CheckVarExpression** (uint32_t index) override
- double **ComputeGaussian** ()
- void **Init** ()
- void **Set** (const **CBratAlgoFilterGaussian1D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override
- virtual void **SetNextValues** () override
- virtual void **SetPreviousValues** (bool fromProduct) override

8.7.1 Detailed Description

Algorithm base class.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 brathl::CBratAlgoFilterGaussian1D::CBratAlgoFilterGaussian1D ()

Default constructor

8.7.2.2 brathl::CBratAlgoFilterGaussian1D::CBratAlgoFilterGaussian1D (const **CBratAlgoFilterGaussian1D** ©)

Copy constructor

8.7.2.3 virtual brathl::CBratAlgoFilterGaussian1D::~~CBratAlgoFilterGaussian1D () [inline],[virtual]

Destructor

8.7.3 Member Function Documentation

8.7.3.1 void brathl::CBratAlgoFilterGaussian1D::Dump (std::ostream &fOut = std::cerr) [override],[virtual]

Dump function

Reimplemented from **brathl::CBratAlgorithmBase** (p. 129).

8.7.3.2 virtual std::string brathl::CBratAlgoFilterGaussian1D::GetDescription () const [inline],[override],[virtual]

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

8.7.3.3 `virtual std::string bratl::CBratAlgoFilterGaussian1D::GetName () const [inline],[override], [virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by Run().

8.7.3.4 `CBratAlgoFilterGaussian1D & bratl::CBratAlgoFilterGaussian1D::operator= (const CBratAlgoFilterGaussian1D & copy)`

Overloads operator '='

8.7.3.5 `double bratl::CBratAlgoFilterGaussian1D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

References bratl::CTools::Format(), and GetName().

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

- BratAlgoFilterGaussian1D.h
- BratAlgoFilterGaussian1D.cpp

8.8 bratl::CBratAlgoFilterGaussian2D Class Reference

`#include <BratAlgoFilterGaussian2D.h>`

Inherits bratl::CBratAlgoFilterGaussian.

Public Member Functions

- **CBratAlgoFilterGaussian2D** ()
- **CBratAlgoFilterGaussian2D** (const **CBratAlgoFilterGaussian2D** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- **CBratAlgoFilterGaussian2D** & **operator=** (const **CBratAlgoFilterGaussian2D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual ~**CBratAlgoFilterGaussian2D** ()

Protected Member Functions

- void **CheckProduct** ()

- virtual void **CheckVarExpression** (uint32_t index) override
- virtual double **ComputeGaussian** (CExpressionValue &exprValue)
- double **ComputeMean** ()
- double **ComputeSingle** ()
- void **Init** ()
- virtual void **OpenProductFile** () override
- void **Set** (const **CBratAlgoFilterGaussian2D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override

8.8.1 Detailed Description

Algorithm base class.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 bratl::CBratAlgoFilterGaussian2D::CBratAlgoFilterGaussian2D ()

Default constructor

8.8.2.2 bratl::CBratAlgoFilterGaussian2D::CBratAlgoFilterGaussian2D (const **CBratAlgoFilterGaussian2D** & copy)

Copy constructor

8.8.2.3 bratl::CBratAlgoFilterGaussian2D::~~CBratAlgoFilterGaussian2D () [virtual]

Destructor

8.8.3 Member Function Documentation

8.8.3.1 void bratl::CBratAlgoFilterGaussian2D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

8.8.3.2 virtual std::string bratl::CBratAlgoFilterGaussian2D::GetDescription () const [inline], [override], [virtual]

Gets the description of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 130).

8.8.3.3 virtual std::string bratl::CBratAlgoFilterGaussian2D::GetName () const [inline], [override], [virtual]

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.8.3.4 **CBratAlgoFilterGaussian2D** & bratl::CBratAlgoFilterGaussian2D::operator= (const **CBratAlgoFilterGaussian2D** & copy)

Overloads operator '='

8.8.3.5 `double bratl::CBratAlgoFilterGaussian2D::Run (CVectorBratAlgorithmParam & args)` `[override],`
`[virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

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

- BratAlgoFilterGaussian2D.h
- BratAlgoFilterGaussian2D.cpp

8.9 bratl::CBratAlgoFilterLanczos1D Class Reference

```
#include <BratAlgoFilterLanczos1D.h>
```

Inherits bratl::CBratAlgoFilterLanczos.

Public Member Functions

- **CBratAlgoFilterLanczos1D** ()
- **CBratAlgoFilterLanczos1D** (const **CBratAlgoFilterLanczos1D** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- **CBratAlgoFilterLanczos1D** & **operator=** (const **CBratAlgoFilterLanczos1D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual ~**CBratAlgoFilterLanczos1D** ()

Protected Member Functions

- virtual void **CheckVarExpression** (uint32_t index) override
- double **ComputeLanczos** ()
- void **Init** ()
- void **Set** (const **CBratAlgoFilterLanczos1D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override
- virtual void **SetNextValues** () override
- virtual void **SetPreviousValues** (bool fromProduct) override

8.9.1 Detailed Description

Algorithm base class.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 bratl::CBratAlgoFilterLanczos1D::CBratAlgoFilterLanczos1D ()

Default contructor

8.9.2.2 `bratl::CBratAlgoFilterLanczos1D::CBratAlgoFilterLanczos1D (const CBratAlgoFilterLanczos1D & copy)`

Copy constructor

8.9.2.3 `virtual bratl::CBratAlgoFilterLanczos1D::~~CBratAlgoFilterLanczos1D () [inline], [virtual]`

Destructor

8.9.3 Member Function Documentation

8.9.3.1 `void bratl::CBratAlgoFilterLanczos1D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]`

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

8.9.3.2 `virtual std::string bratl::CBratAlgoFilterLanczos1D::GetDescription () const [inline], [override], [virtual]`

Gets the description of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 130).

8.9.3.3 `virtual std::string bratl::CBratAlgoFilterLanczos1D::GetName () const [inline], [override], [virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by `Run()`.

8.9.3.4 `CBratAlgoFilterLanczos1D & bratl::CBratAlgoFilterLanczos1D::operator= (const CBratAlgoFilterLanczos1D & copy)`

Overloads operator '='

8.9.3.5 `double bratl::CBratAlgoFilterLanczos1D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

References `bratl::CTools::Format()`, and `GetName()`.

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

- `BratAlgoFilterLanczos1D.h`
- `BratAlgoFilterLanczos1D.cpp`

8.10 bratl::CBratAlgoFilterLanczos2D Class Reference

```
#include <BratAlgoFilterLanczos2D.h>
```

Inherits bratl::CBratAlgoFilterLanczos.

Public Member Functions

- **CBratAlgoFilterLanczos2D** ()
- **CBratAlgoFilterLanczos2D** (const **CBratAlgoFilterLanczos2D** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- **CBratAlgoFilterLanczos2D** & **operator=** (const **CBratAlgoFilterLanczos2D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual ~**CBratAlgoFilterLanczos2D** ()

Protected Member Functions

- void **CheckProduct** ()
- void **CheckVarExpression** (uint32_t index) override
- virtual double **ComputeLanczos** (CExpressionValue &exprValue)
- double **ComputeMean** ()
- double **ComputeSingle** ()
- void **Init** ()
- virtual void **OpenProductFile** () override
- void **Set** (const **CBratAlgoFilterLanczos2D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override

8.10.1 Detailed Description

Algorithm base class.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 bratl::CBratAlgoFilterLanczos2D::CBratAlgoFilterLanczos2D ()

Default constructor

8.10.2.2 bratl::CBratAlgoFilterLanczos2D::CBratAlgoFilterLanczos2D (const CBratAlgoFilterLanczos2D & copy)

Copy constructor

8.10.2.3 bratl::CBratAlgoFilterLanczos2D::~~CBratAlgoFilterLanczos2D () [virtual]

Destructor

8.10.3 Member Function Documentation

8.10.3.1 void bratl::CBratAlgoFilterLanczos2D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

8.10.3.2 `virtual std::string brathl::CBratAlgoFilterLanczos2D::GetDescription () const [inline],[override], [virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

8.10.3.3 `virtual std::string brathl::CBratAlgoFilterLanczos2D::GetName () const [inline],[override], [virtual]`

Gets the name of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 131).

8.10.3.4 `CBratAlgoFilterLanczos2D & brathl::CBratAlgoFilterLanczos2D::operator= (const CBratAlgoFilterLanczos2D & copy)`

Overloads operator '='

8.10.3.5 `double brathl::CBratAlgoFilterLanczos2D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **brathl::CBratAlgorithmBase** (p. 131).

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

- BratAlgoFilterLanczos2D.h
- BratAlgoFilterLanczos2D.cpp

8.11 brathl::CBratAlgoFilterLoess1D Class Reference

`#include <BratAlgoFilterLoess1D.h>`

Inherits **brathl::CBratAlgoFilterLoess**.

Public Member Functions

- **CBratAlgoFilterLoess1D** ()
- **CBratAlgoFilterLoess1D** (const **CBratAlgoFilterLoess1D** ©)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual **CBratAlgorithmParam::bratAlgoParamTypeVal** **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override

- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual double **GetParamDefaultValue** (uint32_t indexParam) const override
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **CBratAlgoFilterLoess1D** & **operator=** (const **CBratAlgoFilterLoess1D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgoFilterLoess1D** ()

Protected Member Functions

- double **ApplyFilter** ()
- virtual void **CheckVarExpression** (uint32_t index) override
- double **ComputeLoess** ()
- void **FitLinearEst** (const double x, const double c0, const double c1, const double cov00, const double cov01, const double cov11, double *y, double *y_err)
- void **FitWLinear** (const double *x, const uint32_t xstride, const double *w, const uint32_t wstride, const double *y, const uint32_t ystride, const uint32_t n, double *c0, double *c1, double *cov_00, double *cov_01, double *cov_11, double *chisq)
- void **Init** ()
- virtual void **InsertCurrentValueDataWindow1D** () override
- virtual void **RemoveFirstItemDataWindow1D** () override
- void **Set** (const **CBratAlgoFilterLoess1D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override
- virtual void **SetNextValues** () override
- virtual void **SetPreviousValues** (bool fromProduct) override
- virtual void **TreatLeftEdge1D** (uint32_t shiftSymmetry, uint32_t index) override
- virtual void **TreatRightEdge1D** (uint32_t shiftSymmetry, uint32_t index) override
- double **Tricube** (double u, double t)

Protected Attributes

- CDoubleArray **m_distances**
- CDoubleArray **m_sortedDistances**
- CDoubleArray **m_xDataWindow**
- double **m_xValue**
- double **m_xValueNext**
- double **m_xValuePrev**

Static Protected Attributes

- static const uint32_t **m_EXTRAPOLATE_PARAM_INDEX**
- static const uint32_t **m_INPUT_PARAMS** = 4
- static const uint32_t **m_VALID_PARAM_INDEX** = 3
- static const uint32_t **m_WINDOW_PARAM_INDEX** = 2
- static const uint32_t **m_X_PARAM_INDEX** = 1

8.11.1 Detailed Description

Algorithm base class.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `brathl::CBratAlgoFilterLoess1D::CBratAlgoFilterLoess1D ()`

Default constructor

8.11.2.2 `brathl::CBratAlgoFilterLoess1D::CBratAlgoFilterLoess1D (const CBratAlgoFilterLoess1D & copy)`

Copy constructor

8.11.2.3 `virtual brathl::CBratAlgoFilterLoess1D::~~CBratAlgoFilterLoess1D () [inline], [virtual]`

Destructor

8.11.3 Member Function Documentation

8.11.3.1 `void brathl::CBratAlgoFilterLoess1D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmBase** (p. 129).

8.11.3.2 `virtual std::string brathl::CBratAlgoFilterLoess1D::GetDescription () const [inline], [override], [virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

8.11.3.3 `virtual std::string brathl::CBratAlgoFilterLoess1D::GetInputParamDesc (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.11.3.4 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::CBratAlgoFilterLoess1D::GetInputParamFormat (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the format of an input parameter : `CBratAlgorithmParam::T_DOUBLE` for double `CBratAlgorithmParam::T_FLOAT` for float `CBratAlgorithmParam::T_INT` for integer `CBratAlgorithmParam::T_LONG` for long integer `CBratAlgorithmParam::T_STRING` for `std::string` `CBratAlgorithmParam::T_CHAR` for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.11.3.5 `virtual std::string brathl::CBratAlgoFilterLoess1D::GetInputParamUnit (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 130).

References bratl::CTools::Format(), and GetNumInputParam().

8.11.3.6 `virtual std::string bratl::CBratAlgoFilterLoess1D::GetName () const [inline],[override], [virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by Run().

8.11.3.7 `virtual uint32_t bratl::CBratAlgoFilterLoess1D::GetNumInputParam () const [inline],[override], [virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.11.3.8 `virtual std::string bratl::CBratAlgoFilterLoess1D::GetOutputUnit () const [inline],[override], [virtual]`

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.11.3.9 `CBratAlgoFilterLoess1D & bratl::CBratAlgoFilterLoess1D::operator= (const CBratAlgoFilterLoess1D & copy)`

Overloads operator '='

8.11.3.10 `double bratl::CBratAlgoFilterLoess1D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

References bratl::CTools::Format(), and GetName().

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

- BratAlgoFilterLoess1D.h
- BratAlgoFilterLoess1D.cpp

8.12 bratl::CBratAlgoFilterLoess2D Class Reference

```
#include <BratAlgoFilterLoess2D.h>
```

Inherits bratl::CBratAlgoFilterLoess.

Public Member Functions

- **CBratAlgoFilterLoess2D** ()
- **CBratAlgoFilterLoess2D** (const **CBratAlgoFilterLoess2D** ©)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual double **GetParamDefaultValue** (uint32_t indexParam) const override
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **CBratAlgoFilterLoess2D** & **operator=** (const **CBratAlgoFilterLoess2D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgoFilterLoess2D** ()

Protected Member Functions

- double **ApplyFilter** ()
- void **CheckProduct** ()
- void **CheckVarExpression** (uint32_t index) override
- void **ComputeInitialWeights** ()
- double **ComputeLoess** ()
- double **ComputeMean** ()
- double **ComputeSingle** ()
- void **Init** ()
- virtual void **OpenProductFile** () override
- void **PrepareDataValues** ()
- void **PrepareDataWindow** ()
- void **Set** (const **CBratAlgoFilterLoess2D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override

Static Protected Attributes

- static const uint32_t **m_EXTRAPOLATE_PARAM_INDEX** = 4
- static const uint32_t **m_INPUT_PARAMS** = 5
- static const uint32_t **m_VALID_PARAM_INDEX** = 3
- static const uint32_t **m_WINDOW_HEIGHT_PARAM_INDEX** = 2
- static const uint32_t **m_WINDOW_WIDTH_PARAM_INDEX** = 1

8.12.1 Detailed Description

Algorithm base class.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 bratl::CBratAlgoFilterLoess2D::CBratAlgoFilterLoess2D ()

Default constructor

8.12.2.2 bratl::CBratAlgoFilterLoess2D::CBratAlgoFilterLoess2D (const CBratAlgoFilterLoess2D & copy)

Copy constructor

8.12.2.3 bratl::CBratAlgoFilterLoess2D::~CBratAlgoFilterLoess2D () [virtual]

Destructor

8.12.3 Member Function Documentation

8.12.3.1 void bratl::CBratAlgoFilterLoess2D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

8.12.3.2 virtual std::string bratl::CBratAlgoFilterLoess2D::GetDescription () const [inline], [override], [virtual]

Gets the description of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 130).

8.12.3.3 virtual std::string bratl::CBratAlgoFilterLoess2D::GetInputParamDesc (uint32_t indexParam) const [inline], [override], [virtual]

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **GetNumInputParam()**.

8.12.3.4 virtual CBratAlgorithmParam::bratAlgoParamTypeVal bratl::CBratAlgoFilterLoess2D::GetInputParamFormat (uint32_t indexParam) const [inline], [override], [virtual]

Gets the format of an input parameter : CBratAlgorithmParam::T_DOUBLE for double CBratAlgorithmParam::T_↔ FLOAT for float CBratAlgorithmParam::T_INT for integer CBratAlgorithmParam::T_LONG for long integer CBrat↔ AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **GetNumInputParam()**.

8.12.3.5 `virtual std::string bratl::CBratAlgoFilterLoess2D::GetInputParamUnit (uint32_t indexParam) const` `[inline],`
`[override], [virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **GetNumInputParam()**.

8.12.3.6 `virtual std::string bratl::CBratAlgoFilterLoess2D::GetName () const` `[inline]`, `[override]`, `[virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.12.3.7 `virtual uint32_t bratl::CBratAlgoFilterLoess2D::GetNumInputParam () const` `[inline]`, `[override]`, `[virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by **GetInputParamDesc()**, **GetInputParamFormat()**, and **GetInputParamUnit()**.

8.12.3.8 `virtual std::string bratl::CBratAlgoFilterLoess2D::GetOutputUnit () const` `[inline]`, `[override]`, `[virtual]`

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.12.3.9 `CBratAlgoFilterLoess2D & bratl::CBratAlgoFilterLoess2D::operator= (const CBratAlgoFilterLoess2D & copy)`

Overloads operator '='

8.12.3.10 `double bratl::CBratAlgoFilterLoess2D::Run (CVectorBratAlgorithmParam & args)` `[override]`, `[virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

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

- BratAlgoFilterLoess2D.h
- BratAlgoFilterLoess2D.cpp

8.13 bratl::CBratAlgoFilterMedian1D Class Reference

```
#include <BratAlgoFilterMedian1D.h>
```

Inherits `brathl::CBratAlgoFilterMedian`.

Public Member Functions

- **CBratAlgoFilterMedian1D** ()
- **CBratAlgoFilterMedian1D** (const **CBratAlgoFilterMedian1D** ©)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual double **GetParamDefaultValue** (uint32_t indexParam)
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **CBratAlgoFilterMedian1D** & **operator=** (const **CBratAlgoFilterMedian1D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgoFilterMedian1D** ()

Protected Member Functions

- virtual void **CheckVarExpression** (uint32_t index) override
- void **Init** ()
- void **Set** (const **CBratAlgoFilterMedian1D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override
- virtual void **SetNextValues** () override
- virtual void **SetPreviousValues** (bool fromProduct) override

Static Protected Attributes

- static const uint32_t **m_EXTRAPOLATE_PARAM_INDEX** = 3
- static const uint32_t **m_INPUT_PARAMS** = 4
- static const uint32_t **m_VALID_PARAM_INDEX** = 2
- static const uint32_t **m_WINDOW_PARAM_INDEX** = 1

8.13.1 Detailed Description

Algorithm base class.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 `brathl::CBratAlgoFilterMedian1D::CBratAlgoFilterMedian1D ()`

Default constructor

8.13.2.2 `brathl::CBratAlgoFilterMedian1D::CBratAlgoFilterMedian1D (const CBratAlgoFilterMedian1D & copy)`

Copy constructor

8.13.2.3 `virtual bratl::CBratAlgoFilterMedian1D::~~CBratAlgoFilterMedian1D () [inline],[virtual]`

Destructor

8.13.3 Member Function Documentation

8.13.3.1 `void bratl::CBratAlgoFilterMedian1D::Dump (std::ostream & fOut = std::cerr) [override],[virtual]`

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

8.13.3.2 `virtual std::string bratl::CBratAlgoFilterMedian1D::GetDescription () const [inline],[override],[virtual]`

Gets the description of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 130).

8.13.3.3 `virtual std::string bratl::CBratAlgoFilterMedian1D::GetInputParamDesc (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References `bratl::CTools::Format()`, and `GetNumInputParam()`.

8.13.3.4 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal bratl::CBratAlgoFilterMedian1D::GetInputParamFormat (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the format of an input parameter : `CBratAlgorithmParam::T_DOUBLE` for double `CBratAlgorithmParam::T_FLOAT` for float `CBratAlgorithmParam::T_INT` for integer `CBratAlgorithmParam::T_LONG` for long integer `CBratAlgorithmParam::T_STRING` for `std::string` `CBratAlgorithmParam::T_CHAR` for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References `bratl::CTools::Format()`, and `GetNumInputParam()`.

8.13.3.5 `virtual std::string bratl::CBratAlgoFilterMedian1D::GetInputParamUnit (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 130).

References `bratl::CTools::Format()`, and `GetNumInputParam()`.

8.13.3.6 `virtual std::string bratl::CBratAlgoFilterMedian1D::GetName () const [inline],[override],[virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by Run().

8.13.3.7 `virtual uint32_t bratl::CBratAlgoFilterMedian1D::GetNumInputParam () const [inline],[override], [virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.13.3.8 `virtual std::string bratl::CBratAlgoFilterMedian1D::GetOutputUnit () const [inline],[override], [virtual]`

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.13.3.9 `CBratAlgoFilterMedian1D & bratl::CBratAlgoFilterMedian1D::operator= (const CBratAlgoFilterMedian1D & copy)`

Overloads operator '='

8.13.3.10 `double bratl::CBratAlgoFilterMedian1D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

References bratl::CTools::Format(), and GetName().

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

- BratAlgoFilterMedian1D.h
- BratAlgoFilterMedian1D.cpp

8.14 bratl::CBratAlgoFilterMedian2D Class Reference

`#include <BratAlgoFilterMedian2D.h>`

Inherits bratl::CBratAlgoFilterMedian.

Public Member Functions

- **CBratAlgoFilterMedian2D** ()
- **CBratAlgoFilterMedian2D** (const **CBratAlgoFilterMedian2D** ©)

- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual uint32_t **GetDataWindowSize** () override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual double **GetParamDefaultValue** (uint32_t indexParam) const override
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **CBratAlgoFilterMedian2D & operator=** (const **CBratAlgoFilterMedian2D** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgoFilterMedian2D** ()

Protected Member Functions

- void **CheckProduct** ()
- void **CheckVarExpression** (uint32_t index) override
- double **ComputeMean** ()
- double **ComputeSingle** ()
- void **Init** ()
- virtual void **OpenProductFile** () override
- void **PrepareDataValues** ()
- void **PrepareDataWindow** ()
- void **Set** (const **CBratAlgoFilterMedian2D** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override

Static Protected Attributes

- static const uint32_t **m_EXTRAPOLATE_PARAM_INDEX** = 4
- static const uint32_t **m_INPUT_PARAMS** = 5
- static const uint32_t **m_VALID_PARAM_INDEX** = 3
- static const uint32_t **m_WINDOW_HEIGHT_PARAM_INDEX** = 2
- static const uint32_t **m_WINDOW_WIDTH_PARAM_INDEX** = 1

8.14.1 Detailed Description

Algorithm base class.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 brathl::CBratAlgoFilterMedian2D::CBratAlgoFilterMedian2D ()

Default constructor

8.14.2.2 brathl::CBratAlgoFilterMedian2D::CBratAlgoFilterMedian2D (const **CBratAlgoFilterMedian2D** ©)

Copy constructor

8.14.2.3 `brathl::CBratAlgoFilterMedian2D::~~CBratAlgoFilterMedian2D () [virtual]`

Destructor

8.14.3 Member Function Documentation

8.14.3.1 `void brathl::CBratAlgoFilterMedian2D::Dump (std::ostream & fOut = std::cerr) [override], [virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmBase** (p. 129).

8.14.3.2 `virtual std::string brathl::CBratAlgoFilterMedian2D::GetDescription () const [inline], [override], [virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

8.14.3.3 `virtual std::string brathl::CBratAlgoFilterMedian2D::GetInputParamDesc (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.14.3.4 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::CBratAlgoFilterMedian2D::GetInputParamFormat (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the format of an input parameter : `CBratAlgorithmParam::T_DOUBLE` for double `CBratAlgorithmParam::T_FLOAT` for float `CBratAlgorithmParam::T_INT` for integer `CBratAlgorithmParam::T_LONG` for long integer `CBratAlgorithmParam::T_STRING` for `std::string` `CBratAlgorithmParam::T_CHAR` for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.14.3.5 `virtual std::string brathl::CBratAlgoFilterMedian2D::GetInputParamUnit (uint32_t indexParam) const [inline], [override], [virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.14.3.6 `virtual std::string brathl::CBratAlgoFilterMedian2D::GetName () const [inline], [override], [virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.14.3.7 `virtual uint32_t bratl::CBratAlgoFilterMedian2D::GetNumInputParam () const [inline],[override], [virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by `GetInputParamDesc()`, `GetInputParamFormat()`, and `GetInputParamUnit()`.

8.14.3.8 `virtual std::string bratl::CBratAlgoFilterMedian2D::GetOutputUnit () const [inline],[override], [virtual]`

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.14.3.9 `CBratAlgoFilterMedian2D & bratl::CBratAlgoFilterMedian2D::operator= (const CBratAlgoFilterMedian2D & copy)`

Overloads operator '='

8.14.3.10 `double bratl::CBratAlgoFilterMedian2D::Run (CVectorBratAlgorithmParam & args) [override], [virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

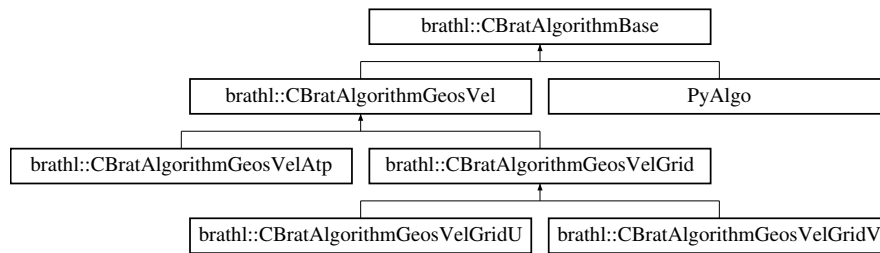
The documentation for this class was generated from the following files:

- BratAlgoFilterMedian2D.h
- BratAlgoFilterMedian2D.cpp

8.15 bratl::CBratAlgorithmBase Class Reference

```
#include <BratAlgorithmBase.h>
```

Inheritance diagram for `bratl::CBratAlgorithmBase`:



Public Member Functions

- **CBratAlgorithmBase** ()
- **CBratAlgorithmBase** (const **CBratAlgorithmBase** &o)
- void **CheckConstantParam** (uint32_t indexParam)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args)
- virtual void **CheckInputTypeParams** (uint32_t index, CBratAlgorithmParam::bratAlgoParamTypeVal expectedType, CVectorBratAlgorithmParam &args)
- virtual void **CheckInputTypeParams** (uint32_t index, const CIntArray &expectedTypes, CVectorBratAlgorithmParam &args)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
- std::string **GetAlgoExpression** ()
- **CObArray** * **GetAlgoParamExpressions** ()
- virtual std::string **GetDescription** () const =0
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const =0
- std::string **GetInputParamDescWithDefValueLabel** (uint32_t indexParam)
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const =0
- virtual std::string **GetInputParamFormatAsString** (uint32_t indexParam)
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const =0
- virtual std::string **GetName** () const =0
- virtual uint32_t **GetNumInputParam** () const =0
- virtual std::string **GetOutputUnit** () const =0
- virtual double **GetParamDefaultValue** (uint32_t) const
- void **GetParamDefValue** (uint32_t indexParam, double &value)
- void **GetParamDefValue** (uint32_t indexParam, float &value)
- void **GetParamDefValue** (uint32_t indexParam, uint32_t &value)
- void **GetParamDefValue** (uint32_t indexParam, uint64_t &value)
- void **GetParamDefValue** (uint32_t indexParam, int32_t &value)
- void **GetParamDefValue** (uint32_t indexParam, int64_t &value)
- std::string **GetParamDefValueAsLabel** (uint32_t indexParam)
- std::string **GetParamDefValueAsString** (uint32_t indexParam)
- virtual std::string **GetParamName** (uint32_t) const =0
- **CProductNetCdf** * **GetProductNetCdf** (CProduct *product)
- std::string **GetSyntax** () const
- **CBratAlgorithmBase** & **operator=** (const **CBratAlgorithmBase** &o)
- virtual double **Run** (CVectorBratAlgorithmParam &args)=0
- void **SetAlgoExpression** (const std::string &value)
- void **SetAlgoParamExpressions** (const CStringArray &values)
- void **SetAlgoParamExpressions** (const **CObArray** &obArray)
- virtual void **SetProduct** (CProduct *product, bool forceReplace=false)
- virtual ~**CBratAlgorithmBase** ()

Static Public Member Functions

- static double **ExecInternal** (CBratAlgorithmBase *algo, CVectorBratAlgorithmParam &arg)
- static CBratAlgorithmBase * **GetNew** (const char *algorithmName)
- static void **RegisterCAlgorithms** ()

Protected Member Functions

- void **AddXOrYFieldDependency** (CFieldNetCdf *field, CFieldNetCdf *field2DAsRef)
- void **AddXOrYFieldDependency** (CFieldNetCdf *field, const std::string &xDimName, const std::string &yDimName)
- virtual void **CheckComplexExpression** (uint32_t index)
- virtual void **CheckVarExpression2D** (uint32_t index)
- virtual void **DeleteExpressionValuesArray** ()
- virtual void **DeleteFieldNetCdf** ()
- virtual void **DeleteProduct** ()
- virtual void **GetAllData** (CExpression *expression, CDoubleArray &data)
- virtual void **GetData1D** (int32_t iRecord)
- CObArray * **GetDataExpressionValues** (uint32_t indexExpr)
- double **GetDataValue** (uint32_t indexExpr)
- double **GetDataValue** (uint32_t indexExpr, uint32_t x)
- double **GetDataValue** (uint32_t indexExpr, uint32_t x, uint32_t y)
- void **GetExpressionDataValuesAsArrayOfSingleValue** (uint32_t indexExpr, double *&values, uint32_t &nbValues)
- CFieldNetCdf * **GetField2DAsRef** ()
- virtual void **GetNextData** ()
- void **Init** ()
- void **InitComplexExpressionArray** ()
- virtual void **NewExpressionValuesArray** ()
- virtual void **OpenProductFile** ()
- virtual void **OpenProductFile** (CProduct *product)
- virtual void **PrepareDataValues2DComplexExpression** (CExpressionValue &exprValue, uint32_t algoExprIndex)
- virtual void **PrepareDataValues2DComplexExpressionWithAlgo** (CExpressionValue &exprValue, uint32_t algoExprIndex)
- virtual void **PrepareDataValues2DOneField** (CExpressionValue &exprValue, uint32_t algoExprIndex)
- virtual void **ProcessOpeningProductNetCdf** ()
- virtual void **ProcessOpeningProductNetCdf** (CProduct *product)
- virtual uint32_t **ReadProductData** (int32_t iRecord)
- virtual uint32_t **ReadProductData** (int32_t iRecord, CExpression *expression)
- virtual uint32_t **ReadProductData** (int32_t iRecord, const CObArrayOb &algoParamExpressions)
- virtual uint32_t **ReadProductData** (CProduct *product, int32_t iRecord, const CObArrayOb &arrayExpressions)
- void **Set** (const CBratAlgorithmBase &o)
- virtual void **SetBeginOfFile** ()
- virtual void **SetEndOfFile** ()
- void **SetField2DAsRef** ()
- virtual void **SetNextValues** ()
- virtual void **SetPreviousValues** (bool fromProduct)

Protected Attributes

- `std::string m_algoExpression`
- `CObArrayOb m_algoParamExpressions`
- `CProduct * m_callerProduct`
- `int32_t m_callerProductRecordPrev`
- `std::string m_currentFileName`
- `CIntArray m_expectedTypes`
- `CObArray * m_expressionValuesArray`
- `CFieldNetCdf * m_field2DAsRef`
- `CObMap m_fieldDependOnXDim`
- `CObMap m_fieldDependOnXYDim`
- `CObMap m_fieldDependOnYDim`
- `CObMap m_fieldVars`
- `CObMap m_fieldVarsCaller`
- `int32_t m_indexRecordToRead`
- `std::vector< bool > m_isComplexExpression`
- `std::vector< bool > m_isComplexExpressionWithAlgo`
- `CStringList m_listFieldsToRead`
- `int32_t m_nProductRecords`
- `CProduct * m_product`
- `CDoubleArray * m_varValueArray`

Static Protected Attributes

- `static bool m_algorithmsRegistered = false`

8.15.1 Detailed Description

Algorithm base class.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `bratl::CBratAlgorithmBase::CBratAlgorithmBase ()`

Default constructor

8.15.2.2 `bratl::CBratAlgorithmBase::CBratAlgorithmBase (const CBratAlgorithmBase & o)`

Copy constructor

8.15.2.3 `bratl::CBratAlgorithmBase::~~CBratAlgorithmBase () [virtual]`

Destructor

8.15.3 Member Function Documentation

8.15.3.1 `void bratl::CBratAlgorithmBase::Dump (std::ostream & fOut = std::cerr) [virtual]`

Dump function

Reimplemented in `bratl::CBratAlgorithmGeosVelGridV` (p. 15), `bratl::CBratAlgorithmGeosVelGridU` (p. 15), `bratl::CBratAlgoFilterLoess1D` (p. 115), `bratl::CBratAlgoFilterLoess2D` (p. 118), `bratl::CBratAlgoFilterMedian2D` (p. 125), `bratl::CBratAlgoFilterMedian1D` (p. 122), `bratl::CBratAlgorithmGeosVelGrid`

(p. 15), **bratl::CBratAlgorithmGeosVelAtp** (p. 135), **bratl::CBratAlgoFilterGaussian2D** (p. 108), **bratl::CBratAlgoFilterLanczos2D** (p. 112), **bratl::CBratAlgoFilterGaussian1D** (p. 106), **bratl::CBratAlgoFilterLanczos1D** (p. 111), and **bratl::CBratAlgorithmGeosVel** (p. 133).

References **bratl::CObArray::Dump()**, **bratl::CObMap::Dump()**, **GetDescription()**, **GetInputParamDesc()**, **GetInputParamFormat()**, **GetInputParamUnit()**, **GetName()**, **GetNumInputParam()**, and **GetOutputUnit()**.

Referenced by **bratl::CBratAlgorithmGeosVel::Dump()**.

8.15.3.2 virtual std::string bratl::CBratAlgorithmBase::GetDescription () const [pure virtual]

Gets the description of the algorithm

Implemented in **PyAlgo** (p. 311), **bratl::CBratAlgorithmGeosVelGridV** (p. 15), **bratl::CBratAlgorithmGeosVelGridU** (p. 15), **bratl::CBratAlgoFilterGaussian1D** (p. 106), **bratl::CBratAlgoFilterGaussian2D** (p. 108), **bratl::CBratAlgoFilterLanczos1D** (p. 111), **bratl::CBratAlgoFilterLanczos2D** (p. 113), **bratl::CBratAlgoFilterLoess1D** (p. 115), **bratl::CBratAlgoFilterLoess2D** (p. 118), **bratl::CBratAlgoFilterMedian1D** (p. 122), **bratl::CBratAlgoFilterMedian2D** (p. 125), and **bratl::CBratAlgorithmGeosVelAtp** (p. 135).

Referenced by **Dump()**.

8.15.3.3 virtual std::string bratl::CBratAlgorithmBase::GetInputParamDesc (uint32_t indexParam) const [pure virtual]

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implemented in **PyAlgo** (p. 312), **bratl::CBratAlgoFilterLoess1D** (p. 115), **bratl::CBratAlgoFilterLoess2D** (p. 118), **bratl::CBratAlgoFilterMedian1D** (p. 122), **bratl::CBratAlgoFilterMedian2D** (p. 125), **bratl::CBratAlgorithmGeosVelAtp** (p. 135), and **bratl::CBratAlgorithmGeosVelGrid** (p. 15).

Referenced by **Dump()**.

8.15.3.4 virtual CBratAlgorithmParam::bratAlgoParamTypeVal bratl::CBratAlgorithmBase::GetInputParamFormat (uint32_t indexParam) const [pure virtual]

Gets the format of an input parameter : **CBratAlgorithmParam::T_DOUBLE** for double **CBratAlgorithmParam::T_FLOAT** for float **CBratAlgorithmParam::T_INT** for integer **CBratAlgorithmParam::T_LONG** for long integer **CBratAlgorithmParam::T_STRING** for std::string **CBratAlgorithmParam::T_CHAR** for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implemented in **PyAlgo** (p. 313), **bratl::CBratAlgoFilterLoess1D** (p. 115), **bratl::CBratAlgoFilterLoess2D** (p. 118), **bratl::CBratAlgoFilterMedian2D** (p. 125), **bratl::CBratAlgoFilterMedian1D** (p. 122), **bratl::CBratAlgorithmGeosVelAtp** (p. 136), and **bratl::CBratAlgorithmGeosVelGrid** (p. 15).

Referenced by **Dump()**.

8.15.3.5 virtual std::string bratl::CBratAlgorithmBase::GetInputParamUnit (uint32_t indexParam) const [pure virtual]

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implemented in **PyAlgo** (p. 313), **bratl::CBratAlgoFilterLoess1D** (p. 115), **bratl::CBratAlgoFilterLoess2D** (p. 119), **bratl::CBratAlgoFilterMedian2D** (p. 125), **bratl::CBratAlgoFilterMedian1D** (p. 122), **bratl::CBratAlgorithmGeosVelAtp** (p. 136), and **bratl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by **Dump()**.

8.15.3.6 `virtual std::string bratl::CBratAlgorithmBase::GetName () const [pure virtual]`

Gets the name of the algorithm

Implemented in **PyAlgo** (p. 313), **bratl::CBratAlgorithmGeosVelGridV** (p. 16), **bratl::CBratAlgorithmGeosVelGridU** (p. 16), **bratl::CBratAlgoFilterGaussian1D** (p. 107), **bratl::CBratAlgoFilterGaussian2D** (p. 108), **bratl::CBratAlgoFilterLanczos1D** (p. 111), **bratl::CBratAlgoFilterLanczos2D** (p. 113), **bratl::CBratAlgoFilterLoess1D** (p. 116), **bratl::CBratAlgoFilterLoess2D** (p. 120), **bratl::CBratAlgoFilterMedian1D** (p. 122), **bratl::CBratAlgoFilterMedian2D** (p. 125), and **bratl::CBratAlgorithmGeosVelAtp** (p. 136).

Referenced by Dump().

8.15.3.7 `virtual uint32_t bratl::CBratAlgorithmBase::GetNumInputParam () const [pure virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implemented in **PyAlgo** (p. 313), **bratl::CBratAlgoFilterLoess1D** (p. 116), **bratl::CBratAlgoFilterLoess2D** (p. 120), **bratl::CBratAlgoFilterMedian1D** (p. 123), **bratl::CBratAlgoFilterMedian2D** (p. 126), **bratl::CBratAlgorithmGeosVelAtp** (p. 136), and **bratl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by Dump().

8.15.3.8 `virtual std::string bratl::CBratAlgorithmBase::GetOutputUnit () const [pure virtual]`

Gets the unit of an output value returned by the 'Run' function.

Implemented in **PyAlgo** (p. 313), **bratl::CBratAlgoFilterLoess1D** (p. 116), **bratl::CBratAlgoFilterLoess2D** (p. 120), **bratl::CBratAlgoFilterMedian2D** (p. 126), **bratl::CBratAlgoFilterMedian1D** (p. 123), **bratl::CBratAlgorithmGeosVelAtp** (p. 136), and **bratl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by Dump().

8.15.3.9 `CBratAlgorithmBase & bratl::CBratAlgorithmBase::operator= (const CBratAlgorithmBase & o)`

Overloads operator '='

8.15.3.10 `virtual double bratl::CBratAlgorithmBase::Run (CVectorBratAlgorithmParam & args) [pure virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implemented in **PyAlgo** (p. 314), **bratl::CBratAlgoFilterLoess1D** (p. 116), **bratl::CBratAlgoFilterLoess2D** (p. 120), **bratl::CBratAlgoFilterMedian2D** (p. 126), **bratl::CBratAlgoFilterMedian1D** (p. 123), **bratl::CBratAlgorithmGeosVelAtp** (p. 136), **bratl::CBratAlgorithmGeosVelGrid** (p. 16), **bratl::CBratAlgoFilterGaussian1D** (p. 107), **bratl::CBratAlgoFilterGaussian2D** (p. 109), **bratl::CBratAlgoFilterLanczos1D** (p. 111), and **bratl::CBratAlgoFilterLanczos2D** (p. 113).

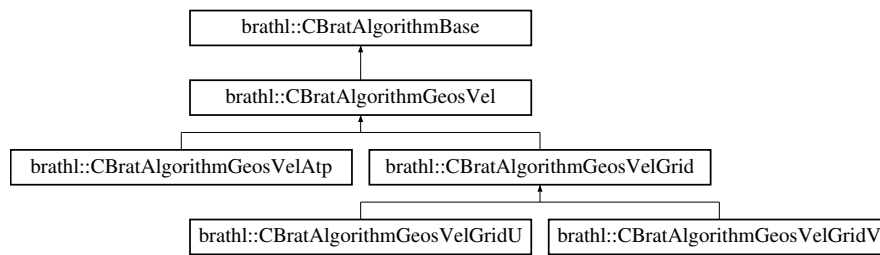
The documentation for this class was generated from the following files:

- BratAlgorithmBase.h
- BratAlgorithmBase.cpp

8.16 bratl::CBratAlgorithmGeosVel Class Reference

```
#include <BratAlgorithmGeosVel.h>
```

Inheritance diagram for bratl::CBratAlgorithmGeosVel:



Public Member Functions

- void **BtoE** (double lonPlane, double latPlane, double betaX, double betaY, double &lon, double &lat)
- **CBratAlgorithmGeosVel** ()
- **CBratAlgorithmGeosVel** (const **CBratAlgorithmGeosVel** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
- void **EtoB** (double lonPlane, double latPlane, double lon, double lat, double &betaX, double &betaY)
- **CBratAlgorithmGeosVel** & **operator=** (const **CBratAlgorithmGeosVel** ©)
- virtual ~**CBratAlgorithmGeosVel** ()

Protected Member Functions

- virtual void **ComputeCoriolis** ()
- void **Init** ()
- void **Set** (const **CBratAlgorithmGeosVel** &o)
- void **SetBeginOfFile** ()
- void **SetEndOfFile** ()
- virtual void **SetNextValues** ()
- virtual void **SetPreviousValues** (bool fromProduct)

Protected Attributes

- double **m_beta**
- double **m_coriolis**
- double **m_degreeToRadianMultiplier**
- double **m_earthRadius**
- bool **m_equatorTransition**
- bool **m_equatorTransitionIsNext**
- double **m_gravity**
- double **m_lat**
- CDoubleArray * **m_latArray**
- double **m_latNext**
- double **m_latPrev**
- double **m_lon**
- CDoubleArray * **m_lonArray**
- double **m_lonNext**
- double **m_lonPrev**
- double **m_omega**
- double **m_p2**
- double **m_velocity**

Static Protected Attributes

- static const std::string **m_LAT_PARAM_NAME** = "%{lat}"
- static const std::string **m_LON_PARAM_NAME** = "%{lon}"

Additional Inherited Members

8.16.1 Detailed Description

Algorithm base class.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 `bratl::CBratAlgorithmGeosVel::CBratAlgorithmGeosVel ()`

Default constructor

8.16.2.2 `bratl::CBratAlgorithmGeosVel::CBratAlgorithmGeosVel (const CBratAlgorithmGeosVel & copy)`

Copy constructor

8.16.2.3 `bratl::CBratAlgorithmGeosVel::~~CBratAlgorithmGeosVel () [virtual]`

Destructor

8.16.3 Member Function Documentation

8.16.3.1 `void bratl::CBratAlgorithmGeosVel::Dump (std::ostream & fOut = std::cerr) [virtual]`

Dump function

Reimplemented from **bratl::CBratAlgorithmBase** (p. 129).

Reimplemented in **bratl::CBratAlgorithmGeosVelGridV** (p. 15), **bratl::CBratAlgorithmGeosVelGridU** (p. 15), **bratl::CBratAlgorithmGeosVelGrid** (p. 15), and **bratl::CBratAlgorithmGeosVelAtp** (p. 135).

References `bratl::CBratAlgorithmBase::Dump()`.

Referenced by `bratl::CBratAlgorithmGeosVelAtp::Dump()`, and `bratl::CBratAlgorithmGeosVelGrid::Dump()`.

8.16.3.2 `CBratAlgorithmGeosVel & bratl::CBratAlgorithmGeosVel::operator= (const CBratAlgorithmGeosVel & copy)`

Overloads operator '='

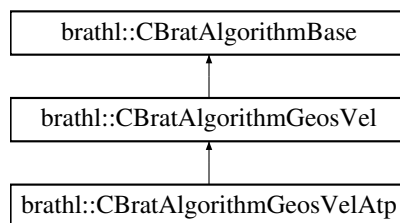
The documentation for this class was generated from the following files:

- `BratAlgorithmGeosVel.h`
- `BratAlgorithmGeosVel.cpp`

8.17 `bratl::CBratAlgorithmGeosVelAtp` Class Reference

```
#include <BratAlgorithmGeosVelAtp.h>
```

Inheritance diagram for `bratl::CBratAlgorithmGeosVelAtp`:



Public Member Functions

- **CBratAlgorithmGeosVelAtp** ()
- **CBratAlgorithmGeosVelAtp** (const **CBratAlgorithmGeosVelAtp** ©)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- double **GetTrackDirection** ()
- **CBratAlgorithmGeosVelAtp** & **operator=** (const **CBratAlgorithmGeosVelAtp** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgorithmGeosVelAtp** ()

Protected Member Functions

- double **ComputeVelocity** ()
- double **ComputeVelocityEquator** ()
- double **ComputeVelocityOutsideEquator** ()
- void **Init** ()
- void **Set** (const **CBratAlgorithmGeosVelAtp** ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override
- void **SetEquatorTransition** ()
- void **SetGap** ()
- virtual void **SetNextValues** () override
- virtual void **SetPreviousValues** (bool fromProduct) override

Protected Attributes

- double **m_gap**
- double **m_varValue**
- double **m_varValueNext**
- double **m_varValuePrev**

Static Protected Attributes

- static const uint32_t **m_INPUT_PARAMS** = 3
- static const uint32_t **m_LAT_PARAM_INDEX** = 0
- static const uint32_t **m_LON_PARAM_INDEX** = 1
- static const uint32_t **m_VAR_PARAM_INDEX** = 2

Additional Inherited Members

8.17.1 Detailed Description

Algorithm base class.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `brathl::CBratAlgorithmGeosVelAtp::CBratAlgorithmGeosVelAtp ()`

Default constructor

8.17.2.2 `brathl::CBratAlgorithmGeosVelAtp::CBratAlgorithmGeosVelAtp (const CBratAlgorithmGeosVelAtp & copy)`

Copy constructor

8.17.2.3 `virtual brathl::CBratAlgorithmGeosVelAtp::~~CBratAlgorithmGeosVelAtp () [inline],[virtual]`

Destructor

8.17.3 Member Function Documentation

8.17.3.1 `void brathl::CBratAlgorithmGeosVelAtp::Dump (std::ostream & fOut = std::cerr) [override],[virtual]`

Dump function

Reimplemented from **brathl::CBratAlgorithmGeosVel** (p. 133).

References `brathl::CBratAlgorithmGeosVel::Dump()`.

8.17.3.2 `virtual std::string brathl::CBratAlgorithmGeosVelAtp::GetDescription () const [inline],[override],[virtual]`

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

8.17.3.3 `virtual std::string brathl::CBratAlgorithmGeosVelAtp::GetInputParamDesc (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References `brathl::CTools::Format()`, and `GetNumInputParam()`.

8.17.3.4 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal bratl::CBratAlgorithmGeosVelAtp::GetInputParamFormat (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the format of an input parameter : CBratAlgorithmParam::T_DOUBLE for double CBratAlgorithmParam::T_FLOAT for float CBratAlgorithmParam::T_INT for integer CBratAlgorithmParam::T_LONG for long integer CBratAlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **GetNumInputParam()**.

8.17.3.5 `virtual std::string bratl::CBratAlgorithmGeosVelAtp::GetInputParamUnit (uint32_t indexParam) const [inline],[override],[virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 130).

References **bratl::CTools::Format()**, and **GetNumInputParam()**.

8.17.3.6 `virtual std::string bratl::CBratAlgorithmGeosVelAtp::GetName () const [inline],[override],[virtual]`

Gets the name of the algorithm

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.17.3.7 `virtual uint32_t bratl::CBratAlgorithmGeosVelAtp::GetNumInputParam () const [inline],[override],[virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **bratl::CBratAlgorithmBase** (p. 131).

Referenced by **GetInputParamDesc()**, **GetInputParamFormat()**, and **GetInputParamUnit()**.

8.17.3.8 `virtual std::string bratl::CBratAlgorithmGeosVelAtp::GetOutputUnit () const [inline],[override],[virtual]`

Gets the unit of an output value returned by the 'Run' function.

Parameters

<i>indexParam</i>	[in] : parameter index.
-------------------	-------------------------

Implements **bratl::CBratAlgorithmBase** (p. 131).

8.17.3.9 `CBratAlgorithmGeosVelAtp & bratl::CBratAlgorithmGeosVelAtp::operator= (const CBratAlgorithmGeosVelAtp & copy)`

Overloads operator '='

8.17.3.10 `double bratl::CBratAlgorithmGeosVelAtp::Run (CVectorBratAlgorithmParam & args) [override],[virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **bratl::CBratAlgorithmBase** (p. 131).

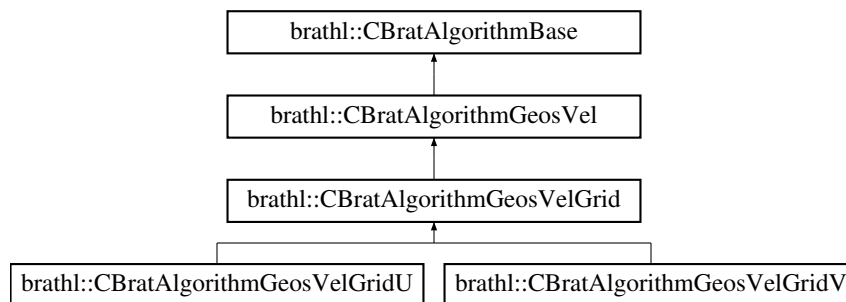
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelAtp.h
- BratAlgorithmGeosVelAtp.cpp

8.18 bratl::CBratAlgorithmGeosVelGrid Class Reference

```
#include <BratAlgorithmGeosVelGrid.h>
```

Inheritance diagram for bratl::CBratAlgorithmGeosVelGrid:



Public Member Functions

- **CBratAlgorithmGeosVelGrid** ()
- **CBratAlgorithmGeosVelGrid** (const **CBratAlgorithmGeosVelGrid** ©)
- virtual void **CheckInputParams** (CVectorBratAlgorithmParam &args) override
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual **CBratAlgorithmParam::bratAlgoParamTypeVal** **GetInputParamFormat** (uint32_t indexParam) const override
- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual double **GetParamDefaultValue** (uint32_t indexParam)
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **CBratAlgorithmGeosVelGrid** & **operator=** (const **CBratAlgorithmGeosVelGrid** ©)
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual void **SetParamValues** (CVectorBratAlgorithmParam &args)
- virtual ~**CBratAlgorithmGeosVelGrid** ()

Protected Member Functions

- void **CheckEquatorLimit** ()
- void **CheckLatLonExpression** (uint32_t index)
- void **CheckProduct** ()
- void **CheckVarExpression** (uint32_t index)
- double **ComputeMean** ()
- double **ComputeSingle** ()
- virtual double **ComputeVelocity** ()=0
- virtual void **DeleteFieldNetCdf** () override
- virtual void **DeleteProduct** () override
- uint32_t **GetLatDimRange** (CFieldNetCdf *field)
- int32_t **GetLatitudeIndex** (double value)
- void **GetLatitudes** ()
- uint32_t **GetLonDimRange** (CFieldNetCdf *field)
- int32_t **GetLongitudeIndex** (double value)
- void **GetLongitudes** ()
- void **GetVarCacheExpressionValue** (int32_t minIndexLat, int32_t maxIndexLat, int32_t minIndexLon, int32_t maxIndexLon)
- double **GetVarExpressionValue** (int32_t indexLat, int32_t indexLon)
- double **GetVarExpressionValueCache** (int32_t indexLat, int32_t indexLon)
- void **Init** ()
- virtual void **OpenProductFile** () override
- bool **PrepareComputeVelocity** ()
- virtual void **PrepareDataReading2D** (int32_t minIndexLat, int32_t maxIndexLat, int32_t minIndexLon, int32_t maxIndexLon)
- virtual void **PrepareDataReading2D** (int32_t indexLat, int32_t indexLon)
- virtual void **PrepareDataValues2DComplexExpression** (CExpressionValue &exprValue)
- virtual void **PrepareDataValues2DComplexExpressionWithAlgo** (CExpressionValue &exprValue)
- virtual void **PrepareDataValues2DOneField** (CExpressionValue &exprValue)
- void **Set** (const CBratAlgorithmGeosVelGrid ©)
- virtual void **SetBeginOfFile** () override
- virtual void **SetEndOfFile** () override

Protected Attributes

- bool **m_allLongitudes**
- double **m_equatorLimit**
- CFieldNetCdf * **m_fieldLat**
- CFieldNetCdf * **m_fieldLon**
- int32_t **m_indexLat**
- int32_t **m_indexLon**
- CDoubleArray **m_latitudes**
- CDoubleArray **m_longitudes**
- double **m_lonMax**
- double **m_lonMin**
- CExpressionValue **m_rawDataCache**
- int32_t **m_varDimLatIndex**
- int32_t **m_varDimLonIndex**
- double **m_varValue**
- double **m_varValueE**
- double **m_varValueN**
- double **m_varValueS**
- double **m_varValueW**

Static Protected Attributes

- static const uint32_t **m_EQUATOR_LAT_LIMIT_INDEX** = 3
- static const uint32_t **m_INPUT_PARAMS** = 4
- static const uint32_t **m_LAT_PARAM_INDEX** = 0
- static const uint32_t **m_LON_PARAM_INDEX** = 1
- static const uint32_t **m_VAR_PARAM_INDEX** = 2

Additional Inherited Members

8.18.1 Detailed Description

Algorithm base class.

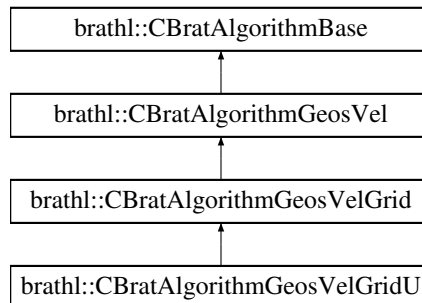
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.19 bratl::CBratAlgorithmGeosVelGridU Class Reference

```
#include <BratAlgorithmGeosVelGrid.h>
```

Inheritance diagram for bratl::CBratAlgorithmGeosVelGridU:



Public Member Functions

- **CBratAlgorithmGeosVelGridU** ()
- **CBratAlgorithmGeosVelGridU** (const **CBratAlgorithmGeosVelGridU** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- virtual ~**CBratAlgorithmGeosVelGridU** ()

Protected Member Functions

- double **ComputeVelocity** () override
- void **Init** ()

Additional Inherited Members

8.19.1 Detailed Description

Algorithm base class.

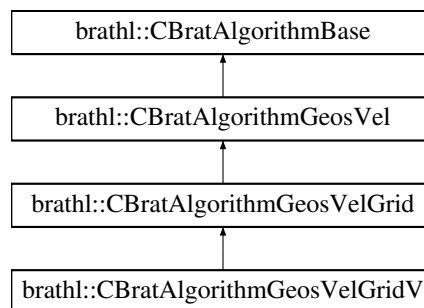
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.20 bratl::CBratAlgorithmGeosVelGridV Class Reference

```
#include <BratAlgorithmGeosVelGrid.h>
```

Inheritance diagram for bratl::CBratAlgorithmGeosVelGridV:



Public Member Functions

- **CBratAlgorithmGeosVelGridV** ()
- **CBratAlgorithmGeosVelGridV** (const **CBratAlgorithmGeosVelGridV** ©)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
- virtual std::string **GetDescription** () const override
- virtual std::string **GetName** () const override
- virtual ~**CBratAlgorithmGeosVelGridV** ()

Protected Member Functions

- double **ComputeVelocity** () override
- void **Init** ()

Additional Inherited Members

8.20.1 Detailed Description

Algorithm base class.

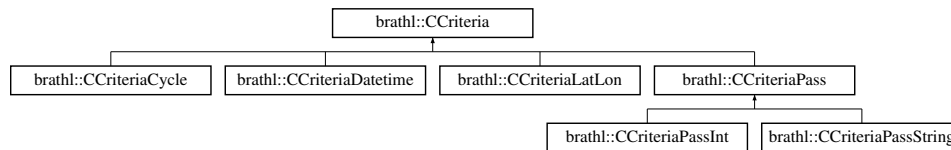
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.21 bratl::CCriteria Class Reference

```
#include <Criteria.h>
```

Inheritance diagram for bratl::CCriteria:



Public Types

- enum **CriteriaKind** {
 UNKNOWN, **LATLON**, **DATETIME**, **PASS**,
 CYCLE }

Public Member Functions

- **CCriteria** ()
 Empty **CCriteria** (p. 141) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
 Dump function.
- virtual std::string **GetAsText** (const std::string &delimiter)=0
- int32_t **GetKey** ()
- virtual bool **IsDefaultValue** ()=0
- virtual void **SetDefaultValue** ()=0
- virtual ~**CCriteria** ()
 Destructor.

Static Public Member Functions

- static void **Adjust** (CIntArray &array)
- static **CCriteria** * **GetCriteria** (CBratObject *ob, bool withExcept=true)

Protected Attributes

- int32_t **m_key**

8.21.1 Detailed Description

Criteria management class.

Version

1.0

8.21.2 Member Function Documentation

8.21.2.1 virtual bool bratl::CCriteria::IsDefaultValue () [pure virtual]

Tests whether value have been initialized or not

Returns

true if not initialized

Implemented in **bratl::CCriteriaPassInt** (p. 61), **bratl::CCriteriaLatLon** (p. 158), **bratl::CCriteriaDatetime** (p. 150), **bratl::CCriteriaCycle** (p. 145), **bratl::CCriteriaPassString** (p. 61), and **bratl::CCriteriaPass** (p. 60).

8.21.2.2 virtual void bratl::CCriteria::SetDefaultValue () [pure virtual]

Sets internal value to the default value (uninitialized)

Implemented in **bratl::CCriteriaPassInt** (p. 62), **bratl::CCriteriaLatLon** (p. 160), **bratl::CCriteriaDatetime** (p. 151), **bratl::CCriteriaCycle** (p. 146), **bratl::CCriteriaPassString** (p. 62), and **bratl::CCriteriaPass** (p. 62).

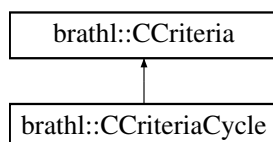
The documentation for this class was generated from the following files:

- Criteria.h
- Criteria.cpp

8.22 bratl::CCriteriaCycle Class Reference

```
#include <CriteriaCycle.h>
```

Inheritance diagram for bratl::CCriteriaCycle:



Public Member Functions

- **CCriteriaCycle** ()
Empty CCriteriaCycle (p. 142) ctor.
- **CCriteriaCycle** (CCriteriaCycle &c)
- **CCriteriaCycle** (CCriteriaCycle *c)
- **CCriteriaCycle** (int32_t from, int32_t to)
- **CCriteriaCycle** (const std::string &from, const std::string &to)
- **CCriteriaCycle** (const CStringArray &array)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- std::string **GetAsText** (const std::string &delimiter=CCriteriaCycle::m_delimiter)
- int32_t **GetFrom** ()
- int32_t **GetTo** ()
- bool **Intersect** (CStringArray &array, CStringArray &intersect)
- bool **Intersect** (CStringArray &array, CIntArray &intersect)
- bool **Intersect** (CIntArray &array, CStringArray &intersect)
- bool **Intersect** (CIntArray &array, CIntArray &intersect)
- bool **Intersect** (int32_t from, int32_t to, CStringArray &intersect)
- bool **Intersect** (int32_t from, int32_t to, CIntArray &intersect)
- bool **Intersect** (const std::string &from, const std::string &to, CIntArray &intersect)
- bool **Intersect** (double otherFrom, double otherTo, CIntArray &intersect)
- bool **Intersect** (const std::string &from, const std::string &to, CStringArray &intersect)
- bool **IsDefaultValue** ()
- const **CCriteriaCycle** & **operator=** (CCriteriaCycle &c)

- void **Set** (**CCriteriaCycle** &c)
- void **Set** (int32_t from, int32_t to)
- void **Set** (const std::string &from, const std::string &to)
- void **Set** (const CStringArray &array)
- void **SetDefaultValue** ()
- void **SetFrom** (int32_t from)
- void **SetFrom** (const std::string &from)
- void **SetFromText** (const std::string &values, const std::string &delimiter=**CCriteriaCycle**::m_delimiter)
- void **SetTo** (int32_t to)
- void **SetTo** (const std::string &to)
- virtual ~**CCriteriaCycle** ()

Destructor.

Static Public Member Functions

- static **CCriteriaCycle** * **GetCriteria** (CBratObject *ob, bool withExcept=true)

Static Public Attributes

- static const std::string **m_delimiter** = " "

Protected Member Functions

- void **Adjust** ()
- void **Init** ()

Protected Attributes

- int32_t **m_from**
- int32_t **m_to**

Additional Inherited Members

8.22.1 Detailed Description

Pass number (from/to) Criteria management class.

Version

1.0

8.22.2 Constructor & Destructor Documentation

8.22.2.1 brathl::CCriteriaCycle::CCriteriaCycle (int32_t from, int32_t to)

Constructor.

Parameters

<i>from</i>	start pass
-------------	------------

<i>to</i>	end pass
-----------	----------

8.22.2.2 bratl::CCriteriaCycle::CCriteriaCycle (const std::string & *from*, const std::string & *to*)

Constructor.

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

8.22.2.3 bratl::CCriteriaCycle::CCriteriaCycle (const CStringArray & *array*)

Constructor from a array that contains start pass as std::string, end pass as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

8.22.3 Member Function Documentation

8.22.3.1 bool bratl::CCriteriaCycle::Intersect (CStringArray & *array*, CStringArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

Referenced by Intersect().

8.22.3.2 bool bratl::CCriteriaCycle::Intersect (CStringArray & *array*, CIntArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References Intersect().

8.22.3.3 bool bratl::CCriteriaCycle::Intersect (CIntArray & *array*, CStringArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References Intersect().

8.22.3.4 bool brathl::CCriteriaCycle::Intersect (CIntArray & *array*, CIntArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>array</i>	that contains start pass as std::string, end pass as std::string
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References Intersect().

8.22.3.5 bool brathl::CCriteriaCycle::IsDefaultValue () [virtual]

Tests whether the pass have been initialized or not

Returns

true if not initialized

Implements **brathl::CCriteria** (p. 141).

References m_from, and m_to.

8.22.3.6 void brathl::CCriteriaCycle::Set (int32_t *from*, int32_t *to*)

Sets date period from start and end pass

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

References SetFrom(), and SetTo().

8.22.3.7 void brathl::CCriteriaCycle::Set (const std::string & *from*, const std::string & *to*)

Sets date period from start and end pass

Parameters

<i>from</i>	start pass
<i>to</i>	end pass

References brathl::CTools::StrToInt32().

8.22.3.8 void brathl::CCriteriaCycle::Set (const CStringArray & *array*)

Sets a date period from a array that contains start pass as std::string, end pass as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

8.22.3.9 void brathl::CCriteriaCycle::SetDefaultValue () [virtual]

Sets internal value to the default value (uninitialized)

Implements **brathl::CCriteria** (p. 142).

References m_from, and m_to.

8.22.3.10 void brathl::CCriteriaCycle::SetFrom (int32_t from)

Sets start pass

Parameters

<i>to</i>	start pass
-----------	------------

References m_from.

Referenced by Set().

8.22.3.11 void brathl::CCriteriaCycle::SetFrom (const std::string & from)

Sets start pass

Parameters

<i>to</i>	start pass
-----------	------------

References m_from, and brathl::CTools::StrToInt32().

8.22.3.12 void brathl::CCriteriaCycle::SetTo (int32_t to)

Sets end pass

Parameters

<i>to</i>	end pass
-----------	----------

References m_to.

Referenced by Set().

8.22.3.13 void brathl::CCriteriaCycle::SetTo (const std::string & to)

Sets end pass

Parameters

<i>to</i>	end pass
-----------	----------

References m_to, and brathl::CTools::StrToInt32().

8.22.4 Member Data Documentation

8.22.4.1 int32_t brathl::CCriteriaCycle::m_from [protected]

start pass

Referenced by Dump(), IsDefaultValue(), SetDefaultValue(), and SetFrom().

8.22.4.2 `int32_t bratl::CCriteriaCycle::m_to` [protected]

end pass

Referenced by `Dump()`, `IsDefaultValue()`, `SetDefaultValue()`, and `SetTo()`.

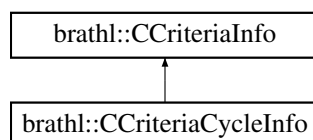
The documentation for this class was generated from the following files:

- `CriteriaCycle.h`
- `CriteriaCycle.cpp`

8.23 `bratl::CCriteriaCycleInfo` Class Reference

`#include <CriteriaInfo.h>`

Inheritance diagram for `bratl::CCriteriaCycleInfo`:



Public Member Functions

- **`CCriteriaCycleInfo ()`**
*Empty **`CCriteriaCycleInfo`** (p. 147) ctor.*
- virtual void **`Dump`** (`std::ostream &fOut=std::cerr`)
Dump function.
- `CFieldInfo *` **`GetEndCycleField ()`**
- `const std::string &` **`GetEndCycleFieldName ()`**
- virtual void **`GetFieldsInfo (CObMap *fieldsInfo)`**
- `CFieldInfo *` **`GetStartCycleField ()`**
- `const std::string` **`GetStartCycleFieldName ()`**
- void **`SetEndCycleField`** (`const std::string &value`)
- void **`SetEndCycleField`** (`CFieldInfo &value`)
- void **`SetStartCycleField`** (`const std::string &value`)
- void **`SetStartCycleField`** (`CFieldInfo &value`)
- virtual `~CCriteriaCycleInfo ()`
Destructor.

Static Public Member Functions

- static `CCriteriaCycleInfo *` **`GetCriteriaInfo (CBratObject *ob, bool withExcept=true)`**

Protected Attributes

- `CFieldInfo` **`m_endCycleField`**
- `CFieldInfo` **`m_startCycleField`**

8.23.1 Detailed Description

Cycle criteria information management class.

Version

1.0

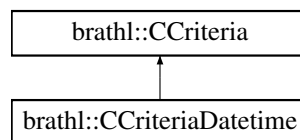
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.24 brathl::CCriteriaDatetime Class Reference

```
#include <CriteriaDatetime.h>
```

Inheritance diagram for brathl::CCriteriaDatetime:



Public Member Functions

- **CCriteriaDatetime** ()
Empty CCriteriaDatetime (p. 148) ctor.
- **CCriteriaDatetime** (CCriteriaDatetime &c)
- **CCriteriaDatetime** (CCriteriaDatetime *c)
- **CCriteriaDatetime** (CDatePeriod &datePeriod)
- **CCriteriaDatetime** (CDate &from, CDate &to)
- **CCriteriaDatetime** (const std::string &from, const std::string &to)
- **CCriteriaDatetime** (double from, double to)
- **CCriteriaDatetime** (const CStringArray &array)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- std::string **GetAsText** (const std::string &delimiter=CDatePeriod::m_delimiter)
- CDatePeriod * **GetDatePeriod** ()
- CDate * **GetFrom** ()
- std::string **GetFromAsText** ()
- CDate * **GetTo** ()
- std::string **GetToAsText** ()
- bool **Intersect** (CDatePeriod &datePeriod, CDatePeriod &intersect)
- bool **Intersect** (double otherFrom, double otherTo, CDatePeriod &intersect)
- bool **Intersect** (double otherFrom, double otherTo)
- bool **IsDefaultValue** ()
- const CCriteriaDatetime & **operator=** (CCriteriaDatetime &c)
- void **Set** (CDatePeriod &datePeriod)
- void **Set** (CDate &from, CDate &to)
- void **Set** (const std::string &from, const std::string &to)
- void **Set** (double from, double to)
- void **Set** (const CStringArray &array)

- void **Set** (**CCriteriaDatetime** &c)
- void **SetDefaultValue** ()
- void **SetFrom** (**CDate** &from)
- void **SetFrom** (const std::string &strDate)
- void **SetFromText** (const std::string &values, const std::string &delimiter=CCriteriaDatetime::m_delimiter)
- void **SetTo** (**CDate** &to)
- void **SetTo** (const std::string &strDate)
- virtual ~**CCriteriaDatetime** ()

Destructor.

Static Public Member Functions

- static **CCriteriaDatetime** * **GetCriteria** (CBratObject *ob, bool withExcept=true)

Protected Member Functions

- void **Init** ()

Protected Attributes

- **CDatePeriod** m_datePeriod

Additional Inherited Members

8.24.1 Detailed Description

Datetime Criteria management class.

Version

1.0

8.24.2 Constructor & Destructor Documentation

8.24.2.1 brathl::CCriteriaDatetime::CCriteriaDatetime (CDatePeriod & datePeriod)

Constructor.

Parameters

<i>datePeriod</i>	period to set
-------------------	---------------

References Set().

8.24.2.2 brathl::CCriteriaDatetime::CCriteriaDatetime (CDate & from, CDate & to)

Constructor.

Parameters

<i>from</i>	start date
<i>to</i>	end date

References Set().

8.24.2.3 brathl::CCriteriaDatetime::CCriteriaDatetime (const std::string & from, const std::string & to)

Constructor.

Parameters

<i>from</i>	start date
<i>to</i>	end date

References Set().

8.24.2.4 brathl::CCriteriaDatetime::CCriteriaDatetime (double *from*, double *to*)

Constructor.

Parameters

<i>from</i>	start date (number of seconds since 1950-01-01)
<i>to</i>	end date (number of seconds since 1950-01-01)

References Set().

8.24.2.5 brathl::CCriteriaDatetime::CCriteriaDatetime (const CStringArray & *array*)

Constructor from a array that contains start date as std::string, end date as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

References Set().

8.24.3 Member Function Documentation

8.24.3.1 bool brathl::CCriteriaDatetime::Intersect (CDatePeriod & *datePeriod*, CDatePeriod & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>datePeriod</i>	intersect with this
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References brathl::CDatePeriod::Intersect(), and m_datePeriod.

8.24.3.2 bool brathl::CCriteriaDatetime::Intersect (double *otherFrom*, double *otherTo*, CDatePeriod & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>otherFrom</i>	start date intersect with this
<i>otherTo</i>	end date intersect with this
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References brathl::CDatePeriod::Intersect(), and m_datePeriod.

8.24.3.3 bool brathl::CCriteriaDatetime::IsDefaultValue () [virtual]

Tests whether date period have been initialized or not

Returns

true if not initialized

Implements **bratl::CCriteria** (p. 141).

References bratl::CDatePeriod::IsDefaultValue(), and m_datePeriod.

8.24.3.4 void bratl::CCriteriaDatetime::Set (CDatePeriod & datePeriod)

Sets date period from another one

Parameters

<i>datePeriod</i>	period to set
-------------------	---------------

References m_datePeriod, and bratl::CDatePeriod::Set().

Referenced by CCriteriaDatetime().

8.24.3.5 void bratl::CCriteriaDatetime::Set (CDate & from, CDate & to)

Sets date period from start and end date

Parameters

<i>from</i>	start date
<i>to</i>	end date

References m_datePeriod, and bratl::CDatePeriod::Set().

8.24.3.6 void bratl::CCriteriaDatetime::Set (const std::string & from, const std::string & to)

Sets date period from start and end date

Parameters

<i>from</i>	start date
<i>to</i>	end date

References m_datePeriod, and bratl::CDatePeriod::Set().

8.24.3.7 void bratl::CCriteriaDatetime::Set (const CStringArray & array)

Sets a date period from a array that contains start date as std::string, end date as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

References m_datePeriod, and bratl::CDatePeriod::Set().

8.24.3.8 void bratl::CCriteriaDatetime::SetDefaultValue () [virtual]

Sets internal value to the default value (uninitialized)

Implements **bratl::CCriteria** (p. 142).

References m_datePeriod, and bratl::CDatePeriod::SetDefaultValue().

8.24.3.9 void bratl::CCriteriaDatetime::SetFrom (CDate & from)

Sets start date

Parameters

<i>to</i>	start date
-----------	------------

References `m_datePeriod`, and `brathl::CDatePeriod::SetFrom()`.

8.24.3.10 void brathl::CCriteriaDatetime::SetFrom (const std::string & *strDate*)

Sets start date

Parameters

<i>to</i>	start date
-----------	------------

References `m_datePeriod`, and `brathl::CDatePeriod::SetFrom()`.

8.24.3.11 void brathl::CCriteriaDatetime::SetTo (CDate & *to*)

Sets end date

Parameters

<i>to</i>	end date
-----------	----------

References `m_datePeriod`, and `brathl::CDatePeriod::SetTo()`.

8.24.3.12 void brathl::CCriteriaDatetime::SetTo (const std::string & *strDate*)

Sets end date

Parameters

<i>to</i>	end date
-----------	----------

References `m_datePeriod`, and `brathl::CDatePeriod::SetTo()`.

8.24.4 Member Data Documentation

8.24.4.1 CDatePeriod brathl::CCriteriaDatetime::m_datePeriod [protected]

Date period

Referenced by `Dump()`, `Intersect()`, `IsDefaultValue()`, `Set()`, `SetDefaultValue()`, `SetFrom()`, and `SetTo()`.

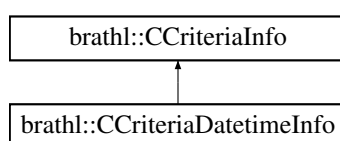
The documentation for this class was generated from the following files:

- `CriteriaDatetime.h`
- `CriteriaDatetime.cpp`

8.25 brathl::CCriteriaDatetimeInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for `brathl::CCriteriaDatetimeInfo`:



Public Member Functions

- **CCriteriaDatetimedInfo** ()
Empty CCriteriaDatetimedInfo (p. 152) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- CFieldInfo * **GetEndDateField** ()
- const std::string & **GetEndDateFieldName** ()
- virtual void **GetFieldsInfo** (CObMap *fieldsInfo)
- brathl_refDate **GetRefDate** ()
- CFieldInfo * **GetStartDateField** ()
- const std::string & **GetStartDateFieldName** ()
- void **SetEndDateField** (const std::string &value)
- void **SetEndDateField** (CFieldInfo &value)
- void **SetRefDate** (brathl_refDate value)
- void **SetStartDateField** (const std::string &value)
- void **SetStartDateField** (CFieldInfo &value)
- virtual ~**CCriteriaDatetimedInfo** ()
Destructor.

Static Public Member Functions

- static **CCriteriaDatetimedInfo** * **GetCriterialInfo** (CBratObject *ob, bool withExcept=true)

Protected Attributes

- CFieldInfo **m_endDateField**
- brathl_refDate **m_refDate**
- CFieldInfo **m_startDateField**

8.25.1 Detailed Description

Date/Time criteria information management class.

Version

1.0

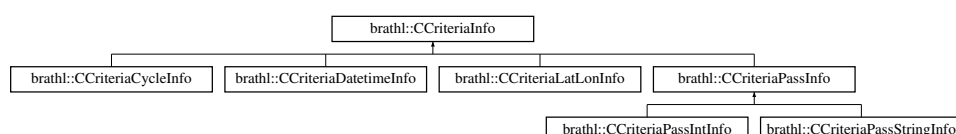
The documentation for this class was generated from the following files:

- CriterialInfo.h
- CriterialInfo.cpp

8.26 brathl::CCriterialInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for brathl::CCriterialInfo:



Public Member Functions

- **CCriterialInfo** ()
*Empty **CCriterialInfo** (p. 153) ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- std::string **GetDataRecord** ()
- virtual void **GetFieldNames** (CStringList &fieldNames)
- virtual void **GetFieldNames** (CStringArray &fieldNames)
- virtual void **GetFields** (CRecordDataMap &listRecord)
- virtual void **GetFieldsInfo** (CObMap *fieldsInfo)=0
- int32_t **GetKey** ()
- void **SetDataRecord** (const std::string &value)
- virtual ~**CCriterialInfo** ()
Destructor.

Static Public Member Functions

- static **CCriterialInfo** * **GetCriterialInfo** (CBratObject *ob, bool withExcept=true)

Protected Attributes

- std::string **m_dataRecord**
- int32_t **m_key**

8.26.1 Detailed Description

Base class for criteria information.

Version

1.0

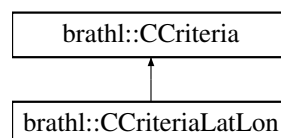
The documentation for this class was generated from the following files:

- CriterialInfo.h
- CriterialInfo.cpp

8.27 bratl::CCriteriaLatLon Class Reference

```
#include <CriteriaLatLon.h>
```

Inheritance diagram for bratl::CCriteriaLatLon:



Public Member Functions

- **CCriteriaLatLon** ()
Empty CCriteriaLatLon (p. 154) ctor.
- **CCriteriaLatLon** (CCriteriaLatLon &c)
- **CCriteriaLatLon** (CCriteriaLatLon *c)
- **CCriteriaLatLon** (CLatLonRect &latLonRect)
- **CCriteriaLatLon** (CLatLonPoint &p1, double deltaLat, double deltaLon)
- **CCriteriaLatLon** (CLatLonPoint &latLonLow, CLatLonPoint &latLonHigh)
- **CCriteriaLatLon** (double latLow, double lonLow, double latHigh, double lonHigh)
- **CCriteriaLatLon** (const std::string &latLow, const std::string &lonLow, const std::string &latHigh, const std::string &lonHigh)
- **CCriteriaLatLon** (const CStringArray &array)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual std::string **GetAsText** (const std::string &delimiter=CLatLonRect::m_delimiter)
- CLatLonRect * **GetLatLonRect** ()
- double **GetLowerLeftLat** ()
- double **GetLowerLeftLon** ()
- double **GetLowerRightLat** ()
- double **GetLowerRightLon** ()
- double **GetUpperLeftLat** ()
- double **GetUpperLeftLon** ()
- double **GetUpperRightLat** ()
- double **GetUpperRightLon** ()
- bool **Intersect** (CLatLonRect &clip, CLatLonRect &intersect)
- bool **IsDefaultValue** ()
- const **CCriteriaLatLon** & **operator=** (CCriteriaLatLon &c)
- void **Set** (CLatLonRect &latLonRect)
- void **Set** (CLatLonPoint &p1, double deltaLat, double deltaLon)
- void **Set** (CLatLonPoint &latLonLow, CLatLonPoint &latLonHigh)
- void **Set** (double latLow, double lonLow, double latHigh, double lonHigh)
- void **Set** (const std::string &latLow, const std::string &lonLow, const std::string &latHigh, const std::string &lonHigh)
- void **Set** (const std::string &latLonRect, const std::string &delimiter=CLatLonRect::m_delimiter)
- void **Set** (CCriteriaLatLon &c)
- void **SetDefaultValue** ()
- virtual ~**CCriteriaLatLon** ()
Destructor.

Static Public Member Functions

- static **CCriteriaLatLon** * **GetCriteria** (CBratObject *ob, bool withExcept=true)
- static double **GetMinOrMaxLon** (double lon1, double lon2, bool wantMin)

Protected Member Functions

- void **Init** ()

Protected Attributes

- CLatLonRect **m_latLonRect**

Additional Inherited Members

8.27.1 Detailed Description

Latitude/Longitude Criteria management class.

Version

1.0

8.27.2 Constructor & Destructor Documentation

8.27.2.1 bratl::CCriteriaLatLon::CCriteriaLatLon (CLatLonRect & *latLonRect*)

Constructor.

Parameters

<i>latLonRect</i>	lat/lon bounding box
-------------------	----------------------

References Set().

8.27.2.2 bratl::CCriteriaLatLon::CCriteriaLatLon (CLatLonPoint & *p1*, double *deltaLat*, double *deltaLon*)

Construct a lat/lon bounding box from a point, and a delta lat, lon. This disambiguates which way the box wraps around the globe.

Parameters

<i>p1</i>	one corner of the box
<i>deltaLat</i>	delta lat from p1. (may be positive or negative)
<i>deltaLon</i>	delta lon from p1. (may be positive or negative)

References Set().

8.27.2.3 bratl::CCriteriaLatLon::CCriteriaLatLon (CLatLonPoint & *latLonLow*, CLatLonPoint & *latLonHigh*)

Constructor.

Parameters

<i>latLonLow</i>	lat/lon low point
<i>latLonHigh</i>	lat/lon high point

References Set().

8.27.2.4 bratl::CCriteriaLatLon::CCriteriaLatLon (double *latLow*, double *lonLow*, double *latHigh*, double *lonHigh*)

Constructor.

Parameters

<i>latLow</i>	latitude low
<i>lonLow</i>	longitude low
<i>latHigh</i>	latitude high
<i>lonHigh</i>	longitude high

References Set().

8.27.2.5 bratl::CCriteriaLatLon::CCriteriaLatLon (const std::string & *latLow*, const std::string & *lonLow*, const std::string & *latHigh*, const std::string & *lonHigh*)

Constructor.

Parameters

<i>latLow</i>	latitude low
<i>lonLow</i>	longitude low
<i>latHigh</i>	latitude high
<i>lonHigh</i>	longitude high

References Set().

8.27.2.6 bratl::CCriteriaLatLon::CCriteriaLatLon (const CStringArray & array)

Constructor from a list that contains low latitude value, low longitude value, high latitude value, high longitude value.

Parameters

<i>array</i>	to be converted
--------------	-----------------

References m_latLonRect.

8.27.2.7 bratl::CCriteriaLatLon::~~CCriteriaLatLon () [virtual]

Destructor.

Getter of the property `latLonRect`.

Returns

Returns the latLonRect.

8.27.3 Member Function Documentation

8.27.3.1 double bratl::CCriteriaLatLon::GetLowerLeftLat () [inline]

Returns

lower left latitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.2 double bratl::CCriteriaLatLon::GetLowerLeftLon () [inline]

Returns

lower left longitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.3 double bratl::CCriteriaLatLon::GetLowerRightLat () [inline]

Returns

lower right latitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.4 double bratl::CCriteriaLatLon::GetLowerRightLon () [inline]

Returns

lower right longitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.5 double bratl::CCriteriaLatLon::GetMinOrMaxLon (double lon1, double lon2, bool wantMin) [static]

Gets the min. or max. of two longitudes.

Parameters

<i>lon1</i>	first longitude
<i>lon2</i>	second longitude
<i>wantMin</i>	true: returns min., false: returns max.

Returns

min. lon or max. lon, depends on wantMin.

References bratl::CTools::Max(), and bratl::CTools::Min().

8.27.3.6 double bratl::CCriteriaLatLon::GetUpperLeftLat () [inline]

Returns

upper left latitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.7 double bratl::CCriteriaLatLon::GetUpperLeftLon () [inline]

Returns

upper left longitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.8 double bratl::CCriteriaLatLon::GetUpperRightLat () [inline]

Returns

upper right latitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.9 double bratl::CCriteriaLatLon::GetUpperRightLon () [inline]

Returns

upper right longitude of the lat/lon box, Double.MAX_VALUE if not set.

References m_latLonRect.

8.27.3.10 bool bratl::CCriteriaLatLon::Intersect (CLatLonRect & clip, CLatLonRect & intersect)

Create the intersection of this LatLon Criteria with the given one

Parameters

<i>clip</i>	intersect with this
<i>intersection</i>	

Returns

true, or false if there is no intersection

References m_latLonRect.

8.27.3.11 bool bratl::CCriteriaLatLon::IsDefaultValue () [virtual]

Tests whether date period have been initialized or not

Returns

true if not initialized

Implements **brathl::CCriteria** (p. 141).

References `m_latLonRect`.

8.27.3.12 **void brathl::CCriteriaLatLon::Set (CLatLonRect & *latLonRect*)**

Setter of the property `&latLonRect`;

Parameters

<i>latLonRect</i>	The <code>latLonRect</code> to set.
-------------------	-------------------------------------

References `m_latLonRect`.

Referenced by `CCriteriaLatLon()`.

8.27.3.13 **void brathl::CCriteriaLatLon::Set (CLatLonPoint & *p1*, double *deltaLat*, double *deltaLon*)**

Set a lat/lon bounding box from a point, and a delta lat, lon. This disambiguates which way the box wraps around the globe.

Parameters

<i>p1</i>	one corner of the box
<i>deltaLat</i>	delta lat from <i>p1</i> . (may be positive or negative)
<i>deltaLon</i>	delta lon from <i>p1</i> . (may be positive or negative)

References `m_latLonRect`.

8.27.3.14 **void brathl::CCriteriaLatLon::Set (CLatLonPoint & *latLonLow*, CLatLonPoint & *latLonHigh*)**

Setter of the property `&latLonRect`;

Parameters

<i>latLonLow</i>	lat/lon low point
<i>latLonHigh</i>	lat/lon high point .property name="latLonRect"

References `m_latLonRect`.

8.27.3.15 **void brathl::CCriteriaLatLon::Set (double *latLow*, double *lonLow*, double *latHigh*, double *lonHigh*)**

Setter of the property `&latLonRect`;

Parameters

<i>latLow</i>	latitude low
<i>lonLow</i>	longitude low
<i>latHigh</i>	latitude high
<i>lonHigh</i>	longitude high

References `m_latLonRect`.

8.27.3.16 **void brathl::CCriteriaLatLon::Set (const std::string & *latLow*, const std::string & *lonLow*, const std::string & *latHigh*, const std::string & *lonHigh*)**

Setter of the property `&latLonRect`;

Parameters

<i>latLow</i>	latitude low
<i>lonLow</i>	longitude low
<i>latHigh</i>	latitude high
<i>lonHigh</i>	longitude high

References `m_latLonRect`.

8.27.3.17 `void brathl::CCriteriaLatLon::Set (const std::string & latLonRect, const std::string & delimiter = CLatLonRect::m_delimiter)`

Setter of the property `t<tt>latLonRect</tt>`

Parameters

<i>latLonRect</i>	latitude low, longitude low, latitude high, longitude high
-------------------	--

References `m_latLonRect`.

8.27.3.18 `void brathl::CCriteriaLatLon::SetDefaultValue ()` `[virtual]`

Sets internal value to the default value (uninitialized)

Implements **brathl::CCriteria** (p. 142).

References `m_latLonRect`.

8.27.4 Member Data Documentation

8.27.4.1 `CLatLonRect brathl::CCriteriaLatLon::m_latLonRect` `[protected]`

Bounding box for latitude/longitude points. This is a rectangle in lat/lon coordinates. Note that `LatLonPoint` always has lon in the range ± 180 . *

Referenced by `CCriteriaLatLon()`, `Dump()`, `GetLowerLeftLat()`, `GetLowerLeftLon()`, `GetLowerRightLat()`, `GetLowerRightLon()`, `GetUpperLeftLat()`, `GetUpperLeftLon()`, `GetUpperRightLat()`, `GetUpperRightLon()`, `Intersect()`, `IsDefaultValue()`, `Set()`, and `SetDefaultValue()`.

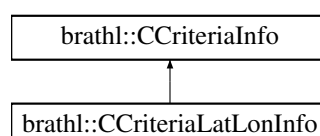
The documentation for this class was generated from the following files:

- `CriteriaLatLon.h`
- `CriteriaLatLon.cpp`

8.28 brathl::CCriteriaLatLonInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for `brathl::CCriteriaLatLonInfo`:



Public Member Functions

- **CCriteriaLatLonInfo** ()
Empty **CCriteriaLatLonInfo** (p. 160) ctor.

- virtual void **Dump** (std::ostream &fOut=std::cerr)

Dump fonction.

- CFieldInfo * **GetEndLatField** ()
- const std::string & **GetEndLatFieldName** ()
- CFieldInfo * **GetEndLonField** ()
- const std::string & **GetEndLonFieldName** ()
- virtual void **GetFieldsInfo** (CObMap *fieldsInfo)
- CFieldInfo * **GetStartLatField** ()
- const std::string & **GetStartLatFieldName** ()
- CFieldInfo * **GetStartLonField** ()
- const std::string & **GetStartLonFieldName** ()
- void **SetEndLatField** (const std::string &value)
- void **SetEndLatField** (CFieldInfo &value)
- void **SetEndLonField** (const std::string &value)
- void **SetEndLonField** (CFieldInfo &value)
- void **SetStartLatField** (const std::string &value)
- void **SetStartLatField** (CFieldInfo &value)
- void **SetStartLonField** (const std::string &value)
- void **SetStartLonField** (CFieldInfo &value)
- virtual ~**CCriteriaLatLonInfo** ()

Destructor.

Static Public Member Functions

- static CCriteriaLatLonInfo * **GetCriteriaInfo** (CBratObject *ob, bool withExcept=true)

Protected Attributes

- CFieldInfo **m_endLatField**
- CFieldInfo **m_endLonField**
- CFieldInfo **m_startLatField**
- CFieldInfo **m_startLonField**

8.28.1 Detailed Description

Lat/Lon criteria information management class.

Version

1.0

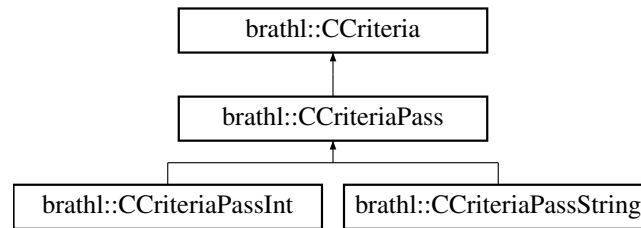
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.29 brathl::CCriteriaPass Class Reference

```
#include <CriteriaPass.h>
```

Inheritance diagram for brathl::CCriteriaPass:



Public Member Functions

- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- virtual bool **IsDefaultValue** ()=0
- virtual void **SetDefaultValue** ()=0
- virtual ~**CCriteriaPass** ()
Destructor.

Static Public Member Functions

- static **CCriteriaPass** * **GetCriteria** (CBratObject *ob, bool withExcept=true)

Protected Member Functions

- **CCriteriaPass** ()
Empty CCriteriaPass (p. 162) ctor.
- void **Init** ()

Additional Inherited Members

8.29.1 Detailed Description

Pass number Criteria management class.

Version

1.0

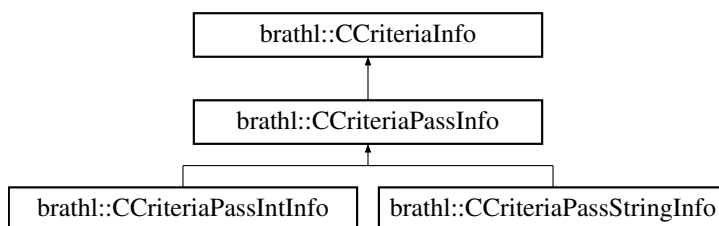
The documentation for this class was generated from the following files:

- CriteriaPass.h
- CriteriaPass.cpp

8.30 brathl::CCriteriaPassInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for brathl::CCriteriaPassInfo:



Public Member Functions

- **CCriteriaPassInfo ()**
Empty CCriteriaPassInfo (p. 162) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- CFieldInfo * **GetEndPassField** ()
- const std::string & **GetEndPassFieldName** ()
- virtual void **GetFieldsInfo** (CObMap *fieldsInfo)
- CFieldInfo * **GetStartPassField** ()
- const std::string & **GetStartPassFieldName** ()
- void **SetEndPassField** (const std::string &value)
- void **SetEndPassField** (CFieldInfo &value)
- void **SetStartPassField** (const std::string &value)
- void **SetStartPassField** (CFieldInfo &value)
- virtual ~**CCriteriaPassInfo** ()
Destructor.

Static Public Member Functions

- static **CCriteriaPassInfo * GetCriteriaInfo** (CBratObject *ob, bool withExcept=true)

Protected Attributes

- CFieldInfo **m_endPassField**
- CFieldInfo **m_startPassField**

8.30.1 Detailed Description

Pass criteria information management class.

Version

1.0

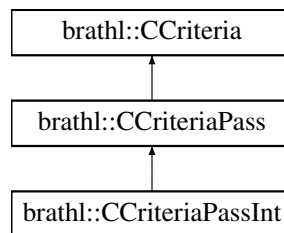
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.31 brathl::CCriteriaPassInt Class Reference

```
#include <CriteriaPass.h>
```

Inheritance diagram for brathl::CCriteriaPassInt:



Public Member Functions

- **CCriteriaPassInt** ()
Empty CCriteriaPassInt (p. 164) ctor.
- **CCriteriaPassInt** (CCriteriaPassInt &c)
- **CCriteriaPassInt** (CCriteriaPassInt *c)
- **CCriteriaPassInt** (int32_t from, int32_t to)
- **CCriteriaPassInt** (const std::string &from, const std::string &to)
- **CCriteriaPassInt** (const CStringArray &array)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- std::string **GetAsText** (const std::string &delimiter=CCriteriaPassInt::m_delimiter)
- int32_t **GetFrom** ()
- int32_t **GetTo** ()
- bool **Intersect** (CStringArray &array, CStringArray &intersect)
- bool **Intersect** (CStringArray &array, CIntArray &intersect)
- bool **Intersect** (CIntArray &array, CStringArray &intersect)
- bool **Intersect** (CIntArray &array, CIntArray &intersect)
- bool **Intersect** (int32_t from, int32_t to, CStringArray &intersect)
- bool **Intersect** (int32_t from, int32_t to, CIntArray &intersect)
- bool **Intersect** (double otherFrom, double otherTo, CIntArray &intersect)
- bool **Intersect** (const std::string &from, const std::string &to, CIntArray &intersect)
- bool **Intersect** (const std::string &from, const std::string &to, CStringArray &intersect)
- bool **IsDefaultValue** ()
- const **CCriteriaPassInt** & **operator=** (CCriteriaPassInt &c)
- void **Set** (CCriteriaPassInt &c)
- void **Set** (int32_t from, int32_t to)
- void **Set** (const std::string &from, const std::string &to)
- void **Set** (const CStringArray &array)
- void **SetDefaultValue** ()
- void **SetFrom** (int32_t from)
- void **SetFrom** (const std::string &from)
- void **SetFromText** (const std::string &values, const std::string &delimiter=CCriteriaPassInt::m_delimiter)
- void **SetTo** (int32_t to)
- void **SetTo** (const std::string &to)
- virtual ~**CCriteriaPassInt** ()
Destructor.

Static Public Member Functions

- static **CCriteriaPassInt** * **GetCriteria** (CBratObject *ob, bool withExcept=true)

Static Public Attributes

- static const std::string **m_delimiter** = " "

Protected Member Functions

- void **Adjust** ()
- void **Init** ()

Protected Attributes

- int32_t **m_from**
- int32_t **m_to**

Additional Inherited Members

8.31.1 Detailed Description

Pass number (from/to) Criteria management class.

Version

1.0

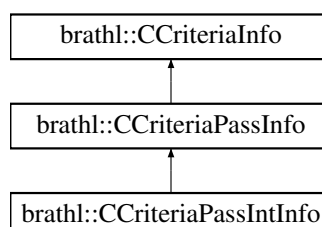
The documentation for this class was generated from the following files:

- CriteriaPass.h
- CriteriaPass.cpp

8.32 brathl::CCriteriaPassIntInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for brathl::CCriteriaPassIntInfo:



Public Member Functions

- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.

Static Public Member Functions

- static **CCriteriaPassIntInfo** * **GetCriteriaInfo** (CBratObject *ob, bool withExcept=true)

Additional Inherited Members

8.32.1 Detailed Description

Integer Pass criteria information management class.

Version

1.0

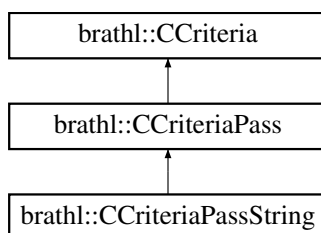
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.33 brathl::CCriteriaPassString Class Reference

```
#include <CriteriaPass.h>
```

Inheritance diagram for brathl::CCriteriaPassString:



Public Member Functions

- **CCriteriaPassString** ()
Empty CCriteriaPassString (p. 166) ctor.
- **CCriteriaPassString** (CCriteriaPassString &c)
- **CCriteriaPassString** (CCriteriaPassString *c)
- **CCriteriaPassString** (const std::string &passes, const std::string &delimiter=CCriteriaPassString::m_delimiter)
- **CCriteriaPassString** (const CStringArray &array)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- std::string **GetAsText** (const std::string &delimiter=CCriteriaPassString::m_delimiter)
- CStringArray * **GetPasses** ()
- bool **Intersect** (const std::string &passes, CStringArray &intersect)
- bool **Intersect** (CStringArray &passes, CStringArray &intersect)
- bool **IsDefaultValue** ()
- const **CCriteriaPassString** & **operator=** (CCriteriaPassString &c)
- void **Set** (const std::string &passes, const std::string &delimiter=CCriteriaPassString::m_delimiter)
- void **Set** (const CStringArray &array)
- void **Set** (CCriteriaPassString &c)
- void **SetDefaultValue** ()
- virtual ~**CCriteriaPassString** ()
Destructor.

Static Public Member Functions

- static **CCriteriaPassString * GetCriteria** (CBratObject *ob, bool withExcept=true)

Static Public Attributes

- static const std::string **m_delimiter** = ","

Protected Member Functions

- void **Init** ()

Static Protected Member Functions

- static void **ExtractPass** (const std::string &passes, CStringArray &arrayPass, const std::string &delimiter=CCriteriaPassString::m_delimiter)
- static void **ExtractPass** (const CStringArray &array, CStringArray &arrayPass)

Protected Attributes

- CStringArray **m_passes**

Additional Inherited Members

8.33.1 Detailed Description

Pass number (as std::string) Criteria management class.

Version

1.0

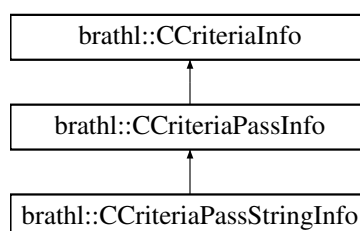
The documentation for this class was generated from the following files:

- CriteriaPass.h
- CriteriaPass.cpp

8.34 bratl::CCriteriaPassStringInfo Class Reference

```
#include <CriteriaInfo.h>
```

Inheritance diagram for bratl::CCriteriaPassStringInfo:



Public Member Functions

- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.

Static Public Member Functions

- static **CCriteriaPassStringInfo** * **GetCriteriaInfo** (CBratObject *ob, bool withExcept=true)

Additional Inherited Members

8.34.1 Detailed Description

String Pass criteria information management class.

Version

1.0

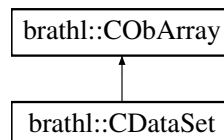
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.35 brathl::CDataSet Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CDataSet:



Public Member Functions

- **CRecordSet** * **Back** (bool withExcept=true)
- **CDataSet** (const std::string &name="", bool bDelete=true)
Ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual bool **Erase** (**CRecordSet** *recordSet)
- bool **EraseCurrentRecordSet** ()
- void **EraseFieldSet** (const std::string &fieldSetKey)
- **CRecordSet** * **FindRecord** (const std::string &recordSetName)
- **CRecordSet** * **GetCurrentRecordSet** ()
- **CFieldSet** * **GetFieldSet** (const std::string &fieldSetKey)
- **CFieldSetArrayDbI** * **GetFieldSetAsArrayDbI** (const std::string &fieldSetKey)
- **CFieldSetDbI** * **GetFieldSetAsDbI** (const std::string &fieldSetKey)
- double **GetFieldSetAsDbIValue** (const std::string &fieldSetKey)
- **CFieldSetString** * **GetFieldSetAsString** (const std::string &fieldSetKey)

- `std::string GetFieldSetAsStringValue` (const `std::string` &fieldSetKey)
- `CRecordSet * GetFirstRecordSet` ()
- const `std::string` & `GetName` ()
- `CRecord * GetRecord` (const `std::string` &recordSetName)
- `CRecord * GetRecord` (`CRecordSet *recordSet`)
- `CRecordSet * GetRecordSet` (`CDataSet::iterator itDataSet`)
- `CRecordSet * GetRecordSet` (`int32_t` index)
- `CObMap * GetRecordSetMap` ()
- void `InsertDataset` (`CDataSet *dataSet`, bool setAsCurrent=true)
- void `InsertFieldSet` (const `std::string` &fieldSetKey, `CFieldSet *fieldSet`)
- `CRecordSet * InsertRecord` (const `std::string` &recordSetName, bool setAsCurrent=true)
- virtual void `RemoveAll` ()
- void `SetCurrentRecordSet` (`int32_t` index)
- void `SetCurrentRecordSet` (`CDataSet::iterator itDataSet`)
- void `SetCurrentRecordSet` (const `std::string` &recordSetName)
- void `SetCurrentRecordSet` (`CRecordSet *recordSet`)
- void `SetName` (const `std::string` &name)
- virtual `~CDataSet` ()

Dtor.

Protected Attributes

- `CRecordSet * m_currentRecordSet`
- `std::string m_name`
- `CObMap m_recordSetMap`

8.35.1 Detailed Description

a set of recordset management classes.

Version

1.0

8.35.2 Member Function Documentation

8.35.2.1 void brathl::CDataSet::Dump (std::ostream & fOut = std::cerr) [virtual]

Dump fonction.

Copy a new **CDataSet** (p. 168) to the object

References brathl::CObArray::Dump().

Referenced by EraseFieldSet(), and InsertFieldSet().

8.35.2.2 void brathl::CDataSet::EraseFieldSet (const std::string & fieldSetKey)

remove a fieldset object (identify by its name) from the current recordset

Parameters

<i>fieldSetKey</i>	[in] : fieldset key
--------------------	---------------------

References Dump(), brathl::CObMap::Erase(), and brathl::CTools::Format().

8.35.2.3 CFieldSet * brathl::CDataSet::GetFieldSet (const std::string & fieldSetKey)

Gets the fieldset object (identify by its name) of the current recordset

Parameters

<i>fieldSetKey</i>	[in] : fieldset key to be searched
--------------------	------------------------------------

Returns

a pointer to the fieldset object if found, otherwise NULL

8.35.2.4 void brathl::CDataSet::InsertFieldSet (const std::string & *fieldSetKey*, CFieldSet * *fieldSet*)

Inserts a fieldset object (identify by its name) into the current recordset

Parameters

<i>fieldSetKey</i>	[in] : fieldset key
<i>fieldSet</i>	[in] : fieldset object to be inserted

References Dump(), brathl::CTools::Format(), and brathl::CObMap::Insert().

8.35.2.5 void brathl::CDataSet::RemoveAll () [virtual]

Remove all elements and clear the std::list

Reimplemented from **brathl::CObArray** (p. 44).

References brathl::CObArray::RemoveAll(), and brathl::CObMap::RemoveAll().

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

- Field.h
- Field.cpp

8.36 brathl::CDate Class Reference

```
#include <Date.h>
```

Public Member Functions

- int32_t **Add** (const **CDate** &d)
- int32_t **AddDays** (uint32_t days)
- std::string **AsString** (const std::string &format="", bool withMuSecond=false) const
- **CDate** ()
Constructs a date with a 1950/01/01 value.
- **CDate** (const char *strDate)
- **CDate** (const **CDate** &date)
*Constructs a date from another **CDate** (p. 170) object.*
- **CDate** (const uint32_t year, const uint32_t month=1, const uint32_t day=1, const uint32_t hour=0, const uint32_t minute=0, const uint32_t second=0, const uint32_t muSecond=0)
Constructs a date from year, month, day, hour, minute, second, microsecond.
- **CDate** (const uint32_t days, const uint32_t seconds, const uint32_t muSeconds, const brathl_refDate refDate=REF19500101)
Constructs a date from days, seconds, microseconds.
- **CDate** (const double days, const double seconds, const double muSeconds, const brathl_refDate refDate=REF19500101)
Constructs a date from days, seconds, microseconds.
- **CDate** (const double dateSeconds, brathl_refDate refDate=REF19500101)
- **CDate** (brathl_refDate refDate)
- int32_t **ConstructDate** (const brathl_refDate refDate)

- `int32_t Convert2DecimalJulian` (double &julian, const **brathl_refDate** refDate=**REF19500101**) const
- `int32_t Convert2DMM` (int32_t &days, int32_t &milliSeconds, int32_t &muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t Convert2DMM` (double &days, double &milliSeconds, double &muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t Convert2DSM` (int32_t &days, int32_t &seconds, int32_t &muSeconds, const **brathl_refDate** refDate=**REF19500101**) const
- `int32_t Convert2DSM` (double &days, double &seconds, double &muSeconds, const **brathl_refDate** refDate=**REF19500101**) const
- `int32_t Convert2Second` (double &seconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t Convert2SM` (int32_t &seconds, int32_t &muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t Convert2SM` (double &seconds, double &muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t Convert2YMDHMSM` (uint32_t &year, uint32_t &month, uint32_t &day, uint32_t &hour, uint32_t &minute, uint32_t &second, uint32_t &muSecond) const
- `uint32_t DayOfYear` ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
 - Dump fonction.*
- `uint32_t GetDay` () const
 - Gets the day of the date.*
- `uint32_t GetHour` () const
 - Gets the hour of the date.*
- `uint32_t GetMinute` () const
 - Gets the minutes of the date.*
- `uint32_t GetMonth` () const
 - Gets the month of the date.*
- `uint32_t GetMuSecond` () const
 - Gets the microseconds of the date.*
- `uint32_t GetSecond` () const
 - Gets the seconds of the date.*
- `uint32_t GetYear` () const
 - Gets the year of the date.*
- `uint32_t HowManyLeapYear` (const uint32_t year) const
- void **InitDateZero** ()
- bool **IsDefaultValue** () const
- bool **IsLeapYear** ()
- `int32_t LeapYearIndex` ()
- double **operator+** (const **CDate** &d)
- double **operator-** (const **CDate** &d)
- const **CDate** & **operator=** (const **CDate** &date)
- const **CDate** & **operator=** (const char *strDate)
- const **CDate** & **operator=** (double seconds)
- const **CDate** & **operator=** (const **brathl_refDate** refDate)
- `int32_t SetDate` (const char *strDate)
- `int32_t SetDate` (const **brathl_DateYMDHMSM** &date)
- `int32_t SetDate` (const **brathl_DateDSM** &date)
- `int32_t SetDate` (const uint32_t days, const uint32_t seconds, const uint32_t muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t SetDate` (const double days, const double seconds, const double muSeconds, const **brathl_refDate** refDate=**REF19500101**)
- `int32_t SetDate` (const **brathl_DateSecond** &date)
- `int32_t SetDate` (const **brathl_DateJulian** &date)
- `int32_t SetDate` (const uint32_t year, const uint32_t month=1, const uint32_t day=1, const uint32_t hour=0, const uint32_t minute=0, const uint32_t second=0, const uint32_t muSecond=0)
- `int32_t SetDate` (const double dateSeconds, **brathl_refDate** refDate=**REF19500101**)

- int32_t **SetDateJulian** (const double dateJulian, brathl_refDate refDate=REF19500101)
- int32_t **SetDateNow** ()
- void **SetDefaultValue** ()
- int32_t **SubtractDays** (uint32_t days)
- double **Value** () const

returns the date in a number of seconds since internal reference date, ie 1950)

- double **ValueJulian** () const

returns the date in a decimal julian day (since internal reference date, ie 1950)

- bool **operator<** (CDate &d)
- bool **operator<** (double d)
- bool **operator>** (CDate &d)
- bool **operator>** (double d)
- bool **operator==** (CDate &d)
- bool **operator==** (double d)
- bool **operator<=** (CDate &d)
- bool **operator<=** (double d)
- bool **operator>=** (CDate &d)
- bool **operator>=** (double d)
- bool **operator!=** (CDate &d)
- bool **operator!=** (double d)

Static Public Member Functions

- static int32_t **CheckDate** (const uint32_t year, const uint32_t month=1, const uint32_t day=1, const uint32_t hour=0, const uint32_t minute=0, const uint32_t second=0, const uint32_t muSecond=0)
- static int32_t **CheckDay** (uint32_t day, uint32_t month, uint32_t year)
- static int32_t **CheckHour** (uint32_t hour)
- static int32_t **CheckMinute** (uint32_t minute)
- static int32_t **CheckMonth** (uint32_t month)
- static int32_t **CheckMuSecond** (uint32_t muSecond)
- static int32_t **CheckSecond** (uint32_t second)
- static int32_t **CheckYear** (uint32_t year)
- static double **CvDate** (const char *strDate)
- static uint32_t **DayOfYear** (uint32_t year, uint32_t month, uint32_t day)
- static uint32_t **DayOfYear** (CDate &date)
- static int32_t **GetDateRef** (const CDate &date, brathl_refDate &refDate)
- static int32_t **GetDaysInMonth** (const uint32_t month, const uint32_t year, uint32_t &nbDaysInMonth)
- static bool **IsCharDate** (const char *strDate)
- static bool **IsLeapYear** (const uint32_t year)
- static int32_t **LeapYearIndex** (const uint32_t year)

Static Public Attributes

- static const uint32_t **m_daysInMonth** [2][12]
- static const uint32_t **m_daysOfYear** [2][12]
- static const char * **m_DEFAULT_UNIT_SECOND** = "second"
- static const uint32_t **m_internalRefYear** = 1950
- static const double **m_minutesInDay** = 1440.0
- static const double **m_minutesInHour** = 60.0
- static const double **m_secInDay** = 86400.0
- static const double **m_secInHour** = 3600.0
- static const double **m_secInMinute** = 60.0

8.36.1 Detailed Description

Date management and conversion class.

This class allows calendar an date conversion.

Warning

Date before 1950/01/01 00:00:00:00 are not accepted

Version

1.0

8.36.2 Constructor & Destructor Documentation

8.36.2.1 brathl::CDate::CDate (const char * *strDate*)

Constructs a date from a std::string

Parameters

<i>strDate</i>	: Allowed format are : <ul style="list-style-type: none"> • YYYY-MM-DD HH:MM:SS.MS std::string • a julian std::string (format:positive 'Days Seconds Microseconds' or positive decimal julian day)
----------------	--

References SetDate().

8.36.2.2 brathl::CDate::CDate (const double *dateSeconds*, brathl_refDate *refDate* = REF19500101)

Constructs a date value from a decimal number of seconds

Parameters

<i>dateSeconds</i>	[in]: decimal number of seconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

References SetDate().

8.36.3 Member Function Documentation

8.36.3.1 int32_t brathl::CDate::Add (const CDate & *d*)

Adds a date to the date object

Parameters

<i>d</i>	[in]: a CDate (p. 170) object to add
----------	---

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by SetDate(), and SetDateJulian().

8.36.3.2 int32_t brathl::CDate::AddDays (uint32_t *days*)

Adds a number of day to the date object

Parameters

<i>days</i>	[in]: number of days to add (if < 0, a subtract operation is performed)
-------------	---

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References m_minutesInDay.

8.36.3.3 `std::string brathl::CDate::AsString (const std::string & format = " ", bool withMuSecond = false) const`

Formats a date as std::string.

Parameters

<i>Format</i>	[in] : String controlling how the date will be converted into std::string. This format std::string consists of zero or more conversion specifications and ordinary characters. A conversion specification consists of a " (percent) character and one or two terminating conversion characters that determine the conversion specification's behavior. All ordinary characters are copied unchanged into the result. Each conversion specification is replaced by appropriate characters as described in the following list. The appropriate characters are determined by the LC_TIME category of the program's locale. %% Same as %. a Locale's abbreviated weekday name. A Locale's full weekday name. b Locale's abbreviated month name. B Locale's full month name. c Locale's appropriate date and time representation. C Century number (the year divided by 100 and truncated to an integer as a decimal number [1,99]); single digits are preceded by 0; see standards(5). d Day of month [1,31]; single digits are preceded by 0. H Hour (24-hour clock) [0,23]; single digits are preceded by 0. I Hour (12-hour clock) [1,12]; single digits are preceded by 0. j Day number of year [1,366]; single digits are preceded by 0. m Month number [1,12]; single digits are preceded by 0. M Minute [00,59]; leading 0 is permitted but not required. p Locale's equivalent of either a.m. or p.m. S Seconds [00,61]; the range of values is [00,61] rather than [00,59] to allow for the occasional leap second and even more occasional double leap second. U Week number of year as a decimal number [00,53], with Sunday as the first day of week 1. w Weekday as a decimal number [0,6], with 0 representing Sunday. W Week number of year as a decimal number [00,53], with Monday as the first day of week 1. x Locale's appropriate date representation. X Locale's appropriate time representation. y Year within century [00,99]. Y Year, including the century (for example 1993). Z Time zone name or abbreviation, or no bytes if no time zone information exists. If the format is an empty std::string it is forced to be "%Y-%m-%d %H:%M:%S" (ISO 8601)
---------------	--

<i>withMuSecond</i>	[in] : add the microseconds of the date at the end of the std::string (format "%.06u")
---------------------	--

Returns

Formatted std::string

References Convert2YMDHMSM(), brathl::CTools::Format(), and IsDefaultValue().

8.36.3.4 `int32_t brathl::CDate::CheckDate (const uint32_t year, const uint32_t month = 1, const uint32_t day = 1, const uint32_t hour = 0, const uint32_t minute = 0, const uint32_t second = 0, const uint32_t muSecond = 0)`
[static]

Check if a date value (year, month, day, hour, minute, second, microsecond) is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References CheckDay(), CheckHour(), CheckMinute(), CheckMonth(), CheckMuSecond(), CheckSecond(), and CheckYear().

8.36.3.5 `int32_t brathl::CDate::CheckDay (uint32_t day, uint32_t month, uint32_t year)` [static]

Checks if a day value is valid, according to a month an a year

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References CheckMonth(), and GetDaysInMonth().

Referenced by CheckDate().

8.36.3.6 `int32_t brathl::CDate::CheckHour (uint32_t hour)` [static]

Checks if an hour value is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate().

8.36.3.7 `int32_t brathl::CDate::CheckMinute (uint32_t minute)` [static]

Checks if a minute is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate().

8.36.3.8 `int32_t brathl::CDate::CheckMonth (uint32_t month)` [static]

Checks if a month value is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate(), CheckDay(), DayOfYear(), and GetDaysInMonth().

8.36.3.9 `int32_t brathl::CDate::CheckMuSecond (uint32_t muSecond) [static]`

Checks if a month value is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate().

8.36.3.10 `int32_t brathl::CDate::CheckSecond (uint32_t second) [static]`

Checks if a second value is valid

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate().

8.36.3.11 `int32_t brathl::CDate::CheckYear (uint32_t year) [static]`

Checks if a year value is valid year have to be >= internal reference year (1950)

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by CheckDate(), and DayOfYear().

8.36.3.12 `int32_t brathl::CDate::ConstructDate (const brathl_refDate refDate)`

Converts a date whose value corresponds to the date reference enumeration

Parameters

<i>refDate</i>	[in]: date reference - see brathl_refDate (p. 317))
----------------	--

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_refDateUser1, brathl_refDateUser2, REF19500101, REF19580101, REF19850101, REF19900101, REF20000101, REFUSER1, REFUSER2, and SetDate().

8.36.3.13 `int32_t brathl::CDate::Convert2DecimalJulian (double & julian, const brathl_refDate refDate = REF19500101) const`

Converts the date value into a decimal julian day

Parameters

<i>julian</i>	[out]: decimal julian day (can be < 0)
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References m_secInDay.

Referenced by brathl_DSM2Julian(), brathl_Seconds2Julian(), brathl_YMDHMSM2Julian(), and ValueJulian().

8.36.3.14 `int32_t brathl::CDate::Convert2DMM (int32_t & days, int32_t & milliSeconds, int32_t & muSeconds, const
brathl_refDate refDate = REF19500101)`

Converts the date value into a number of days, milliseconds, microseconds

Parameters

<i>days</i>	[out]: number of days (can be < 0)
<i>milliSeconds</i>	[out]: number of milliseconds
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References IsDefaultValue(), and m_minutesInDay.

Referenced by Convert2DMM().

8.36.3.15 `int32_t brathl::CDate::Convert2DMM (double & days, double & milliSeconds, double & muSeconds, const brathl_refDate refDate = REF19500101)`

Converts the date value into a number of days, milliseconds, microseconds

Parameters

<i>days</i>	[out]: number of days (can be < 0)
<i>milliSeconds</i>	[out]: number of milliseconds
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References Convert2DMM().

8.36.3.16 `int32_t brathl::CDate::Convert2DSM (int32_t & days, int32_t & seconds, int32_t & muSeconds, const brathl_refDate refDate = REF19500101) const`

Converts the date value into a number of days, seconds, microseconds

Parameters

<i>days</i>	[out]: number of days (can be < 0)
<i>seconds</i>	[out]: number of seconds
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References m_minutesInDay, m_secInDay, and m_secInMinute.

Referenced by brathl_Julian2DSM(), brathl_Seconds2DSM(), and brathl_YMDHMSM2DSM().

8.36.3.17 `int32_t brathl::CDate::Convert2DSM (double & days, double & seconds, double & muSeconds, const brathl_refDate refDate = REF19500101) const`

Converts the date value into a number of days, seconds, microseconds

Parameters

<i>days</i>	[out]: number of days (can be < 0)
<i>seconds</i>	[out]: number of seconds
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

8.36.3.18 `int32_t brathl::CDate::Convert2Second (double & seconds, const brathl_refDate refDate = REF19500101)`

Converts the date value into a decimal number of seconds

Parameters

<i>seconds</i>	[out]: decimal number of seconds day (can be < 0)
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References Value().

Referenced by brathl_DSM2Seconds(), brathl_Julian2Seconds(), and brathl_YMDHMSM2Seconds().

8.36.3.19 `int32_t brathl::CDate::Convert2SM (int32_t & seconds, int32_t & muSeconds, const brathl_refDate refDate = REF19500101)`

Converts the date value into a number of seconds, microseconds

Parameters

<i>seconds</i>	[out]: number of milliseconds (can be < 0)
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References IsDefaultValue(), and m_secInMinute.

Referenced by Convert2SM().

8.36.3.20 `int32_t brathl::CDate::Convert2SM (double & seconds, double & muSeconds, const brathl_refDate refDate = REF19500101)`

Converts the date value into a number of seconds, microseconds

Parameters

<i>seconds</i>	[out]: number of milliseconds (can be < 0)
<i>muSeconds</i>	[out]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References Convert2SM().

8.36.3.21 `int32_t brathl::CDate::Convert2YMDHMSM (uint32_t & year, uint32_t & month, uint32_t & day, uint32_t & hour,
uint32_t & minute, uint32_t & second, uint32_t & muSecond) const`

Converts the date value into year, month, day, hour, minute, second, microsecond

Parameters

<i>year</i>	[out]: year
<i>month</i>	[out]: month
<i>day</i>	[out]: day
<i>hour</i>	[out]: hour
<i>minute</i>	[out]: minute
<i>second</i>	[out]: second
<i>muSecond</i>	[out]: microsecond

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References HowManyLeapYear(), IsDefaultValue(), LeapYearIndex(), m_internalRefYear, m_minutesInDay, and m_minutesInHour.

Referenced by AsString(), brathl_Cycle2YMDHMSM(), brathl_DSM2YMDHMSM(), brathl_Julian2YMDHMSM(), brathl_NowYMDHMSM(), brathl_Seconds2YMDHMSM(), GetDay(), GetHour(), GetMinute(), GetMonth(), GetMuSecond(), GetSecond(), and GetYear().

8.36.3.22 double brathl::CDate::CvDate (const char * *strDate*) [static]

Convert a date std::string to a number of seconds since internal reference year (ie 1950) Allowed format are :

- YYYY-MM-DD HH:MM:SS.MS std::string
- a julian std::string (format:positive 'Days Seconds Microseconds' or positive decimal julian day) For julian std::string, it can contain its date reference at the end by specifying where YYYY the reference year. If no date reference is specified the default date reference is used.

Parameters

<i>strDate</i>	: date std::string
----------------	--------------------

Returns

number of seconds since internal reference year (ie 1950)

References brathl::CTools::Format(), SetDate(), and Value().

8.36.3.23 uint32_t brathl::CDate::DayOfYear (uint32_t *year*, uint32_t *month*, uint32_t *day*) [static]

Retrieves the day of a year if year is not valid, methods force the value to the internal reference year (1950) if month is not valid, methods force the value to 1 day value is not check

Parameters

<i>year</i>	[in]: year
<i>month</i>	[in]: month of year
<i>day</i>	[in]: day of the month

Returns

the day of year

References CheckMonth(), CheckYear(), LeapYearIndex(), and m_internalRefYear.

Referenced by brathl_DayOfYear().

8.36.3.24 uint32_t brathl::CDate::DayOfYear (CDate & *date*) [static]

Retrieves the day of year of a **CDate** (p. 170) object

Parameters

<i>date</i>	[in]: date
-------------	------------

Returns

the day of year

References GetDay(), GetMonth(), and LeapYearIndex().

8.36.3.25 uint32_t brathl::CDate::DayOfYear ()

Retrieves the day of year of the date object

Returns

the day of year

8.36.3.26 int32_t brathl::CDate::GetDateRef (const CDate & date, brathl_refDate & refDate) [static]

Construct a date reference enumeration according to a **CDate** (p. 170) object Only date according to **brathl_refDate** (p. 317) enumeration are valid, furthermore REFUSER1 and REFUSER2 are not allowed.

Parameters

<i>date</i>	[in] : CDate (p. 170) object whose value corresponds to the refDate parameter
<i>refDate</i>	[out]: date reference enumeration value (see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS if **CDate** (p. 170) object is according to **brathl_refDate** (p. 317) enumeration except REFUSER1 and REFUSER2. Otherwise returns a erro code (see Date_error_codes)

References GetYear(), REF19500101, REF19580101, REF19850101, REF19900101, and REF20000101.

8.36.3.27 int32_t brathl::CDate::GetDaysInMonth (const uint32_t month, const uint32_t year, uint32_t & nbDaysInMonth) [static]

Retrieves the number of days in a month, according to a year and a month

Parameters

<i>month</i>	[in] : month
<i>year</i>	[in] : year
<i>nbDaysInMonth</i> ↔ <i>Month[out]</i>	: number of days in the month

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References CheckMonth(), LeapYearIndex(), and m_daysInMonth.

Referenced by CheckDay().

8.36.3.28 uint32_t brathl::CDate::HowManyLeapYear (const uint32_t year) const

Computes the number of leap years since a year

Parameters

<i>year</i>	[in]: year
-------------	------------

Returns

number of leap years

References `IsLeapYear()`, and `m_internalRefYear`.

Referenced by `Convert2YMDHMSM()`.

8.36.3.29 void brathl::CDate::InitDateZero ()

Initializes a **CDate** (p. 170) object to 0

Referenced by `CDate()`.

8.36.3.30 bool brathl::CDate::IsDefaultValue () const

Tests the internal value to the default value

Returns

true if default value, otherwise false

Referenced by `AsString()`, `Convert2DMM()`, `Convert2SM()`, `Convert2YMDHMSM()`, `brathl::CDatePeriod::Intersect()`, and `brathl::CDatePeriod::IsDefaultValue()`.

8.36.3.31 bool brathl::CDate::IsLeapYear (const uint32_t year) [static]

Testd if the year is a leap year

Parameters

<i>year</i>	[in]: year to test
-------------	--------------------

Returns

true if the year is a leap year, otherwise false

8.36.3.32 bool brathl::CDate::IsLeapYear ()

Tests if the year of the date object is a leap year

Returns

true if the year of the date object is a leap year, otherwise false

References `GetYear()`.

Referenced by `HowManyLeapYear()`, and `LeapYearIndex()`.

8.36.3.33 int32_t brathl::CDate::LeapYearIndex (const uint32_t year) [static]

Retrieves the index of the **m_daysOfYear** (p. 186) or **m_daysInMonth** (p. 186) arrays in accordance with the year (leap year or not)

Parameters

<i>year</i>	[in]: year to test
-------------	--------------------

Returns

0 if year is a leap year, otherwise 1

References IsLeapYear().

Referenced by DayOfYear().

8.36.3.34 int32_t brathl::CDate::LeapYearIndex ()

Retrieve sthe index of the daysOfYear or daysInMonth arrays in accordance with the year of the date object (leap year or not)

Returns

0 if year of the date object is a leap year, otherwise 1

References GetYear().

Referenced by Convert2YMDHMSM(), DayOfYear(), and GetDaysInMonth().

8.36.3.35 double brathl::CDate::operator+ (const CDate & d) [inline]

Plus operator redefinition Computes the addition of two dates, the result is expressed in a decimal number of seconds

References Value().

8.36.3.36 double brathl::CDate::operator- (const CDate & d) [inline]

Minus operator redefinition Computes the difference between two dates, the result is expressed in a decimal number of seconds

References Value().

8.36.3.37 bool brathl::CDate::operator< (CDate & d) [inline]

Comparison operators

References Value().

8.36.3.38 const CDate & brathl::CDate::operator= (const CDate & date)

Assigns a new value to the **CDate** (p. 170) object, with a **CDate** (p. 170) object

8.36.3.39 const CDate & brathl::CDate::operator= (const char * strDate)

Assigns a new value to the **CDate** (p. 170) object, with a date std::string (format: YYYY-MM-DD HH:MN:SS.MS)

References SetDate().

8.36.3.40 const CDate & brathl::CDate::operator= (double seconds)

Assigns a new value to the **CDate** (p. 170) object, with a number of seconds since 1950-01-01

References SetDate().

8.36.3.41 const CDate & brathl::CDate::operator= (const brathl_refDate refDate)

Assigns a new value to the **CDate** (p. 170) object, with a reference date

References SetDate().

8.36.3.42 int32_t brathl::CDate::SetDate (const char * *strDate*)

Sets date value from a std::string Allowed format are :

- YYYY-MM-DD HH:MM:SS.MS std::string
- a julian std::string (format:positive 'Days Seconds Microseconds' or positive decimal julian day) For julian std::string, it can contain its date reference at the end by specifying where YYYY the reference year. If no date reference is specified the default date reference is used.

Parameters

<i>strDate</i>	: date std::string
----------------	--------------------

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References SetDefaultValue().

Referenced by brathl_DayOfYear(), brathl_DiffDSM(), brathl_DiffJulian(), brathl_DiffYMDHMSM(), brathl_DS↵M2Julian(), brathl_DSM2Seconds(), brathl_DSM2YMDHMSM(), brathl_Julian2DSM(), brathl_Julian2Seconds(), brathl_Julian2YMDHMSM(), brathl_Seconds2DSM(), brathl_Seconds2Julian(), brathl_Seconds2YMDHMS↵M(), brathl_YMDHMSM2Cycle(), brathl_YMDHMSM2DSM(), brathl_YMDHMSM2Julian(), brathl_YMDHMSM2↵Seconds(), CDate(), ConstructDate(), CvDate(), operator=(), SetDate(), brathl::CDatePeriod::SetFrom(), and brathl::CDatePeriod::SetTo().

8.36.3.43 int32_t brathl::CDate::SetDate (const brathl_DateYMDHMSM & *date*)

Sets date value from a **brathl_DateYMDHMSM** (p. 316) structure

Parameters

<i>date</i>	[in]: brathl_DateYMDHMSM (p. 316) structure date
-------------	---

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References SetDate().

8.36.3.44 int32_t brathl::CDate::SetDate (const brathl_DateDSM & *date*)

Sets date value from a **brathl_DateDSM** (p. 316) structure

Parameters

<i>date</i>	[in]: brathl_DateDSM (p. 316) structure date
-------------	---

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References _structDateDSM::days, _structDateDSM::muSeconds, _structDateDSM::refDate, and _structDateDS↵M::seconds.

8.36.3.45 int32_t brathl::CDate::SetDate (const uint32_t *days*, const uint32_t *seconds*, const uint32_t *muSeconds*, const brathl_refDate *refDate* = REF19500101)

Sets date value from year, month, day, hour, minute, second, microsecond

Parameters

<i>days</i>	[in]: number of days
<i>seconds</i>	[in]: number of seconds
<i>muSeconds</i>	[in]: number of microseconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

8.36.3.46 int32_t brathl::CDate::SetDate (const brathl_DateSecond & date)

Sets date value from a **brathl_DateSecond** (p. 316) structure

Parameters

<i>date</i>	[in]: brathl_DateSecond (p. 316) structure date
-------------	--

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References _structDateSecond::nbSeconds, _structDateSecond::refDate, and SetDate().

8.36.3.47 int32_t brathl::CDate::SetDate (const brathl_DateJulian & date)

Sets date value from a **brathl_DateJulian** (p. 316) structure

Parameters

<i>date</i>	[in]: brathl_DateJulian (p. 316) structure date
-------------	--

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References _structDateJulian::julian, _structDateJulian::refDate, and SetDateJulian().

8.36.3.48 int32_t brathl::CDate::SetDate (const uint32_t year, const uint32_t month = 1, const uint32_t day = 1, const uint32_t hour = 0, const uint32_t minute = 0, const uint32_t second = 0, const uint32_t muSecond = 0)

Sets date value from year, month, day, hour, minute, second, microsecond

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

8.36.3.49 int32_t brathl::CDate::SetDate (const double dateSeconds, brathl_refDate refDate = REF19500101)

Sets date value from a decimal number of seconds

Parameters

<i>dateSeconds</i>	[in]: decimal number of seconds
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References Add(), m_secInMinute, SetDate(), and SetDefaultValue().

8.36.3.50 `int32_t brathl::CDate::SetDateJulian (const double dateJulian, brathl_refDate refDate = REF19500101)`

Sets date value from a decimal julian day

Parameters

<i>dateJulian</i>	[in]: decimal julian day
<i>refDate</i>	[in]: date reference (default value is REF19500101 - see brathl_refDate (p. 317))

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References Add(), m_minutesInDay, SetDefaultValue(), and ValueJulian().

Referenced by SetDate().

8.36.3.51 int32_t brathl::CDate::SetDateNow ()

Sets the date object to the current time

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

Referenced by brathl_NowYMDHMSM().

8.36.3.52 void brathl::CDate::SetDefaultValue ()

Sets internal value to the default value

Referenced by SetDate(), SetDateJulian(), and brathl::CDatePeriod::SetDefaultValue().

8.36.3.53 int32_t brathl::CDate::SubtractDays (uint32_t days)

Subtracts a number of day from the date object

Parameters

<i>days</i>	[in]: number of days to subtract (if < 0, a addition operation is performed)
-------------	--

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References m_minutesInDay.

8.36.4 Member Data Documentation

8.36.4.1 const uint32_t brathl::CDate::m_daysInMonth [static]

Initial value:

```
=
{
    { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
    { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }
}
```

Array[i,j] of number of days in month i : 0 corresponds to a leap year, 1 corresponds to a non-leap year j : index of the month

Referenced by GetDaysInMonth().

8.36.4.2 `const uint32_t bratl::CDate::m_daysOfYear` [static]

Initial value:

```
=
{
    {0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335},
    {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334}
}
```

Array[i,j] of day of year i : 0 corresponds to a leap year, 1 corresponds to a non-leap year j : index of the month

8.36.4.3 `const uint32_t bratl::CDate::m_internalRefYear = 1950` [static]

Internal reference year (1950)

Referenced by `Convert2YMDHMSM()`, `DayOfYear()`, and `HowManyLeapYear()`.

8.36.4.4 `const double bratl::CDate::m_minutesInDay = 1440.0` [static]

Number of minutes in a day

Referenced by `AddDays()`, `Convert2DMM()`, `Convert2DSM()`, `Convert2YMDHMSM()`, `SetDateJulian()`, and `SubtractDays()`.

8.36.4.5 `const double bratl::CDate::m_minutesInHour = 60.0` [static]

Number of minutes in an hour

Referenced by `Convert2YMDHMSM()`.

8.36.4.6 `const double bratl::CDate::m_secInDay = 86400.0` [static]

Number of seconds in a day

Referenced by `Convert2DecimalJulian()`, and `Convert2DSM()`.

8.36.4.7 `const double bratl::CDate::m_secInHour = 3600.0` [static]

Number of seconds in an hour

8.36.4.8 `const double bratl::CDate::m_secInMinute = 60.0` [static]

Number of seconds in a minute

Referenced by `Convert2DSM()`, `Convert2SM()`, and `SetDate()`.

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

- `Date.h`
- `Date.cpp`

8.37 `bratl::CDatePeriod` Class Reference

```
#include <DatePeriod.h>
```

Inherits `bratl::CBratObject`.

Public Member Functions

- `std::string AsString` (const `std::string` &format="", bool withMuSecond=false)
- `CDatePeriod` ()
Empty *CDatePeriod* (p. 187) ctor.

- **CDatePeriod** (**CDatePeriod** &datePeriod)
 - **CDatePeriod** (**CDate** &from, **CDate** &to)
 - **CDatePeriod** (const std::string &from, const std::string &to)
 - **CDatePeriod** (double from, double to)
 - **CDatePeriod** (const CStringArray &array)
 - virtual void **Dump** (std::ostream &fOut=std::cerr)
- Dump function.*
- std::string **GetAsText** (const std::string &delimiter=CDatePeriod::m_delimiter)
 - std::string **GetFormat** ()
 - **CDate** & **GetFrom** ()
 - std::string **GetFromAsText** ()
 - **CDate** & **GetTo** ()
 - std::string **GetToAsText** ()
 - bool **GetWithMuSecond** ()
 - bool **Intersect** (**CDatePeriod** &datePeriod, **CDatePeriod** &intersect)
 - bool **Intersect** (**CDate** &otherFrom, **CDate** &otherTo, **CDatePeriod** &intersect)
 - bool **IsDefaultValue** ()
 - const **CDatePeriod** & **operator=** (**CDatePeriod** &datePeriod)
 - void **Set** (**CDate** &from, **CDate** &to)
 - void **Set** (const std::string &from, const std::string &to)
 - void **Set** (double from, double to)
 - void **Set** (const CStringArray &array)
 - void **Set** (**CDatePeriod** &datePeriod)
 - void **SetDefaultValue** ()
 - void **SetFormat** (const std::string &value)
 - void **SetFrom** (**CDate** &from)
 - void **SetFrom** (const std::string &strDate)
 - void **SetTo** (**CDate** &to)
 - void **SetTo** (const std::string &strDate)
 - void **SetWithMuSecond** (bool value)
 - bool **Union** (**CDatePeriod** &datePeriod)
 - bool **Union** (**CDate** &otherFrom, **CDate** &otherTo)
 - bool **Union** (**CDatePeriod** &datePeriod, **CDatePeriod** &unionDate)
 - bool **Union** (**CDate** &otherFrom, **CDate** &otherTo, **CDatePeriod** &unionDate)
 - virtual ~**CDatePeriod** ()

Destructor.

Static Public Attributes

- static const std::string **m_delimiter** = "/"

Protected Member Functions

- void **Adjust** ()
- void **Init** ()

Protected Attributes

- std::string **m_format**
- **CDate** **m_from**
- **CDate** **m_to**
- bool **m_withMuSecond**

8.37.1 Detailed Description

Date interval management class.

Version

1.0

8.37.2 Constructor & Destructor Documentation

8.37.2.1 `brathl::CDatePeriod::CDatePeriod (CDatePeriod & datePeriod)`

Copy constructor.

Parameters

<i>datePeriod</i>	period to set
-------------------	---------------

References Set().

8.37.2.2 `brathl::CDatePeriod::CDatePeriod (CDate & from, CDate & to)`

Constructor.

Parameters

<i>from</i>	start date
<i>to</i>	end date

References SetFrom(), and SetTo().

8.37.2.3 `brathl::CDatePeriod::CDatePeriod (const std::string & from, const std::string & to)`

Constructor.

Parameters

<i>from</i>	start date
<i>to</i>	end date

References Set().

8.37.2.4 `brathl::CDatePeriod::CDatePeriod (double from, double to)`

Constructor.

Parameters

<i>from</i>	start date (number of seconds since 1950-01-01)
<i>to</i>	end date (number of seconds since 1950-01-01)

References Set().

8.37.2.5 `brathl::CDatePeriod::CDatePeriod (const CStringArray & array)`

Constructor from a array that contains start date as std::string, end date as std::string

Parameters

<i>array</i>	start and end dates
--------------	---------------------

References Set().

8.37.3 Member Function Documentation

8.37.3.1 CDate& brathl::CDatePeriod::GetFrom () [inline]

Gets start date

Returns

start date

References m_from.

Referenced by Intersect(), and Set().

8.37.3.2 CDate& brathl::CDatePeriod::GetTo () [inline]

Gets end date

Returns

end date

References m_to.

Referenced by Intersect(), and Set().

8.37.3.3 bool brathl::CDatePeriod::Intersect (CDatePeriod & *datePeriod*, CDatePeriod & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>datePeriod</i>	intersect with this
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References GetFrom(), and GetTo().

Referenced by brathl::CCriteriaDatetime::Intersect().

8.37.3.4 bool brathl::CDatePeriod::Intersect (CDate & *otherFrom*, CDate & *otherTo*, CDatePeriod & *intersect*)

Create the intersection of this date period with the given one

Parameters

<i>otherFrom</i>	start date intersect with this
<i>otherTo</i>	end date intersect with this
<i>intersect</i>	intersection period

Returns

true, or false if there is no intersection

References IsDefaultValue(), brathl::CDate::IsDefaultValue(), m_from, m_to, SetFrom(), and SetTo().

8.37.3.5 bool brathl::CDatePeriod::IsDefaultValue ()

Tests whether date period have been initialized or not

Returns

true if not initialized

References `brathl::CDate::IsDefaultValue()`, `m_from`, and `m_to`.

Referenced by `Intersect()`, and `brathl::CCriteriaDatetime::IsDefaultValue()`.

8.37.3.6 const CDatePeriod & brathl::CDatePeriod::operator= (CDatePeriod & datePeriod)

Assigns a new value to the **CDatePeriod** (p. 187) object, with a **CDatePeriod** (p. 187) object

References `Set()`.

8.37.3.7 void brathl::CDatePeriod::Set (CDate & from, CDate & to)

Sets date period from start and end date

Parameters

<i>from</i>	start date
<i>to</i>	end date

References `SetFrom()`, and `SetTo()`.

Referenced by `CDatePeriod()`, `operator=()`, `brathl::CCriteriaDatetime::Set()`, and `Set()`.

8.37.3.8 void brathl::CDatePeriod::Set (const std::string & from, const std::string & to)

Sets date period from start and end date

Parameters

<i>from</i>	start date
<i>to</i>	end date

References `SetFrom()`, and `SetTo()`.

8.37.3.9 void brathl::CDatePeriod::Set (const CStringArray & array)

Sets a date period from a array that contains start date as `std::string`, end date as `std::string`

Parameters

<i>array</i>	start and end dates
--------------	---------------------

References `Set()`.

8.37.3.10 void brathl::CDatePeriod::Set (CDatePeriod & datePeriod)

Sets date period from another one

Parameters

<i>datePeriod</i>	period to set
-------------------	---------------

References `GetFrom()`, `GetTo()`, `SetFrom()`, and `SetTo()`.

8.37.3.11 void brathl::CDatePeriod::SetDefaultValue ()

Sets internal value to the default value (uninitialized)

References `m_from`, `m_to`, and `brathl::CDate::SetDefaultValue()`.

Referenced by `brathl::CCriteriaDatetime::SetDefaultValue()`.

8.37.3.12 void brathl::CDatePeriod::SetFrom (CDate & *from*)

Sets start date

Parameters

<i>to</i>	start date
-----------	------------

References `m_from`.

Referenced by `CDatePeriod()`, `Intersect()`, `Set()`, and `brathl::CCriteriaDatetime::SetFrom()`.

8.37.3.13 `void brathl::CDatePeriod::SetFrom (const std::string & strDate)`

Sets start date

Parameters

<i>to</i>	start date
-----------	------------

References `brathl::CTools::Format()`, `m_from`, and `brathl::CDate::SetDate()`.

8.37.3.14 `void brathl::CDatePeriod::SetTo (CDate & to)`

Sets end date

Parameters

<i>to</i>	end date
-----------	----------

References `m_to`.

Referenced by `CDatePeriod()`, `Intersect()`, `Set()`, and `brathl::CCriteriaDatetime::SetTo()`.

8.37.3.15 `void brathl::CDatePeriod::SetTo (const std::string & strDate)`

Sets end date

Parameters

<i>to</i>	end date
-----------	----------

References `brathl::CTools::Format()`, `m_to`, and `brathl::CDate::SetDate()`.

8.37.4 Member Data Documentation

8.37.4.1 `CDate brathl::CDatePeriod::m_from` `[protected]`

Start date

Referenced by `Dump()`, `GetFrom()`, `Intersect()`, `IsDefaultValue()`, `SetDefaultValue()`, and `SetFrom()`.

8.37.4.2 `CDate brathl::CDatePeriod::m_to` `[protected]`

End date

Referenced by `Dump()`, `GetTo()`, `Intersect()`, `IsDefaultValue()`, `SetDefaultValue()`, and `SetTo()`.

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

- `DatePeriod.h`
- `DatePeriod.cpp`

8.38 `brathl::CDoubleMap` Class Reference

```
#include <List.h>
```

Inherits `mapdouble`.

Public Member Functions

- **CDoubleMap** ()
CDoubleMap (p. 192) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CDoubleMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual double **Exists** (const std::string &key) const
- virtual double **Insert** (const std::string &key, double value, bool withExcept=true)
- virtual double **operator[]** (const std::string &key)
- virtual void **RemoveAll** ()
- virtual ~**CDoubleMap** ()
CDoubleMap (p. 192) dtor.

8.38.1 Detailed Description

a set of double value management classes.

Version

1.0

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

- List.h
- List.cpp

8.39 brathl::CDoublePtrArray Class Reference

```
#include <List.h>
```

Inherits doubleptrarray.

Public Member Functions

- **CDoublePtrArray** (bool bDelete=true)
Empty CDoublePtrArray (p. 193) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CDoublePtrArray::iterator it)
- virtual bool **Erase** (int32_t index)
- bool **GetDelete** ()
- uint32_t **GetMatrixDim** (uint32_t row)
- CUIntArray * **GetMatrixDims** ()
- size_t **GetMatrixNumberOfDims** ()
- virtual void **Insert** (DoublePtr ob)
- virtual CDoublePtrArray::iterator **InsertAt** (CDoublePtrArray::iterator where, DoublePtr ob)
- DoublePtr **NewMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- virtual bool **PopBack** ()
- virtual void **RemoveAll** ()
- virtual CDoublePtrArray::iterator **ReplaceAt** (CDoublePtrArray::iterator where, DoublePtr ob)
- void **SetDelete** (bool value)
- void **SetMatrixDims** (const CUIntArray &matrixDims)
- virtual ~**CDoublePtrArray** ()
Destructor.

Protected Member Functions

- void **Delete** (DoublePtr matrix)

Protected Attributes

- bool **m_bDelete**
- CUIntArray **m_matrixDims**

8.39.1 Detailed Description

An array (std::vector) of duple pointer management class.

Version

1.0

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

- List.h
- List.cpp

8.40 brathl::CDoublePtrDoubleMap Class Reference

```
#include <List.h>
```

Inherits mapdoubledoubleptr.

Public Member Functions

- **CDoublePtrDoubleMap** (bool bDelete=true)
CDoublePtrDoubleMap (p. 194) ctor.
- **CDoublePtrDoubleMap** (const CUIntArray &matrixDims, bool bDelete=true)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CDoublePtrDoubleMap::iterator it)
- virtual bool **Erase** (double key)
- virtual DoublePtr * **Exists** (double key) const
- bool **GetDelete** ()
- virtual void **GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- uint32_t **GetMatrixColDim** (uint32_t row)
- CUIntArray * **GetMatrixDims** ()
- size_t **GetMatrixNumberOfRows** () const
- virtual DoublePtr * **Insert** (double key, DoublePtr *ob, bool withExcept=true)
- virtual DoublePtr * **Insert** (double key, double initialValue=**CTools::m_defaultValueDOUBLE**)
- DoublePtr * **NewMatrix** (double initialValue=**CTools::m_defaultValueDOUBLE**)
- virtual DoublePtr * **operator[]** (double key)
- virtual void **RemoveAll** ()
- bool **RenameKey** (double oldKey, double newKey)
- void **SetDelete** (bool value)
- void **SetMatrixDims** (const CUIntArray &matrixDims)
- virtual ~**CDoublePtrDoubleMap** ()
CDoublePtrDoubleMap (p. 194) dtor.

Protected Member Functions

- void **Delete** (DoublePtr *matrix)

Protected Attributes

- bool **m_bDelete**
- CUIntArray **m_matrixDims**

8.40.1 Detailed Description

a set of a non rectangular matrix of double management classes.

Version

1.0

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

- List.h
- List.cpp

8.41 bratl::CExpressionValue Class Reference

```
#include <Expression.h>
```

Inherits bratl::CBratObject.

Public Member Functions

- std::string **AsString** (const CUnit &Unit=CUnit(""), const std::string Format="", bool dateAsPeriod=false) const
- **CExpressionValue** (double FloatValue=**CTools::m_defaultValueDOUBLE**)
- **CExpressionValue** (const std::vector< double > &FloatValues)
- **CExpressionValue** (const std::string &StrValue)
- **CExpressionValue** (ExpressionValueType Type, ExpressionValueDimensions &Dimensions, double *Value, bool MakeCopy=true)
- **CExpressionValue** (ExpressionValueType type, ExpressionValueDimensions &dimensions, const C↵DoubleArray &value)
- **CExpressionValue** (const **CExpressionValue** &Copy)
- **CExpressionValue** (ExpressionCallableFunction1 &Function, bool IsNumeric, **CExpressionValue** &Parameter1)
- **CExpressionValue** (ExpressionCallableFunctionStrToStr1 &Function, **CExpressionValue** &Parameter1)
- **CExpressionValue** (ExpressionCallableFunctionStrToFlt1 &Function, **CExpressionValue** &Parameter1)
- **CExpressionValue** (ExpressionCallableFunction2 &Function, bool IsNumeric, **CExpressionValue** &Parameter1, **CExpressionValue** &Parameter2)
- **CExpressionValue** (ExpressionCallableFunction3 &Function, bool IsNumeric, **CExpressionValue** &Parameter1, **CExpressionValue** &Parameter2, **CExpressionValue** &Parameter3)
- **CExpressionValue** (ExpressionCallableFunctionAlgoN &function, const char *functionName, CVectorBrat↵AlgorithmParam &arg)
- **CExpressionValue** (ExpressionCallableFunctionBratAlgoBaseN &function, **CBratAlgorithmBase** *algo, CVectorBratAlgorithmParam &arg)
- double **Compare** (**CExpressionValue** &WithWhat)
- void **DeleteValue** ()
- void **Dump** (std::ostream &fOut=std::cerr)

- const ExpressionValueDimensions & **GetDimensions** () const
- std::string **GetDimensionsAsString** ()
- std::string **GetName** ()
- size_t **GetNbDimensions** () const
- size_t **GetNbValues** () const
- std::string **GetString** () const
- ExpressionValueType **GetType** () const
- double **GetValue** (uint32_t index) const
- double **GetValue** (uint32_t i, uint32_t j) const
- double * **GetValues** () const
- bool **HasValue** ()
- int32_t **IsTrue** ()
- **CExpressionValue** & **operator=** (const **CExpressionValue** &Copy)
- **CExpressionValue** & **operator=** (const std::string &String)
- **CExpressionValue** & **operator=** (double value)
- **CExpressionValue** & **operator=** (const std::vector< double > &Vector)
- void **Set** (const **CExpressionValue** &Copy)
- void **SetName** (const std::string &value)
- void **SetNewValue** (ExpressionValueType type, uint32_t *dims, uint32_t nbDims, double *value, bool makeCopy=true)
- void **SetNewValue** (ExpressionValueType Type, ExpressionValueDimensions &Dimensions, double *Value, bool MakeCopy=true)
- void **SetNewValue** (CDoubleArray &vect, bool makeCopy=true)
- void **SetNewValue** (CObDoubleMap &mp, bool makeCopy=true)
- void **SetNewValue** (CDoublePtrDoubleMap &mp, bool makeCopy=true)
- void **SetNewValue** (double *dataValue, uint32_t nbValues, bool makeCopy=true)

Static Public Member Functions

- static **CExpressionValue** * **GetExpressionValue** (CBratObject *ob, bool withExcept=true)

8.41.1 Detailed Description

Expression management classes.

Version

1.0

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

- Expression.h
- Expression.cpp

8.42 bratl::CExternalFilesAvisoGrid Class Reference

```
#include <ExternalFilesAvisoGrid.h>
```

Inherits bratl::CExternalFilesNetCDFCF.

Inherited by bratl::CExternalFilesDotGrid, and bratl::CExternalFilesMercatorDotGrid.

Public Member Functions

- **CExternalFilesAvisoGrid** (const std::string &Name="")
- virtual void **GetValue** (const std::string &Name, **CExpressionValue** &Value, const std::string &WantedUnit)
- virtual void **GetValue** (const std::string &name, double &value, const std::string &wantedUnit)
- virtual bool **NextRecord** ()
- virtual bool **PrevRecord** ()
- virtual void **Rewind** ()

Static Public Member Functions

- static std::string **TypeOf** ()

Static Public Attributes

- static const std::string **m_INTERNAL_DEPTH_DIM_NAME** = "GridDepth"
- static const std::string **m_INTERNAL_LAT_DIM_NAME** = "NbLatitudes"
- static const std::string **m_INTERNAL_LATLON_DIM_NAME** = "LatLon"
- static const std::string **m_INTERNAL_LON_DIM_NAME** = "NbLongitudes"
- static const std::string **m_LAT_DIM_NAME** = "Latitude"
- static const std::string **m_LATLONMIN_NAME** = "LatLonMin"
- static const std::string **m_LATLONSTEP_NAME** = "LatLonStep"
- static const std::string **m_LON_DIM_NAME** = "Longitude"

Protected Member Functions

- virtual void **AddBratIndexData** ()
- virtual void **AddVar** (int32_t NetcdfId, const std::string &Name, const std::string &Description, const std::string &Unit, int32_t type=NC_NAT, const CUIntArray *dimValues=NULL, const CStringArray *dimNames=NULL, const CIntArray *dimIds=NULL, const **CStringMap** *mapAttributes=NULL)
- virtual void **AddVar** (const std::string &Name)
- virtual void **AddVar** (int32_t netcdfId, const std::string &name, const std::string &description, const std::string &unit, int32_t type, uint32_t dimValue, const std::string dimName, int32_t dimId, const **CStringMap** *mapAttributes=NULL)
- void **AddVirtualVariables** ()
- void **CheckNetCDFDimensions** ()
- virtual void **CheckVariables** ()
- uint32_t **CurrentMeasure** () const
- virtual void **FreeResources** ()
- virtual void **GetLatitudes** (double Min, double Step, uint32_t Count, double *Vector)
- virtual void **GetLongitudes** (double Min, double Step, uint32_t Count, double *Vector)
- void **Init** ()
- virtual void **LoadStructure** ()
- virtual void **SubstituteDimNames** (CStringArray &dimNames)

Protected Attributes

- CNetCDFDimension * **m_depthDim**
- uint32_t **m_depthIndex**
- CNetCDFDimension * **m_latDim**
- uint32_t **m_latIndex**
- CNetCDFDimension * **m_lonDim**
- uint32_t **m_lonIndex**
- uint32_t **m_nbDepths**
- uint32_t **m_nbLatitudes**
- uint32_t **m_nbLongitudes**

8.42.1 Detailed Description

External files access.

Version

1.0

8.42.2 Member Function Documentation

8.42.2.1 void bratl::CExternalFilesAvisoGrid::LoadStructure () [protected],[virtual]

Array of the global dimension's index

Implements **bratl::CExternalFilesNetCDF** (p. 200).

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

- ExternalFilesAvisoGrid.h
- ExternalFilesAvisoGrid.cpp

8.43 bratl::CExternalFilesJason2 Class Reference

```
#include <ExternalFilesJason2.h>
```

Inherits bratl::CExternalFilesNetCDFCF.

Inherited by bratl::CExternalFilesJason2GDR, bratl::CExternalFilesJason2SGDR, and bratl::CExternalFilesJason2SSHA.

Public Member Functions

- **CExternalFilesJason2** (const std::string &name="")

Static Public Member Functions

- static std::string **TypeOf** ()

Static Public Attributes

- static const std::string **smMissionName** = **CTools::StringToUpper**("Jason-2")

8.43.1 Detailed Description

Jason-2 files access.

Version

1.0

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

- ExternalFilesJason2.h
- ExternalFilesJason2.cpp

8.44 brathl::CExternalFilesNetCDF Class Reference

```
#include <ExternalFilesNetCDF.h>
```

Inherits brathl::CExternalFiles.

Inherited by brathl::CExternalFilesNetCDFCF.

Public Member Functions

- virtual void **AddAttributesAsField** (CFieldNetCdf *field=NULL)
- virtual void **AddOffset** (double value, bool force=false)
- **CExternalFilesNetCDF** (const std::string &Name="")
- virtual void **Close** ()
- void **ExecuteExpression** (CExpression &expr, CExpressionValue &exprValue, const std::string &wantedUnit, CProduct *product=NULL)
- virtual CFieldNetCdf * **FindCycleField** ()
- virtual CFieldNetCdf * **FindLatField** ()
- virtual CFieldNetCdf * **FindLonField** ()
- virtual CFieldNetCdf * **FindPassField** ()
- virtual CFieldNetCdf * **FindTimeField** ()
- virtual void **GetAllValues** (const std::string &name, CExpressionValue &value, const std::string &wantedUnit)
- virtual void **GetAllValues** (const std::string &name, CDoubleArray &vect, const std::string &wantedUnit)
- virtual void **GetAllValues** (CFieldNetCdf *field, CExpressionValue &value, const std::string &wantedUnit)
- virtual void **GetAllValues** (CFieldNetCdf *field, const std::string &wantedUnit)
- int **GetAttribute** (const std::string &varName, const std::string &attName, double &attValue, bool mustExist=true, double defaultValue=CTools::m_defaultValueDOUBLE)
- int **GetAttribute** (const std::string &varName, const std::string &attName, std::string &attValue, bool mustExist=true, std::string defaultValue="")
- nc_type **GetAttributeType** (const std::string &attName)
- nc_type **GetAttributeType** (const std::string &varName, const std::string &attName)
- virtual void **GetDimensions** (const std::string &varName, CUIntArray &dimensions)
- virtual void **GetDimensions** (const std::string &varName, CStringArray &dimensions)
- CIntMap & **GetDimIds** ()
- CUIntMap & **GetDimValues** ()
- virtual void **GetFieldNames** (CStringArray &names)
- CFieldNetCdf * **GetFieldNetCdf** (const std::string &name, bool withExcept=true)
- virtual CObMap * **GetFields** ()
- CNetCDFFiles * **GetFile** ()
- int **GetGlobalAttribute** (const std::string &attName, double &attValue, bool mustExist=true, double defaultValue=CTools::m_defaultValueDOUBLE)
- int **GetGlobalAttribute** (const std::string &attName, std::string &attValue, bool mustExist=true, std::string defaultValue="")
- void **GetGlobalAttributes** (CStringMap &mapAttributes)
- void **GetGlobalAttributes** (CDoubleMap &mapAttributes)
- void **GetGlobalAttributes** (std::string &attributes)
- virtual std::string **GetName** () const
- int32_t **GetNetCdfId** (const std::string &name, bool withExcept=true)
- void **GetOrderedDimNames** (const std::string &value, CStringArray &commonDimensionNames)
- void **GetOrderedDimNames** (const CExpression &value, CStringArray &commonDimensionNames)
- void **GetOrderedDimNames** (const CStringArray *fieldNames, CStringArray &commonDimensionNames)
- void **GetOrderedDimNamesFromFieldNetcdf** (const CStringArray *fieldNames, CStringArray &commonDimensionNames)
- virtual void **GetValue** (const std::string &name, CExpressionValue &value, const std::string &wantedUnit)
- virtual void **GetValue** (const std::string &name, double &value, const std::string &wantedUnit)

- virtual void **GetValues** (const std::string &name, **CExpressionValue** &value, const std::string &wantedUnit)
- virtual void **GetValues** (**CFieldNetCdf** *field, **CExpressionValue** &value, const std::string &wantedUnit)
- **CFieldNetCdf** * **GetVarByAttribute** (const std::string &attrName, const std::string &attrValueToSearch)
- virtual void **GetVariables** (CStringArray &varNames)
- nc_type **GetVarType** (const std::string &name)
- virtual std::string **GetVarTypeName** (const std::string &name)
- virtual bool **IsAxisVar** (const std::string &name)
- bool **IsLatField** (**CFieldNetCdf** *field)
- bool **IsLonField** (**CFieldNetCdf** *field)
- virtual bool **IsOpened** () const
- virtual int32_t **NumberOfRecords** ()
- virtual void **Open** ()
- virtual void **SetMode** (**bratl_FileMode** mode)
- virtual void **SetName** (const std::string &Name)
- virtual void **SetOffset** (double value, bool force=false)
- virtual bool **VarExists** (const std::string &name)

Static Public Member Functions

- static std::string **TypeOf** ()

Protected Member Functions

- virtual void **AddBratIndexData** ()
- virtual void **AddVar** (int32_t NetcdfId, const std::string &Name, const std::string &Description, const std::string &Unit, int32_t type=NC_NAT, const CUIntArray *dimValues=NULL, const CStringArray *dimNames=NULL, const CIntArray *dimIds=NULL, const **CStringMap** *mapAttributes=NULL)
- virtual void **AddVar** (int32_t netcdfId, const std::string &name, const std::string &description, const std::string &unit, int32_t type, uint32_t dimValue, const std::string dimName, int32_t dimId, const **CStringMap** *mapAttributes=NULL)
- virtual void **AddVar** (const std::string &Name)
- virtual void **CheckDimensions** ()
- virtual void **CheckVariables** ()
- virtual void **FreeResources** ()
- virtual void **LoadStructure** ()=0
- void **SetOffset** (bool force=false)
- virtual void **SubstituteDimNames** (CStringArray &dimNames)

Protected Attributes

- **CIntMap** m_dimIds
- **CUIntMap** m_dimValues
- **CNetCDFFiles** m_file
- **uint32_t** m_nbMeasures
- **CObMap** m_varList

8.44.1 Detailed Description

External NetCdf files access.

Version

1.0

8.44.2 Member Function Documentation

8.44.2.1 virtual void brathl::CExternalFilesNetCDF::LoadStructure () [protected],[pure virtual]

Array of the global dimension's index

Implemented in **brathl::CExternalFilesAvisoGrid** (p. 198).

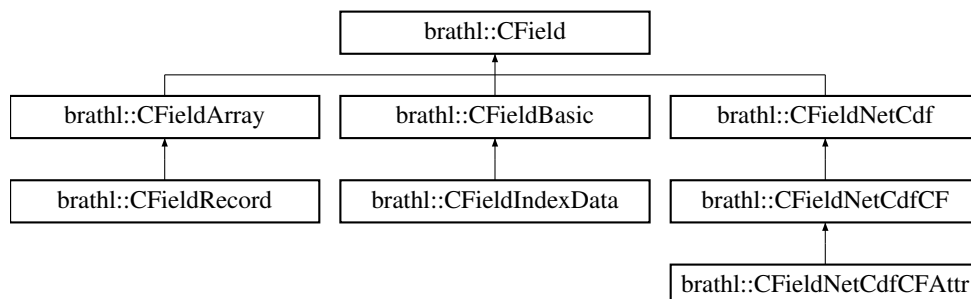
The documentation for this class was generated from the following files:

- ExternalFilesNetCDF.h
- ExternalFilesNetCDF.cpp

8.45 brathl::CField Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CField:



Classes

- class **CListField**

Public Member Functions

- void **AddFieldIndexes** (CFieldIndex *value)
- void **AddFieldIndexes** (CObArray *vect, bool removeAll=true)
- virtual void **AddOffset** (double value)
- virtual void **AdjustValidMinMax** (double *data, int32_t size)
- virtual void **AdjustValidMinMax** (double value)
- **CField** ()
Ctor.
- **CField** (const std::string &name, const std::string &description="", const std::string &unit="")
- **CField** (const **CField** &f)
- void **Convert** (double *data, int32_t size)
- void **ConvertDefaultValueFloat** (double *data, int32_t size)
- void **ConvertDefaultValueInt16** (double *data, int32_t size)
- void **ConvertDefaultValueInt32** (double *data, int32_t size)
- void **ConvertDefaultValueInt64** (double *data, int32_t size)
- void **ConvertDefaultValueInt8** (double *data, int32_t size)
- void **ConvertDefaultValueUInt16** (double *data, int32_t size)
- void **ConvertDefaultValueUInt32** (double *data, int32_t size)
- void **ConvertDefaultValueUInt64** (double *data, int32_t size)
- void **ConvertDefaultValueUInt8** (double *data, int32_t size)

- virtual **CFieldSet** * **CreateFieldSet** (const **CField::CListField** &listFields)=0
- void **DeleteFieldIndexes** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
 - Dump fonction.*
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- bool **End** ()
- bool **GetConvertDate** ()
- int32_t **GetCurrentPos** ()
- coda_Cursor * **GetCursor** ()
- const **CDate** & **GetDateRef** ()
- const std::string & **GetDescription** () const
- long * **GetDim** ()
- virtual std::string **GetDimAsString** ()
- void **GetDimAsVector** (CUIntArray &dim)
- long **GetDimAt** (int32_t index)
- bool **GetExpandArray** ()
- **CObArray** * **GetFieldIndexes** ()
- virtual std::string **GetFullName** () const
- virtual std::string **GetFullNameWithRecord** ()
- virtual bool **GetHidden** ()
- virtual bool **GetHighResolution** ()
- int32_t **GetIndex** ()
- const std::string & **GetKey** () const
- int **GetMaxPos** ()
- const std::string & **GetName** () const
- coda_native_type **GetNativeType** ()
- virtual std::string **GetNativeTypeName** ()
- int32_t **GetNbDims** ()
- int **GetNbElts** ()
- virtual uint32_t **GetNumHighResolutionMeasure** ()
- double **GetOffset** ()
- virtual uint32_t **GetOffsetDim** ()
- virtual std::string **GetRecordName** ()
- coda_special_type **GetSpecialType** ()
- virtual std::string **GetSpecialTypeName** ()
- coda_type_class **GetTypeClass** ()
- int32_t **GetUnion** ()
- const std::string & **GetUnit** () const
- double **GetValidMax** ()
- double **GetValidMin** ()
- virtual int32_t **GetVirtualNbDims** ()
- void **HandleBratError** (const std::string &str="", int32_t errClass=BRATHL_LOGIC_ERROR)
- bool **HasDim** ()
- bool **HasEqualDims** (**CField** *field)
- virtual bool **HasVirtualNbDims** ()
- bool **HasXDim** ()
- bool **HasYDim** ()
- virtual bool **IsDimTransposed** ()
- bool **IsExpandArray** ()
- bool **IsFieldHasDefaultValue** ()
- bool **IsFieldNetCdfCFAAttr** ()
- bool **IsFixedSize** () const
- bool **IsGoToAvailableUnionField** ()
- virtual bool **IsHidden** ()

- virtual bool **IsHighResolution** ()
- bool **IsMetaData** ()
- virtual bool **IsSpecialType** ()
- bool **IsToBeRemoved** ()
- bool **IsUnion** ()
- virtual bool **IsVirtual** () const
- bool **LastRecord** ()
- **CField** & **operator=** (const **CField** &f)
- virtual void **PopCursor** ()=0
- void **PopRecordCusor** (**CObList** *parentFieldList)
- virtual void **PushPos** ()=0
- virtual void **Read** (CDoubleArray &vect, bool skip=false)
- virtual void **Read** (double *data, bool skip=false)
- virtual void **Read** (std::string &value, bool skip=false)
- virtual void **ReadParent** (CDoubleArray &vect, **CFieldRecord** *parentField)
- virtual void **ReadParent** (CDoubleArray &vect, **CObList** *parentFieldList)
- void **Set** (const **CField** &f)
- void **SetConvertDate** (bool value)
- void **SetCurrentPos** (int32_t currentPos)
- void **SetCurrentPosToLast** ()
- void **SetCursor** (coda_Cursor &cursor)
- void **SetDateRef** (**brathl_refDate** refDate)
- void **SetDateRef** (const **CDate** &value)
- void **SetDefaultValue** (double *data, int32_t size)
- void **SetDescription** (const std::string &description)
- void **SetDim** (int32_t nbDims, const long dim[])
- void **SetDim** (int32_t nbDims, const CUIntArray &dim)
- void **SetDim** (const CUIntArray &dim)
- void **SetDim** (const CUIntArray *dim)
- void **SetDim** (int32_t nbElts)
- void **SetExpandArray** (bool value)
- void **SetFieldHasDefaultValue** (bool value)
- void **SetFixedSize** (bool isFixedSize)
- void **SetGoToAvailableUnionField** (bool value)
- virtual void **SetHidden** (bool value)
- virtual void **SetHighResolution** (bool value)
- void **SetIndex** (int32_t index)
- void **SetKey** (const std::string &key)
- void **SetMetaData** (bool metaData)
- void **SetName** (const std::string &name)
- void **SetNativeType** (coda_native_type nativeType)
- virtual void **SetNumHighResolutionMeasure** (uint32_t value)
- virtual void **SetOffset** (double value)
- void **SetSpecialType** (coda_special_type specialType)
- void **SetToBeRemoved** (bool value)
- void **SetTypeClass** (coda_type_class typeClass)
- void **SetUnion** (int32_t value)
- virtual void **SetUnit** (const std::string &unit)
- void **SetValidMax** (double value)
- void **SetValidMin** (double value)
- virtual void **SetVirtual** (bool value)
- bool **TransposeDim** ()
- bool **TransposeValues** (double *data, int32_t size)
- bool **UnitIsDate** ()
- virtual ~**CField** ()

Dtor.

Static Public Member Functions

- static void **AdjustValidMinMax** (double *data, int32_t size, double &min, double &max)
- static void **AdjustValidMinMax** (double value, double &min, double &max)
- static CFieldNetCdfCFAttr * **GetFieldNetCdfCFAttr** (CBratObject *ob, bool withExcept=true)
- static CFieldNetCdfIndexData * **GetFieldNetCdfIndexData** (CBratObject *ob, bool withExcept=true)
- static bool **IsFieldNetCdfCFAttr** (CBratObject *ob)

Static Public Attributes

- static const std::string **m_BRAT_INDEX_DATA_DESC** = "data index"
- static const std::string **m_BRAT_INDEX_DATA_NAME** = "brat_index_data"

Protected Member Functions

- void **Init** ()

Protected Attributes

- bool **m_convertDate**
- int32_t **m_currentPos**
- coda_Cursor **m_cursor**
- CDate **m_dateRef**
- std::string **m_description**
- long **m_dim** [MAX_NUM_DIMS]
- bool **m_dimsTransposed**
- bool **m_expandArray**
- bool **m_fieldHasDefaultValue**
- CObArray * **m_fieldIndexes**
- std::string **m_fullName**
- bool **m_goToAvailableUnionField**
- bool **m_hidden**
- bool **m_highResolution**
- int32_t **m_index**
- bool **m_isFixedSize**
- int32_t **m_isUnion**
- std::string **m_key**
- int32_t **m_maxPos**
- bool **m_metaData**
- std::string **m_name**
- coda_native_type **m_nativeType**
- int32_t **m_nbDims**
- uint32_t **m_numHighResolutionMeasure**
- double **m_offset**
- std::string **m_recordName**
- coda_special_type **m_specialType**
- bool **m_toBeRemoved**
- coda_type_class **m_typeClass**
- std::string **m_unit**
- bool **m_unitIsDate**
- double **m_validMax**
- double **m_validMin**
- bool **m_virtualField**

8.45.1 Detailed Description

Field management base classe.

Version

1.0

8.45.2 Member Data Documentation

8.45.2.1 `long bratl::CField::m_dim[MAX_NUM_DIMS]` `[protected]`

total number of dimensions

8.45.2.2 `bool bratl::CField::m_isFixedSize` `[protected]`

(maximum) dimensions

Referenced by `Dump()`.

8.45.2.3 `double bratl::CField::m_validMax` `[protected]`

Valid max value

8.45.2.4 `double bratl::CField::m_validMin` `[protected]`

Valid min value

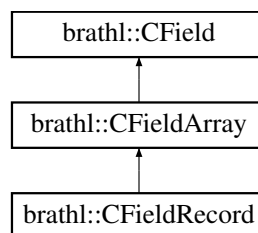
The documentation for this class was generated from the following files:

- `Field.h`
- `Field.cpp`

8.46 bratl::CFieldArray Class Reference

```
#include <Field.h>
```

Inheritance diagram for `bratl::CFieldArray`:



Public Member Functions

- **CFieldArray** ()
Ctor.
- **CFieldArray** (const std::string &name, const std::string &description="", const std::string &unit="")
- **CFieldArray** (int32_t nbDims, const long dim[], const std::string &name, const std::string &description="", const std::string &unit="")
- **CFieldArray** (CFieldArray &f)
- void **CreateFieldIndexes** (CObArray &vect)
- virtual **CFieldSet** * **CreateFieldSet** (const CField::CListField &listFields) override

- virtual void **Dump** (std::ostream &fOut=std::cerr) override
Dump fonction.
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout) override
- virtual uint32_t **GetOffsetDim** () override
- virtual int32_t **GetVirtualNbDims** () override
- const **CFieldArray** & **operator=** (**CFieldArray** &f)
- virtual void **PopCursor** () override
- virtual void **PushPos** () override
- virtual void **PushPos** (int32_t iDim)
- virtual void **Read** (CDoubleArray &vect, bool skip=false) override
- virtual void **Read** (double *data, bool skip=false) override
- void **Set** (**CFieldArray** &f)
- virtual ~**CFieldArray** ()

Dtor.

Additional Inherited Members

8.46.1 Detailed Description

Field of type 'array' management classes.

Version

1.0

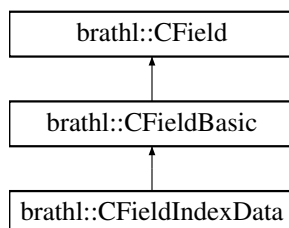
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.47 brathl::CFieldBasic Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldBasic:



Public Member Functions

- **CFieldBasic** ()
Ctor.
- **CFieldBasic** (long length, const std::string &name, const std::string &description, const std::string &unit)
- **CFieldBasic** (**CFieldBasic** &f)
- virtual **CFieldSet** * **CreateFieldSet** (const **CField::CListField** &listFields)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.

- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- const **CFieldBasic** & **operator=** (**CFieldBasic** &f)
- virtual void **PopCursor** ()
- virtual void **PushPos** ()
- virtual void **Read** (CDoubleArray &vect, bool skip=false)
- virtual void **Read** (double *data, bool skip=false)
- virtual void **Read** (std::string &data, bool skip=false)
- void **Set** (**CFieldBasic** &f)
- virtual ~**CFieldBasic** ()

Dtor.

Public Attributes

- long **m_length**

Additional Inherited Members

8.47.1 Detailed Description

Field of type 'basic' management classes.

Version

1.0

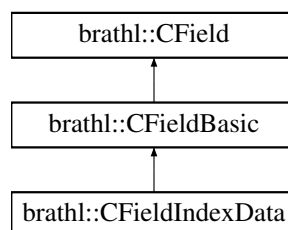
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.48 brathl::CFieldIndexData Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldIndexData:



Public Member Functions

- **CFieldIndexData** ()
Ctor.
- **CFieldIndexData** (const std::string &name, const std::string &description, const std::string &unit="")
- **CFieldIndexData** (**CFieldIndexData** &f)
- virtual **CFieldSet** * **CreateFieldSet** (const **CField::CListField** &listFields)
- virtual void **Dump** (std::ostream &fOut=std::cerr)

Dump fonction.

- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- double **GetValue** ()
- const **CFieldIndexData** & **operator=** (**CFieldIndexData** &f)
- virtual void **PopCursor** ()
- virtual void **PushPos** ()
- virtual void **Read** (CDoubleArray &vect, bool skip=false)
- virtual void **Read** (double *data, bool skip=false)
- virtual void **Read** (std::string &data, bool skip=false)
- virtual void **Read** (double &value)
- virtual double **Read** ()
- void **Set** (**CFieldIndexData** &f)
- virtual ~**CFieldIndexData** ()

Dtor.

Protected Member Functions

- void **Init** ()

Additional Inherited Members

8.48.1 Detailed Description

Field of type 'basic' management classes.

Version

1.0

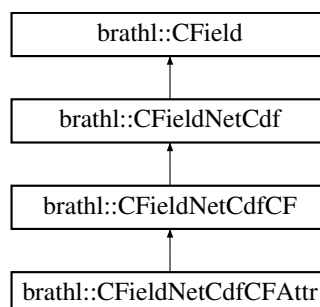
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.49 brathl::CFieldNetCdf Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldNetCdf:



Public Member Functions

- void **AdjustValidMinMaxFromValues** ()
- **CFieldNetCdf** ()
 - Ctor.*
- **CFieldNetCdf** (const std::string &name, const std::string &description="", const std::string &unit="", int32_t netCdfId=NC_GLOBAL, int32_t type=NC_NAT, const CUIntArray *dimValues=NULL, const CStringArray *dimNames=NULL, const CIntArray *dimIds=NULL, const CDoubleArray *values=NULL)
- **CFieldNetCdf** (**CFieldNetCdf** &f)
- virtual CBratObject * **Clone** () override
- **CFieldNetCdf** * **CloneThis** ()
- virtual **CFieldSet** * **CreateFieldSet** (const **CField::CListField** &listFields) override
- virtual **CFieldSet** * **CreateFieldSet** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
 - Dump fonction.*
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout) override
- void **EmptyValues** ()
- double **GetAddOffset** ()
- virtual std::string **GetAttribute** (const std::string attrName)
- const **CStringMap** & **GetAttributes** ()
- int32_t **GetCounFromDimCountArray** ()
- const **CIntArray** & **GetDimIds** ()
- void **GetDimIdsAsArray** (CIntArray &values, bool bRemoveAll=true)
- const CStringArray & **GetDimNames** ()
- uint32_t **GetDimRange** (const std::string &dimName)
- const **CUIntArray** & **GetDimRanges** ()
- uint32_t * **GetDimsCountArray** ()
- uint32_t * **GetDimsIndexArray** ()
- const **CUIntArray** & **GetDimValues** ()
- void **GetDimValuesAsArray** (CUIntArray &values, bool bRemoveAll=true)
- double **GetFillValue** ()
- virtual std::string **GetFullName** () const override
- virtual std::string **GetFullNameWithRecord** () override
- virtual std::string **GetMostExplicitName** ()
- int32_t **GetNativeType** ()
- virtual std::string **GetNativeTypeName** () override
- int32_t **GetNetCdfId** ()
- CUnit * **GetNetCdfUnit** ()
- int32_t **GetPosFromDimIndexArray** ()
- virtual std::string **GetRecordName** () override
- double **GetScaleFactor** ()
- int32_t **GetSpecialType** ()
- virtual std::string **GetSpecialTypeName** () override
- int32_t **GetType** ()
- virtual std::string **GetTypeName** ()
- virtual CDoubleArray & **GetValues** ()
- double * **GetValuesAsArray** ()
- virtual CDoubleArray & **GetValuesWithUnitConversion** (const std::string &wantedUnit)
- virtual int32_t **GetVirtualNbDims** () override
- virtual void **InitDimIndexes** (uint32_t value)
- virtual void **InitDimsIndexToMax** ()
- virtual void **InitDimsIndexToMax** (uint32_t index)
- bool **IsAtBeginning** ()
- virtual bool **IsSpecialType** () override
- uint32_t * **NewDimIndexArray** (**CFieldNetCdf** *fromField=NULL)

- bool **NextIndex** ()
- const **CFieldNetCdf** & **operator=** (**CFieldNetCdf** &f)
- virtual void **PopCursor** () override
- bool **PrevIndex** ()
- virtual void **PushPos** () override
- virtual void **Read** (CDoubleArray &vect, bool skip=false) override
- virtual void **Read** (**CExpressionValue** &value, bool skip=false)
- NetCDFVarKind **SearchDimKind** ()
- void **Set** (**CFieldNetCdf** &f)
- void **SetAddOffset** (double value)
- void **SetAtBeginning** (bool value)
- virtual void **SetAttributes** (const **CStringMap** &mapAttributes)
- virtual void **SetAttributes** (const **CStringMap** *mapAttributes)
- void **SetDimIds** (const **CIntMap** &dimIds)
- void **SetDimIds** (const **CIntMap** *dimIds)
- virtual void **SetDimInfo** (const CStringArray &dimNames, const CIntArray &dimIds, const CUIntArray &dim↵
Values)
- virtual void **SetDimInfo** (const CStringArray *dimNames, const CIntArray *dimIds, const CUIntArray *dim↵
Values)
- virtual void **SetDimNames** (const CStringArray &dimNames)
- virtual void **SetDimNames** (const CStringArray *dimNames)
- virtual void **SetDimValues** (const **CUIntMap** &dimValues)
- virtual void **SetDimValues** (const **CUIntMap** *dimValues)
- void **SetFillValue** (double value)
- virtual void **SetIndex** (const std::string &dimName, uint32_t index, uint32_t count)
- void **SetNativeType** (int32_t type)
- void **SetNetCdfId** (int32_t id)
- void **SetScaleFactor** (double value)
- virtual void **SetType** (int32_t type)
- virtual void **SetUnit** (const std::string &unit) override
- virtual void **SetUnit** (const CUnit &unit)
- virtual void **SetValues** (double values)
- virtual void **SetValues** (double *values, size_t length)
- virtual void **SetValues** (const CDoubleArray &values)
- virtual void **SetValues** (const CDoubleArray *values)
- virtual void **SetValues** (const CInt16Array &values)
- virtual void **SetValues** (const CInt16Array *values)
- virtual void **SetValues** (const CInt8Array &values)
- virtual void **SetValues** (const CInt8Array *values)
- virtual void **SetValues** (const CIntArray &values)
- virtual void **SetValues** (const CIntArray *values)
- virtual void **SetValues** (const CUInt8Array &values)
- virtual void **SetValues** (const CUInt8Array *values)
- virtual void **SetValues** (const CFloatArray &values)
- virtual void **SetValues** (const CFloatArray *values)
- virtual void **SetValues** (const std::string &values)
- void **SetValuesAsArray** ()
- void **SetValuesAsArray** (const CDoubleArray &values)
- void **SetValuesAsArray** (const CDoubleArray *values)
- virtual ~**CFieldNetCdf** ()

Dtor.

Protected Member Functions

- void **DeleteDimIndexArray** ()
- void **DeleteValuesAsArray** ()
- void **Init** ()

Protected Attributes

- double **m_addOffset**
- bool **m_atBeginning**
- **CIntMap** **m_dimIds**
- **CStringArray** **m_dimNames**
- **CUIntMap** **m_dimRanges**
- **uint32_t** * **m_dimsCountArray**
- **uint32_t** * **m_dimsIndexArray**
- **CUIntMap** **m_dimValues**
- double **m_fillValue**
- **CStringMap** **m_mapAttributes**
- **int32_t** **m_netCdfId**
- **CUnit** **m_netCdfUnit**
- double **m_scaleFactor**
- **int32_t** **m_type**
- **CDoubleArray** **m_values**
- double * **m_valuesAsArray**
- **CDoubleArray** **m_valuesWithUnitConversion**

Additional Inherited Members

8.49.1 Detailed Description

Field from a NetCdf file management classes.

Version

1.0

8.49.2 Member Data Documentation

8.49.2.1 **double** **brathl::CFieldNetCdf::m_addOffset** [protected]

data add offset

Referenced by Dump().

8.49.2.2 **bool** **brathl::CFieldNetCdf::m_atBeginning** [protected]

'At beginning" flag

Referenced by Dump().

8.49.2.3 **CIntMap** **brathl::CFieldNetCdf::m_dimIds** [protected]

Map of the dimension's ids of the field (key is dim. name)

Referenced by Dump().

8.49.2.4 CStdStringArray brathl::CFieldNetCdf::m_dimNames [protected]

Array of the dimension's names of the field (index is dim. range)

Referenced by Dump().

8.49.2.5 CUIntMap brathl::CFieldNetCdf::m_dimRanges [protected]

Map of the dimension's range of the field (key is dim. name)

Referenced by Dump().

8.49.2.6 uint32_t* brathl::CFieldNetCdf::m_dimsCountArray [protected]

Array of the dimension count for reading

Referenced by Dump().

8.49.2.7 uint32_t* brathl::CFieldNetCdf::m_dimsIndexArray [protected]

Array of the dimension's index

Referenced by Dump().

8.49.2.8 CUIntMap brathl::CFieldNetCdf::m_dimValues [protected]

Map of the dimension's values of the field (key is dim. name)

Referenced by Dump().

8.49.2.9 double brathl::CFieldNetCdf::m_fillValue [protected]

data default value (fill value)

Referenced by Dump().

8.49.2.10 CStdStringMap brathl::CFieldNetCdf::m_mapAttributes [protected]

Map of the netcdf attributes (as std::string representation).

Referenced by Dump().

8.49.2.11 int32_t brathl::CFieldNetCdf::m_netCdfId [protected]

The netcdf external id

Referenced by Dump().

8.49.2.12 CUnit brathl::CFieldNetCdf::m_netCdfUnit [protected]

The netcdf unit

Referenced by Dump().

8.49.2.13 double brathl::CFieldNetCdf::m_scaleFactor [protected]

data scale factor

Referenced by Dump().

8.49.2.14 int32_t brathl::CFieldNetCdf::m_type [protected]

The netcdf external data types

Referenced by Dump().

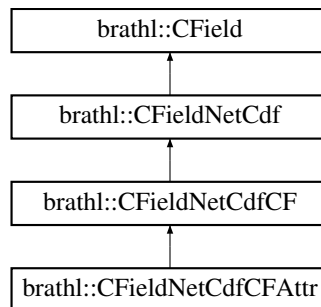
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.50 brathl::CFieldNetCdfCF Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldNetCdfCF:



Public Member Functions

- **CFieldNetCdfCF** ()
Ctor.
- **CFieldNetCdfCF** (const std::string &name, const std::string &description="", const std::string &unit="", int32_t netCdfId=NC_GLOBAL, int32_t type=NC_NAT, const CUIntArray *dimValues=NULL, const CStringArray *dimNames=NULL, const CIntArray *dimIds=NULL, const CDoubleArray *values=NULL)
- **CFieldNetCdfCF** (**CFieldNetCdfCF** &f)
- virtual CBratObject * **Clone** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- virtual std::string **GetDimAsString** ()
- std::string **GetDimAsStringWithIndexes** ()
- std::string **GetDimAsStringWithNames** ()
- const **CFieldNetCdfCF** & **operator=** (**CFieldNetCdfCF** &f)
- void **Set** (**CFieldNetCdfCF** &f)
- virtual ~**CFieldNetCdfCF** ()
Dtor.

Protected Member Functions

- void **Init** ()

Additional Inherited Members

8.50.1 Detailed Description

Field from a NetCdf file management classes.

Version

1.0

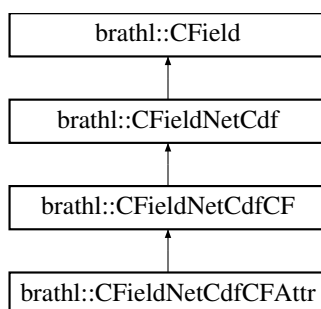
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.51 brathl::CFieldNetCdfCFAAttr Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldNetCdfCFAAttr:



Public Member Functions

- **CFieldNetCdfCFAAttr** ()
Ctor.
- **CFieldNetCdfCFAAttr** (CNetCDFVarDef *netCDFVarDef, CNetCDFAttr *netCDFAttr)
- **CFieldNetCdfCFAAttr** (CNetCDFAttr *netCDFAttr)
- **CFieldNetCdfCFAAttr** (CFieldNetCdfCFAAttr &f)
- virtual CBratObject * **Clone** ()
- **CFieldNetCdfCFAAttr** * **CloneThis** ()
- void **DeleteNetCDFAttr** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- virtual std::string **GetMostExplicitName** ()
- CNetCDFAttr * **GetNetCDFAttr** ()
- const std::string & **GetRelatedVarName** ()
- bool **IsFieldNetCdfCFAAttrGlobal** ()
- bool **IsFieldNetCdfCFAAttrVariable** ()
- const **CFieldNetCdfCFAAttr** & **operator=** (CFieldNetCdfCFAAttr &f)
- void **Set** (CFieldNetCdfCFAAttr &f)
- virtual void **SetAttributes** (const CStringMap &mapAttributes)
- virtual void **SetAttributes** (const CStringMap *mapAttributes)
- void **SetInfoFromAttr** (CNetCDFVarDef *netCDFVarDef=NULL)
- void **SetInfoFromAttr** (CNetCDFAttr *netCDFAttr, CNetCDFVarDef *netCDFVarDef=NULL)
- void **SetNetCDFAttr** (CNetCDFAttr *value)
- void **SetRelatedVarName** (const std::string &value)
- virtual void **SetType** (int32_t type)
- void **SetValuesFromAttr** ()
- void **SetValuesFromAttr** (CNetCDFAttr *netCDFAttr)
- virtual ~**CFieldNetCdfCFAAttr** ()
Dtor.

Static Public Member Functions

- static bool **IsFieldNetCdfCFAAttrGlobal** (CBratObject *ob)
- static bool **IsFieldNetCdfCFAAttrVariable** (CBratObject *ob)

Protected Member Functions

- void **Init** ()

Protected Attributes

- CNetCDFAttr * **m_netCDFAttr**
- std::string **m_relatedVarName**

Additional Inherited Members

8.51.1 Detailed Description

Field from a NetCdf Attribute file management classes.

Version

1.0

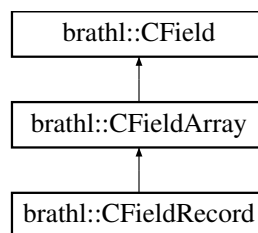
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.52 bratl::CFieldRecord Class Reference

```
#include <Field.h>
```

Inheritance diagram for bratl::CFieldRecord:



Public Member Functions

- **CFieldRecord** ()
Ctor.
- **CFieldRecord** (size_t nbFields, const std::string &name, const std::string &description="", const std::string &unit="")
- **CFieldRecord** (int32_t nbDims, const long dim[], size_t nbFields, const std::string &name, const std::string &description="", const std::string &unit="")
- **CFieldRecord** (CFieldRecord &f)
- virtual **CFieldSet** * **CreateFieldSet** (const CField::CListField &listFields)

- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- size_t **GetNbFields** ()
- virtual int32_t **GetVirtualNbDims** ()
- const **CFieldRecord** & **operator=** (**CFieldRecord** &f)
- virtual void **PopCursor** ()
- virtual void **PushPos** ()
- virtual void **PushPos** (int32_t iDim)
- virtual void **Read** (CDoubleArray &vect, bool skip=false)
- virtual void **Read** (double *data, bool skip=false)
- void **Set** (**CFieldRecord** &f)
- void **SetNbFields** (size_t value)
- virtual ~**CFieldRecord** ()
Dtor.

Protected Attributes

- size_t **m_nbFields**

Additional Inherited Members

8.52.1 Detailed Description

Field of type 'record' management classes.

Version

1.0

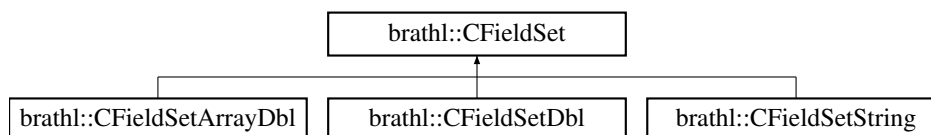
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.53 bratl::CFieldSet Class Reference

```
#include <Field.h>
```

Inheritance diagram for bratl::CFieldSet:



Public Member Functions

- **CFieldSet** (const std::string &name="")
Ctor.
- **CFieldSet** (**CFieldSet** &f)
- virtual void **Dump** (std::ostream &fOut=std::cerr)

Dump fonction.

- virtual **CField** * **GetField** ()
- const std::string & **GetName** ()
- virtual void **Insert** (const CDoubleArray &vect, bool bRemove=false)=0
- virtual void **Insert** (double value, bool bRemove=false)=0
- virtual void **Insert** (const std::string &value, bool bRemove=false)=0
- **CFieldSet** & **operator=** (**CFieldSet** &o)
- virtual void **SetField** (**CField** *value)
- virtual ~**CFieldSet** ()

Dtor.

Protected Member Functions

- void **Copy** (**CFieldSet** &f)

Protected Attributes

- **CField** * **m_field**
- std::string **m_name**

8.53.1 Detailed Description

a base class for set of field value.

Version

1.0

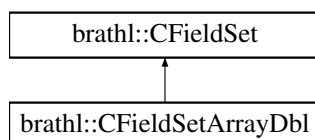
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.54 brathl::CFieldSetArrayDbI Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldSetArrayDbI:



Public Member Functions

- **CFieldSetArrayDbI** (const std::string &name="")
Ctor.
- **CFieldSetArrayDbI** (**CFieldSetArrayDbI** &f)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- CDoubleArray & **GetDataVector** ()

- virtual void **Insert** (const CDoubleArray &vect, bool bRemove=false)
- virtual void **Insert** (double value, bool bRemove=false)
- virtual void **Insert** (const std::string &value, bool bRemove=false)
- **CFieldSetArrayDbI** & **operator=** (**CFieldSetArrayDbI** &o)
- virtual ~**CFieldSetArrayDbI** ()

Dtor.

Public Attributes

- CUIntArray **m_dim**
- int32_t **m_nbDims**
- CDoubleArray **m_vector**

Protected Member Functions

- void **Copy** (**CFieldSetArrayDbI** &f)

Additional Inherited Members

8.54.1 Detailed Description

a set of double array field value.

Version

1.0

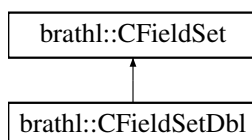
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.55 brathl::CFieldSetDbI Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldSetDbI:



Public Member Functions

- int32_t **AsInt32** ()
 - int32_t **AsUInt32** ()
 - **CFieldSetDbI** (const std::string &name="")
- Ctor.*
- **CFieldSetDbI** (**CFieldSetDbI** &f)
 - virtual void **Dump** (std::ostream &fOut=std::cerr)
- Dump fonction.*

- double **GetData** ()
- double & **GetDataRef** ()
- virtual void **Insert** (const CDoubleArray &vect, bool bRemove=false)
- virtual void **Insert** (double value, bool bRemove=false)
- virtual void **Insert** (const std::string &value, bool bRemove=false)
- **CFieldSetDbI** & **operator=** (**CFieldSetDbI** &o)
- void **SetData** (double value)
- virtual ~**CFieldSetDbI** ()

Dtor.

Public Attributes

- double **m_value**

Protected Member Functions

- void **Copy** (**CFieldSetDbI** &f)

Additional Inherited Members

8.55.1 Detailed Description

a set of double field value.

Version

1.0

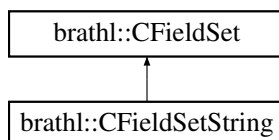
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.56 brathl::CFieldSetString Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CFieldSetString:



Public Member Functions

- **CFieldSetString** (const std::string &name="")
Ctor.
- **CFieldSetString** (**CFieldSetString** &f)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- std::string **GetData** ()

- `std::string & GetDataRef ()`
- virtual void **Insert** (const CDoubleArray &vect, bool bRemove=false)
- virtual void **Insert** (double value, bool bRemove=false)
- virtual void **Insert** (const std::string &value, bool bRemove=false)
- **CFieldSetString & operator= (CFieldSetString &o)**
- void **SetData** (const std::string &value)
- virtual **~CFieldSetString ()**

Dtor.

Public Attributes

- `std::string m_value`

Protected Member Functions

- void **Copy** (CFieldSetString &f)

Additional Inherited Members

8.56.1 Detailed Description

a set of `std::string` field value.

Version

1.0

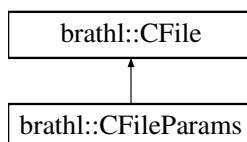
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.57 brathl::CFile Class Reference

```
#include <File.h>
```

Inheritance diagram for `brathl::CFile`:



Public Types

- enum **openFlags** {
modeRead = 0x0001, **modeWrite** = 0x0002, **modeAppend** = 0x0004, **modeReadWrite** = 0x0008,
modeRWCreate = 0x0010, **modeReadAppend** = 0x0020, **typeText** = 0x4000, **typeBinary** = static_cast<int32_t>(0x8000) }

Public Member Functions

- **CFile** ()
Empty CFile (p. 220) ctor.
- **CFile** (const std::string &name, uint32_t mode=**modeRead**|**typeBinary**)
- bool **Close** ()
- bool **Delete** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Gets the las error message encountered.
- bool **Duplicate** (const std::string &newFileName)
- void **Flush** ()
- const std::string & **GetFileName** ()
- long **GetLength** ()
Returns the current length of the file.
- uint32_t **GetMode** ()
- long **GetPosition** ()
Returns the current position of the file pointer.
- bool **IsOpen** ()
- bool **Open** (const std::string &name, uint32_t mode=**modeRead**|**typeBinary**)
- bool **Open** ()
- int32_t **ReadLineData** (char *lineRead, uint32_t size)
- int32_t **ReadToBuffer** (char *destinationBuffer, uint32_t numBytesToRead=CFile::m_maxBufferToRead)
- bool **Rename** (const std::string &newName)
- bool **SeekToBegin** ()
- bool **SeekToEnd** ()
- bool **SetBufferingMode** (bool mode=true)
- bool **SetPosition** (long positionOffset)
- bool **Write** (const int character)
- bool **Write** (const std::string &str)
- bool **Write** (const char *str)
- bool **WriteChar** (const int character)
- uint32_t **WriteFromBuffer** (const char *sourceBuffer, uint32_t sourceBufferLength)
- bool **WriteString** (const char *str)
- virtual ~**CFile** ()
Destructor.

Static Public Member Functions

- static bool **Delete** (const std::string &filename)
- static bool **Rename** (const std::string &oldName, const std::string &newName)

Protected Attributes

- char **m_lastError** [BRATHL_MAX_ERRMSG_LEN+1]
last error message

8.57.1 Detailed Description

File management class.

This class provides unbuffered, binary and ascii disk input/output services.

While managing the file, if an error occurred, a CFileException is raised.

Version

1.0

8.57.2 Constructor & Destructor Documentation

8.57.2.1 `brathl::CFile::CFile (const std::string & name, uint32_t mode = modeRead|typeBinary)`

Creates new **CFile** (p. 220) object and opens the file. If an error occurred, a `CFileException` is raised.

Parameters

<i>name</i>	[in] : full name of the file;
<i>mode</i>	[in] : access mode - default value : <code>modeRead typeBinary</code> (see openFlags (p. 66));

References `Open()`.

8.57.3 Member Function Documentation

8.57.3.1 `bool brathl::CFile::Close ()`

Closes file object. **IsOpen()** (p. 223) and **Open()** (p. 223) are the only functions available just after this operation.

Returns

true on success, otherwise false

References `IsOpen()`.

Referenced by `Delete()`, `brathl::CFileParams::Load()`, `Open()`, and `Rename()`.

8.57.3.2 `bool brathl::CFile::Delete ()`

Closes file object and deletes (removes) the file. **IsOpen()** (p. 223) and **Open()** (p. 223) are the only functions available just after this operation.

Returns

true on success, otherwise false

References `Close()`, `IsOpen()`, and `Open()`.

8.57.3.3 `bool brathl::CFile::Delete (const std::string & filename) [static]`

Deletes (removes) a file.

Parameters

<i>filename</i>	[in] : file to delete/remove IsOpen() (p. 223) and Open() (p. 223) are the only functions available just after this operation.
-----------------	--

Returns

true on success, otherwise false

8.57.3.4 `void brathl::CFile::Dump (std::ostream & fOut = std::cerr) [virtual]`

Gets the las error message encountered.

Dump fonction

Reimplemented in **brathl::CFileParams** (p. 227).

Referenced by `brathl::CFileParams::Dump()`, `Open()`, `ReadToBuffer()`, `WriteChar()`, `WriteFromBuffer()`, and `WriteString()`.

8.57.3.5 `bool brathl::CFile::Duplicate (const std::string & newFileName)`

Creates a copy of current file with the `newFileName`. If file with specified filename exists, it's contents are erased.

Parameters

<i>newFileName</i>	[in] : copy to file name
--------------------	--------------------------

Returns

true on success, otherwise false

References GetLength(), GetPosition(), IsOpen(), modeWrite, ReadToBuffer(), SeekToBegin(), SetPosition(), and WriteFromBuffer().

8.57.3.6 const std::string & brathl::CFile::GetFileName ()

Gets the name of the file

8.57.3.7 uint32_t brathl::CFile::GetMode ()

Gets the name of the file

8.57.3.8 bool brathl::CFile::IsOpen ()

Tests if file is opened or not

Returns

true if opened, false otherwise

Referenced by Close(), Delete(), Duplicate(), GetLength(), GetPosition(), brathl::CFileParams::Load(), ReadToBuffer(), Rename(), SeekToBegin(), SeekToEnd(), SetBufferingMode(), SetPosition(), WriteChar(), WriteFromBuffer(), and WriteString().

8.57.3.9 bool brathl::CFile::Open (const std::string & name, uint32_t mode = modeRead|typeBinary)

Opens a file. If file object is open, it is closed. If an error occurred, a CFileException is raised.

Parameters

<i>name</i>	[in] : full name of the file;
<i>mode</i>	[in] : access mode - default value : modeRead typeBinary (see openFlags (p. 66));

Returns

true on success, otherwise false

References Open().

8.57.3.10 bool brathl::CFile::Open ()

Opens the current file object. If an error occurred, a CFileException is raised.

Returns

true on success, otherwise false

References Close(), Dump(), brathl::CTools::Format(), modeAppend, modeRead, modeReadAppend, modeReadWrite, modeRWCreate, modeWrite, SeekToBegin(), SeekToEnd(), SetBufferingMode(), and typeText.

Referenced by CFile(), Delete(), brathl::CFileParams::Load(), Open(), and Rename().

8.57.3.11 int32_t brathl::CFile::ReadLineData (char * lineRead, uint32_t size)

Same as #ReadLine, but reads only line of data and skip comments and places contents into buffer pointed by lineRead. Comments start with character '#' anywhere in the line. Empty line or space line are also skipped If an error occurred, a CFileException is raised.

Parameters

<i>lineRead</i>	[out] : line data read
<i>size</i>	[in] : max number of bytes of the line

Returns

the number of bytes in the lineRead parameter. -1 if end of file reached

References bratl::CTools::Trim().

Referenced by bratl::CFileParams::Load().

8.57.3.12 `int32_t bratl::CFile::ReadToBuffer (char * destinationBuffer, uint32_t numBytesToRead = CFile::m_maxBufferToRead)`

Function reads 'NumBytesToRead' bytes from the current file position and places file contents into buffer pointed by destinationBuffer. If an error occurred, a CFileException is raised.

Parameters

<i>destinationBuffer</i>	[out] : destination buffer
<i>numBytesToRead</i>	[in] : number of bytes to reads

Returns

the number of bytes actually reads, zero if end of file reached

References Dump(), bratl::CTools::Format(), GetLength(), GetPosition(), and IsOpen().

Referenced by Duplicate().

8.57.3.13 `bool bratl::CFile::Rename (const std::string & newName)`

Renames file object. If file with specified filename exists, its contents are erased. The current file is closed, renamed and opened as new name.

Parameters

<i>newName</i>	[in] : new file name
----------------	----------------------

Returns

true on success, otherwise false

References Close(), IsOpen(), and Open().

8.57.3.14 `bool bratl::CFile::Rename (const std::string & oldName, const std::string & newName) [static]`

Renames a file. If file with specified filename exists, its contents are erased.

Parameters

<i>oldName</i>	[in] : file to rename
<i>newName</i>	[in] : new file name

Returns

true on success, otherwise false

8.57.3.15 `bool bratl::CFile::SeekToBegin ()`

Function moves moves file pointer to the beginning of file.

Returns

true on success, otherwise false

References `IsOpen()`.

Referenced by `Duplicate()`, and `Open()`.

8.57.3.16 `bool bratl::CFile::SeekToEnd ()`

Function moves moves file pointer to the end of file.

Returns

true on success, otherwise false

References `IsOpen()`.

Referenced by `Open()`.

8.57.3.17 `bool bratl::CFile::SetBufferingMode (bool mode = true)`

Change buffering mode. Function must be used before any read/write operation occurs!

Parameters

<i>mode</i>	[in] : true if buffered I/O (default), false if unbuffered I/O
-------------	--

Returns

true on success, otherwise false

References `IsOpen()`.

Referenced by `Open()`.

8.57.3.18 `bool bratl::CFile::SetPosition (long positionOffset)`

Function moves file pointer by `PositionOffset` bytes relative to current position.

Parameters

<i>positionOffset</i>	[in] : offset to move
-----------------------	-----------------------

Returns

true on success, otherwise false

References `IsOpen()`.

Referenced by `Duplicate()`.

8.57.3.19 `bool bratl::CFile::WriteChar (const int character)`

Writes a single character to a file If an error occurred, a `CFileException` is raised.

Parameters

<i>character</i>	[in] : character to write
------------------	---------------------------

Returns

true on success, otherwise false

References Dump(), bratl::CTools::Format(), IsOpen(), and modeRead.

8.57.3.20 `uint32_t bratl::CFile::WriteFromBuffer (const char * sourceBuffer, uint32_t sourceBufferLength)`

Writes data from memory to a file If an error occurred, a CFileException is raised.

Parameters

<i>sourceBuffer</i>	[in] : data to write
<i>sourceBufferLength</i>	[in] : data length to write

Returns

the number of bytes actually written.

References Dump(), bratl::CTools::Format(), IsOpen(), and modeRead.

Referenced by Duplicate().

8.57.3.21 `bool bratl::CFile::WriteString (const char * str)`

Writes a std::string to a file If an error occurred, a CFileException is raised.

Parameters

<i>str</i>	[in] : std::string to write
------------	-----------------------------

Returns

true on success, otherwise false

References Dump(), bratl::CTools::Format(), IsOpen(), and modeRead.

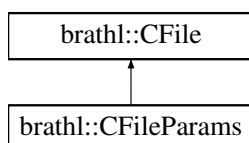
The documentation for this class was generated from the following files:

- File.h
- File.cpp

8.58 bratl::CFileParams Class Reference

```
#include <FileParams.h>
```

Inheritance diagram for bratl::CFileParams:



Public Member Functions

- **CFileParams** ()
Empty CFileParams (p. 227) ctor.
- **CFileParams** (const std::string &name, uint32_t mode=**modeRead**|**typeBinary**)
- unsigned **CheckCount** (const std::string &Key, int32_t ValidMin=1, int32_t ValidMax=1)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- void **GetFieldNames** (const std::string &key, CStringArray &fieldNames)
- **CStringMap** * **GetFieldSpecificUnits** ()
- void **GetFileList** (const std::string &key, CStringArray &fileNames)
- std::string **GetFirstFile** (const std::string &key)
- bool **IsLoaded** ()
- void **Load** ()
- void **LoadAliases** ()
- void **LoadFieldSpecificUnits** ()
- void **SetVerboseLevel** ()
- void **SubstituteAliases** (const **CStringMap** &aliases)
- virtual ~**CFileParams** ()
Destructor.
- void **Load** (const std::string &name, uint32_t mode=**modeRead**|**typeBinary**)

Public Attributes

- **CMapParameter** m_mapParam

Additional Inherited Members

8.58.1 Detailed Description

Parameters file management class.

This class provides ascii parameters file services

It makes it possible to acquire the whole of information which they contain

Parameters are described as 'keyword'='value'

keyword is character strings identifying a type of data. value is character strings associated with the key.

keyword and value have to be on the same line;

It don't make distinction between upper-case and lower-case letters.

While managing the file, if an error occurred, a CFileException is raised. While managing parameter (keyword, value), if an error occurred, a CParameterException is raised.

Version

1.0

8.58.2 Constructor & Destructor Documentation

8.58.2.1 brathl::CFileParams::CFileParams (const std::string & name, uint32_t mode = modeRead|typeBinary)

Creates new **CFileParams** (p.227) object and opens the parameters file. On error, a CFileException or C↔ParameterException exception is raised.

Parameters

<i>name</i>	[in] : full name of the file;
<i>mode</i>	[in] : access mode - default value : <code>modeRead typeBinary</code> (see openFlags (p. 66));

References `Load()`.

8.58.3 Member Function Documentation

8.58.3.1 `unsigned brathl::CFileParams::CheckCount (const std::string & Key, int32_t ValidMin = 1, int32_t ValidMax = 1)`

Throw an exception if the number of values is not valid.

Parameters

<i>ValidMin</i>	[in] : Minimal number of values
<i>ValidMax</i>	[in] : Maximal number of values. If ≤ 0 , it is considered as infinite. If $< ValidMin$, it is considered as equal to <code>ValidMin</code> .

Returns

actual number of occurrences of the parameter

References `brathl::CParameter::Count()`, `brathl::CMapParameter::Exists()`, `brathl::CTools::Format()`, and `m_mapParam`.

Referenced by `SetVerboseLevel()`.

8.58.3.2 `void brathl::CFileParams::Load ()`

Reads file parameters and load parameters On error, a `CFileException` or `CParameterException` exception is raised.

References `brathl::CFile::Close()`, `brathl::CFile::GetLength()`, `brathl::CFile::IsOpen()`, `m_mapParam`, `brathl::CFile::Open()`, `brathl::CFile::ReadLineData()`, and `brathl::CMapParameter::RemoveAll()`.

Referenced by `CFileParams()`, and `Load()`.

8.58.3.3 `void brathl::CFileParams::Load (const std::string & name, uint32_t mode = modeRead|typeBinary)`

Reads file parameters and load parameters On error, a `CFileException` or `CParameterException` exception is raised.

Parameters

<i>name</i>	[in] : full name of the file;
<i>mode</i>	[in] : access mode - default value : <code>modeRead typeBinary</code> (see openFlags (p. 66));

References `Load()`, and `brathl::CFile::Open()`.

8.58.3.4 `void brathl::CFileParams::SetVerboseLevel ()`

Set the verbosity level from the standard keyword `VERBOSE`

References `CheckCount()`, and `m_mapParam`.

8.58.4 Member Data Documentation

8.58.4.1 `CMapParameter brathl::CFileParams::m_mapParam`

A map containing all the parameters

Referenced by `CheckCount()`, `Dump()`, `Load()`, and `SetVerboseLevel()`.

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

- FileParams.h
- FileParams.cpp

8.59 brathl::CProduct::CInfo Class Reference

```
#include <Product.h>
```

Inherits brathl::CBratObject.

Public Attributes

- std::string **m_fieldName**
- int32_t **m_index**
- int32_t **m_isUnion**
- coda_Type * **m_type**
- coda_type_class **m_type_class**

8.59.1 Detailed Description

A class to traverse Brat files

Version

1.0

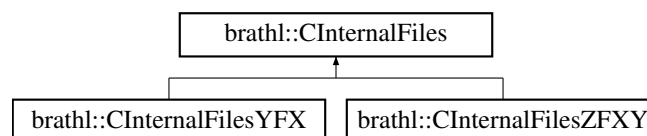
The documentation for this class was generated from the following files:

- Product.h
- Product.cpp

8.60 brathl::CInternalFiles Class Reference

```
#include <InternalFiles.h>
```

Inheritance diagram for brathl::CInternalFiles:



Public Member Functions

- CNetCDFDimension * **AddNetCDFDim** (CNetCDFDimension &dim, bool forceReplace=false)
- CNetCDFVarDef * **AddNetCDFVarDef** (CNetCDFVarDef &var, bool forceReplace=false)
- **CInternalFiles** (std::string Name="", brathl_FileMode Mode=ReadOnly)
- virtual void **Close** ()
- int **GetAttribute** (const std::string &varName, const std::string &attName, double &attValue, bool mustExist=true, double defaultValue=CTools::m_defaultValueDOUBLE)
- int **GetAttribute** (const std::string &varName, const std::string &attName, std::string &attValue, bool mustExist=true, std::string defaultValue="")
- virtual void **GetAxisVars** (std::vector< std::string > &VarNames)
- std::string **GetComment** (const std::string &varName)

- virtual bool **GetCommonVarDims** (const std::string &varName1, const std::string &varName2, CStringArray &intersect)
- virtual bool **GetComplementVarDims** (const std::string &varName1, const std::string &varName2, CStringArray &complement)
- virtual bool **GetComplementVars** (const CStringArray &varNames, CStringArray &complement, bool excludeDim=true)
- virtual void **GetDataVars** (std::vector< std::string > &VarNames)
- int **GetDimId** (const std::string &name)
- CNetCDFFiles * **GetFile** ()
- uint32_t **GetMaxFieldNumberOfDims** (const CStringArray *fieldNames=NULL)
- virtual std::string **GetName** () const
- CNetCDFDimension * **GetNetCDFDim** (const std::string &name)
- CObMap * **GetNetCDFDims** ()
- void **GetNetCDFDims** (const std::string &varName, CObArray *dims)
- CNetCDFVarDef * **GetNetCDFVarDef** (const std::string &name)
- CObMap * **GetNetCDFVarDefs** ()
- virtual std::string **GetTitle** (const std::string &Name)
- virtual std::string **GetType** ()
- virtual CUnit **GetUnit** (const std::string &Name)
- int32_t **GetVarDimIndex** (const std::string &varName, const std::string &dimName)
- virtual void **GetVarDims** (const std::string &Name, ExpressionValueDimensions &Dimensions)
- virtual void **GetVarDims** (const std::string &Name, std::vector< std::string > &Dimensions)
- virtual void **GetVariables** (std::vector< std::string > &VarNames)
- virtual NetCDFVarKind **GetVarKind** (const std::string &Name)
- virtual bool **HasVar** (NetCDFVarKind VarKind)
- bool **IsAxisVar** (const std::string &Name)
- virtual bool **IsGeographic** ()
- virtual bool **IsOpened** ()
- virtual void **Open** (brathl_FileMode mode)
- virtual void **Open** ()
- virtual void **ReadVar** (const std::string &Name, CExpressionValue &Value, const std::string &WantedUnit)
- void **ReplaceNetCDFDim** (CNetCDFDimension &dim)
- virtual void **SetMode** (brathl_FileMode mode)
- virtual void **SetName** (const std::string &name)
- virtual bool **VarExists** (const std::string &Name)
- virtual void **WriteData** (CNetCDFVarDef *varDef, CExpressionValue *data)
- virtual void **WriteData** (CNetCDFVarDef *varDef, CMatrix *matrix)
- virtual void **WriteDimensions** ()
- virtual void **WriteFileTitle** (const std::string &Title)
- virtual void **WriteVar** (const std::string &Name, const CExpressionValue &Value)
- virtual void **WriteVariables** ()

Static Public Member Functions

- static CInternalFiles * **Create** (const std::string &fileName, bool open=true, bool withExcept=true)
- static bool **IsVarNameValid** (const std::string &Name)
- static bool **IsYFXFile** (const std::string &fileName, CInternalFiles **pf=NULL)
- static bool **IsYFXFile** (CInternalFiles *f, CStringArray *fieldNamesIn=NULL)
- static bool **IsZFlatLonFile** (const std::string &fileName, CInternalFiles **pf=NULL)
- static bool **IsZFlatLonFile** (CInternalFiles *f)
- static bool **IsZFXFile** (const std::string &fileName, CStringArray *fieldNames=NULL, CInternalFiles **pf=NULL)
- static bool **IsZFXFile** (CInternalFiles *f, CStringArray *fieldNames=NULL)
- static std::string **TypeOf** ()

Protected Member Functions

- void **SetFixedGlobalAttributes** (void)

Protected Attributes

- CNetCDFFiles **m_file**

8.60.1 Detailed Description

Internal files access.

Version

1.0

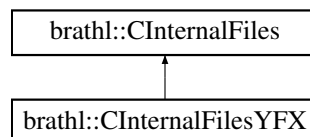
The documentation for this class was generated from the following files:

- InternalFiles.h
- InternalFiles.cpp

8.61 bratl::CInternalFilesYFX Class Reference

```
#include <InternalFilesYFX.h>
```

Inheritance diagram for bratl::CInternalFilesYFX:



Public Member Functions

- **CInternalFilesYFX** (std::string Name="", **bratl_FileMode** Mode=ReadOnly)
- virtual void **CreateData** (const std::string &Name, const std::string &Units, const std::string &LongName, const std::string &Comment="", double ValidMin=**CTools::m_defaultValueDOUBLE**, double ValidMax=**CTools::m_defaultValueDOUBLE**, nc_type Type=NC_DOUBLE)
- virtual void **CreateDim** (NetCDFVarKind Kind, const std::string &XName, const **CExpressionValue** &Values, const std::string &Units, const std::string &LongName, const std::string &Comment="", double ValidMin=**CTools::m_defaultValueDOUBLE**, double ValidMax=**CTools::m_defaultValueDOUBLE**)
- virtual std::string **GetType** ()

Static Public Member Functions

- static std::string **TypeOf** ()

Additional Inherited Members

8.61.1 Detailed Description

Internal files access for internal files used to store Y=F(X) kind of data.

Version

1.0

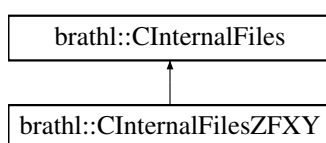
The documentation for this class was generated from the following files:

- InternalFilesYFX.h
- InternalFilesYFX.cpp

8.62 bratl::CInternalFilesZFX Y Class Reference

```
#include <InternalFilesZFX Y.h>
```

Inheritance diagram for bratl::CInternalFilesZFX Y:



Public Member Functions

- **CInternalFilesZFX Y** (std::string Name="", **bratl_FileMode** Mode=ReadOnly)
- virtual void **CreateData** (const std::string &Name, const std::string &Units, const std::string &LongName, const std::string &Dim1Name, const std::string &Dim2Name, const std::string &Comment="", double ValidMin=**CTools::m_defaultValueDOUBLE**, double ValidMax=**CTools::m_defaultValueDOUBLE**, nc_type Type=NC_DOUBLE)
- virtual void **CreateDim** (NetCDFVarKind Kind, const std::string &XName, const **CExpressionValue** &Values, const std::string &Units, const std::string &LongName, const std::string &Comment="", double ValidMin=**CTools::m_defaultValueDOUBLE**, double ValidMax=**CTools::m_defaultValueDOUBLE**)
- virtual std::string **GetType** ()
- virtual bool **IsGeographic** ()

Static Public Member Functions

- static std::string **TypeOf** ()

Additional Inherited Members

8.62.1 Detailed Description

Internal files access for internal files used to store $Y=F(X)$ kind of data.

Version

1.0

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

- InternalFilesZFX Y.h
- InternalFilesZFX Y.cpp

8.63 brathl::CIntList Class Reference

```
#include <List.h>
```

Inherits intlist.

Public Member Functions

- **CIntList** ()
Empty CIntList (p. 233) ctor.
- **CIntList** (const **CIntList** &list)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual void **Insert** (const **CIntList** &list, bool bEnd=true)
- virtual void **Insert** (const int value, bool bEnd=true)
- const **CIntList** & **operator=** (const **CIntList** &lst)
- virtual void **RemoveAll** ()
- virtual ~**CIntList** ()
Destructor.

8.63.1 Detailed Description

A std::list of strings management class.

Version

1.0

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

- List.h
- List.cpp

8.64 brathl::CIntMap Class Reference

```
#include <List.h>
```

Inherits mapint.

Public Member Functions

- **CIntMap** ()
CIntMap (p. 234) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CIntMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual int32_t **Exists** (const std::string &key) const
- virtual int32_t **Insert** (const std::string &key, int32_t value, bool withExcept=true)
- virtual void **Insert** (const **CIntMap** &m, bool bRemoveAll=true, bool withExcept=true)
- virtual void **Insert** (const CStringArray &keys, const CIntArray &values, bool bRemoveAll=true, bool withExcept=true)
- virtual int32_t **operator[]** (const std::string &key)
- virtual void **RemoveAll** ()
- virtual ~**CIntMap** ()
CIntMap (p. 234) dtor.

8.64.1 Detailed Description

a set of integer value management classes.

Version

1.0

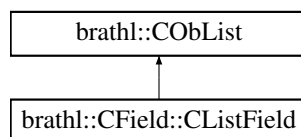
The documentation for this class was generated from the following files:

- List.h
- List.cpp

8.65 brathl::CField::CListField Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CField::CListField:



Public Member Functions

- **CField * Back** (bool withExcept=true)
- **CListField** (bool bDelete)
- **CField * Front** (bool withExcept=true)
- virtual void **InsertField** (**CField** *field, bool hasDataset=true, bool bEnd=true)
- void **RemoveAll** ()

Public Attributes

- CUIntArray **m_fieldSetDim**
- int32_t **m_nbFieldSetDims**

Additional Inherited Members

8.65.1 Detailed Description

A list of **CField** (p. 201) object management class

Version

1.0

8.65.2 Member Function Documentation

8.65.2.1 void brathl::CField::CListField::RemoveAll () [virtual]

Remove all elements and clear the std::list

Reimplemented from **brathl::CObList** (p. 43).

References brathl::CObList::RemoveAll().

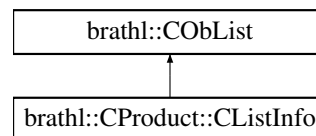
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.66 brathl::CProduct::CListInfo Class Reference

```
#include <Product.h>
```

Inheritance diagram for brathl::CProduct::CListInfo:



Public Member Functions

- **CInfo * AddNew** ()
- **CInfo * Back** (bool withExcept=true)
- **CInfo * Front** (bool withExcept=true)
- **CInfo * PrevBack** (bool withExcept=true)

Additional Inherited Members

8.66.1 Detailed Description

A list of **CInfo** (p. 229) object management class

Version

1.0

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

- Product.h
- Product.cpp

8.67 brathl::CMapParameter Class Reference

```
#include <MapParameter.h>
```

Inherits map_parameter.

Public Member Functions

- **CMapParameter** ()
CMapParameter (p. 236) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- bool **Erase** (CMapParameter::iterator iteratorParameter)

- bool **Erase** (const std::string &key)
- **CParameter** * **Exists** (const std::string &key)
- **CParameter** * **Insert** (const std::string &key, const std::string &value)
- **CParameter** * **operator[]** (const std::string key)
- void **RemoveAll** ()
- virtual ~**CMapParameter** ()

CMapParameter (p. 236) *dtor.*

8.67.1 Detailed Description

Parameter management class.

This class provides a std::map of **CParameter** (p. 242) objects

Version

1.0

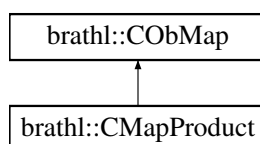
The documentation for this class was generated from the following files:

- **MapParameter.h**
- **MapParameter.cpp**

8.68 bratl::CMapProduct Class Reference

```
#include <Product.h>
```

Inheritance diagram for bratl::CMapProduct:



Public Member Functions

- void **AddCriteriaToProducts** ()
- **CMapProduct** ()
CIntMap (p. 234) *ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr)
- void **GetProductKeysWithCriteria** (CStringArray &keys)
- void **RemoveCriteriaFromProducts** ()
- virtual ~**CMapProduct** ()
CIntMap (p. 234) *dtor.*

Static Public Member Functions

- static **CMapProduct** & **GetInstance** ()

Protected Member Functions

- void **Init** ()

Additional Inherited Members

8.68.1 Detailed Description

Mapping products management class.

Version

1.0

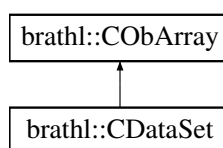
The documentation for this class was generated from the following files:

- Product.h
- Product.cpp

8.69 brathl::CObArray Class Reference

```
#include <List.h>
```

Inheritance diagram for brathl::CObArray:



Public Member Functions

- **CObArray** (bool bDelete=true)
Empty CObArray (p. 237) ctor.
- **CObArray** (const **CObArray** &vect)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- bool **Erase** (CBratObject *ob)
- virtual bool **Erase** (CObArray::iterator it)
- virtual bool **Erase** (int32_t index)
- bool **GetDelete** ()
- virtual void **Insert** (const **CObArray** &vect)
- virtual void **Insert** (CBratObject *ob)
- virtual CObArray::iterator **InsertAt** (CObArray::iterator where, CBratObject *ob)
- virtual const **CObArray** & **operator=** (const **CObArray** &lst)
- virtual bool **PopBack** ()
- virtual void **RemoveAll** ()
- virtual CObArray::iterator **ReplaceAt** (CObArray::iterator where, CBratObject *ob)
- void **SetDelete** (bool value)
- virtual ~**CObArray** ()
Destructor.

Protected Attributes

- bool **m_bDelete**

8.69.1 Detailed Description

An array (std::vector) of CBratObject management class.

Version

1.0

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

- List.h
- List.cpp

8.70 bratl::CObDoubleMap Class Reference

```
#include <List.h>
```

Inherits mapdoubleobject.

Public Member Functions

- **CObDoubleMap** (bool bDelete=true)
CObMap (p. 241) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CObDoubleMap::iterator it)
- virtual bool **Erase** (double key)
- virtual CBratObject * **Exists** (double key) const
- bool **GetDelete** ()
- virtual void **GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- virtual CBratObject * **Insert** (double key, CBratObject *ob, bool withExcept=true)
- virtual void **Insert** (const **CObDoubleMap** &obMap, bool withExcept=true)
- virtual const **CObDoubleMap** & **operator=** (const **CObDoubleMap** &obMap)
- virtual CBratObject * **operator[]** (double key)
- virtual void **RemoveAll** ()
- bool **RenameKey** (double oldKey, double newKey)
- void **SetDelete** (bool value)
- virtual ~**CObDoubleMap** ()
CObMap (p. 241) dtor.

Protected Attributes

- bool **m_bDelete**

8.70.1 Detailed Description

a set of object management classes.

Version

1.0

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

- List.h
- List.cpp

8.71 brathl::COblntMap Class Reference

```
#include <List.h>
```

Inherits mapintobject.

Public Member Functions

- **COblntMap** (bool bDelete=true)
COBMap (p. 241) *ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (COblntMap::iterator it)
- virtual bool **Erase** (int32_t key)
- virtual CBratObject * **Exists** (int32_t key) const
- bool **GetDelete** ()
- virtual void **GetKeys** (CIntArray &keys, bool bRemoveAll=true)
- virtual CBratObject * **Insert** (int32_t key, CBratObject *ob, bool withExcept=true)
- virtual void **Insert** (const **COblntMap** &obMap, bool withExcept=true)
- virtual const **COblntMap** & **operator=** (const **COblntMap** &obMap)
- virtual CBratObject * **operator[]** (int32_t key)
- virtual void **RemoveAll** ()
- bool **RenameKey** (int32_t oldKey, int32_t newKey)
- void **SetDelete** (bool value)
- virtual ~**COblntMap** ()
COBMap (p. 241) *dtor.*

Protected Attributes

- bool **m_bDelete**

8.71.1 Detailed Description

a set of object management classes.

Version

1.0

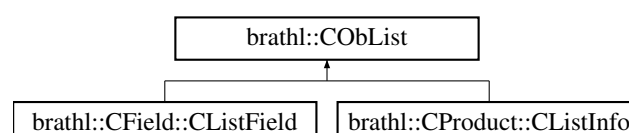
The documentation for this class was generated from the following files:

- List.h
- List.cpp

8.72 brathl::CObList Class Reference

```
#include <List.h>
```

Inheritance diagram for brathl::CObList:



Public Member Functions

- **CObList** (bool bDelete=true)
*Empty **CObList** (p. 240) ctor.*
- **CObList** (const **CObList** &lst)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- bool **Erase** (CBratObject *ob)
- virtual bool **Erase** (CObList::iterator it)
- bool **GetDelete** ()
- virtual void **Insert** (const **CObList** &list, bool bEnd=true)
- virtual void **Insert** (CBratObject *ob, bool bEnd=true)
- virtual const **CObList** & **operator=** (const **CObList** &lst)
- virtual bool **PopBack** ()
- virtual void **RemoveAll** ()
- void **SetDelete** (bool value)
- virtual ~**CObList** ()
Destructor.

Protected Attributes

- bool **m_bDelete**

8.72.1 Detailed Description

A std::list of CBratObject management class.

Version

1.0

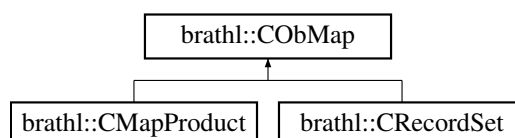
The documentation for this class was generated from the following files:

- List.h
- List.cpp

8.73 bratl::CObMap Class Reference

```
#include <List.h>
```

Inheritance diagram for bratl::CObMap:



Public Member Functions

- **CObMap** (bool bDelete=true)
CObMap (p. 241) *ctor.*
- **CObMap** (const **CObMap** &obMap)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CObMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual CBratObject * **Exists** (const std::string &key) const
- bool **GetDelete** ()
- virtual void **GetKeys** (CStringArray &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual void **GetKeys** (CStringList &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual CBratObject * **Insert** (const std::string &key, CBratObject *ob, bool withExcept=true)
- virtual void **Insert** (const **CObMap** &obMap, bool withExcept=true)
- virtual const **CObMap** & **operator=** (const **CObMap** &obMap)
- virtual CBratObject * **operator[]** (const std::string &key)
- virtual void **RemoveAll** ()
- bool **RenameKey** (const std::string &oldKey, const std::string &newKey)
- void **SetDelete** (bool value)
- virtual void **ToArray** (CObArray &obArray)
- virtual ~**CObMap** ()
CObMap (p. 241) *dtor.*

Protected Attributes

- bool **m_bDelete**

8.73.1 Detailed Description

a set of object management classes.

Version

1.0

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

- List.h
- List.cpp

8.74 brathl::CObStack Class Reference

```
#include <List.h>
```

Inherits obstack.

Public Member Functions

- **CObStack** (bool bDelete=true)
Empty CObArray (p. 237) *ctor.*
- bool **GetDelete** ()
- virtual void **Pop** ()
- virtual void **Push** (CBratObject *ob)

- virtual void **RemoveAll** ()
- void **SetDelete** (bool value)
- virtual CBratObject * **Top** ()
- virtual ~**CObStack** ()

Destructor.

Protected Attributes

- bool **m_bDelete**

Dump function.

8.74.1 Detailed Description

An std::stack of CBratObject management class.

Version

1.0

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

- List.h
- List.cpp

8.75 brathl::CParameter Class Reference

```
#include <Parameter.h>
```

Public Member Functions

- size_t **Count** ()
- **CParameter** ()
Empty CParameter (p. 242) ctor.
- **CParameter** (const char *keyword)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- void **GetValue** (char *value, size_t bufferSize, int32_t pos=0, const char *DefValue="")
- bool **RemoveAllValue** ()
- bool **RemoveValue** (uint32_t i)
- void **SetAliases** (const CStringMap &aliases)
- virtual ~**CParameter** ()
Destructor.
- **CParameter** (const char *keyword, const char *value)
- **CParameter** (const std::string &keyword, const std::string &value)
- void **AddValue** (const char *value)
- void **AddValue** (const std::string &value)
- void **GetValue** (int32_t &value, int32_t pos=0, int32_t DefValue=CTools::m_defaultValueINT32)
- void **GetValue** (uint32_t &value, int32_t pos=0, uint32_t DefValue=CTools::m_defaultValueUINT32)
- void **GetValue** (double &value, int32_t pos=0, double DefValue=CTools::m_defaultValueDOUBLE)

- void **GetValue** (bool &value, int32_t pos=0, bool DefValue=false)
- void **GetValue** (CDate &value, int32_t pos=0)
- void **GetValue** (CDate &value, CUnit &unit, int32_t pos=0)
- void **GetValue** (CDate &value, const std::string &strUnit, int32_t pos=0)
- void **GetValue** (CDate &value, CUnit *unit, int32_t pos=0)
- void **GetValue** (std::string &value, int32_t pos=0, const std::string &DefValue="")
- void **GetValue** (CExpression &value, int32_t pos=0)
- void **GetValue** (CUnit &value, int32_t pos=0, const std::string &DefValue="count")
- void **GetValue** (uint32_t &value, std::string &ValueName, const KWValueListEntry *KeywordList, int32_t pos=0, uint32_t DefValue=CTools::m_defaultValueUINT32)
- void **GetValue** (bitSet32 &value, const KWValueListEntry *KeywordList, int32_t pos=0, const bitSet32 &DefValue=0)
- void **GetValue** (uint32_t &value, std::string &ValueName, CUIntMap &KeywordList, int32_t pos, uint32_t DefValue)
- void **GetAllValues** (CExpression &value, const std::string &Combine="&&")
- void **GetAllValues** (CStringList &listValues)
- void **GetAllValues** (CStringArray &listValues)

8.75.1 Detailed Description

Parameter management class.

One parameter can have 1 to n value.

This class stands for parameters

Version

1.0

8.75.2 Constructor & Destructor Documentation

8.75.2.1 brathl::CParameter::CParameter (const char * *keyword*)

Creates a new **CParameter** (p. 242) object.

Parameters

<i>keyword</i>	[in] : parameter name
----------------	-----------------------

8.75.2.2 brathl::CParameter::CParameter (const char * *keyword*, const char * *value*)

Creates a new **CParameter** (p. 242) object.

Parameters

<i>keyword</i>	[in] : parameter name
<i>value</i>	[in] : parameter value

References AddValue().

8.75.3 Member Function Documentation

8.75.3.1 void brathl::CParameter::AddValue (const char * *value*)

Adds a value to the **CParameter** (p. 242) object.

Parameters

<i>value</i>	[in] : parameter value
--------------	------------------------

References `brathl::CTools::ExpandShellVar()`.

Referenced by `CParameter()`, and `brathl::CMapParameter::Insert()`.

8.75.3.2 `size_t brathl::CParameter::Count ()`

Returns

the number of values.

Referenced by `brathl::CFileParams::CheckCount()`.

8.75.3.3 `void brathl::CParameter::GetValue (int32_t & value, int32_t pos = 0, int32_t DefValue = CTools::m_defaultValueINT32)`

gets a **CParameter** (p. 242) object value at a given position If the list of values is empty or index `pos` is out of range a `CParameterException` is raised.

Parameters

<i>value</i>	[out] : parameter value
<i>pos</i>	[in] : position of the parameter 0.. n (default is 0, first value)

References `brathl::CTools::Format()`, and `brathl::CTools::StrCaseCmp()`.

8.75.3.4 `void brathl::CParameter::GetValue (char * value, size_t bufferSize, int32_t pos = 0, const char * DefValue = " ")`

gets a **CParameter** (p. 242) object value at a given position If the list of values is empty or index `pos` is out of range a `CParameterException` is raised. WARNING : if size of `std::string` value is smaller than the size of the parameter value, data will be truncated

Parameters

<i>value</i>	[out] : parameter value
<i>bufferSize</i>	[in] : size of value
<i>pos</i>	[in] : position of the parameter 0.. n (default is 0, first value)

Returns

false if one can't get the value, otherwise true

References `brathl::CTools::StrCaseCmp()`.

8.75.3.5 `bool brathl::CParameter::RemoveAllValue ()`

Removes all values.

Referenced by `~CParameter()`.

8.75.3.6 `bool brathl::CParameter::RemoveValue (uint32_t i)`

Removes a value at a given position. The first value is at the index 0.

Parameters

<i>i</i>	[in] : index value to remove
----------	------------------------------

8.75.3.7 `void brathl::CParameter::SetAliases (const CStringMap & aliases)`

Register the formulas aliases defined.

Parameters

<i>Aliases</i>	[in] : Names/values of aliases
----------------	--------------------------------

References brathl::CTools::ExpandVariables().

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

- Parameter.h
- Parameter.cpp

8.76 brathl::CProductAop Class Reference

```
#include <ProductAop.h>
```

Inherits brathl::CProduct.

Public Member Functions

- **CProductAop** ()
Empty CProductAop (p. 245) ctor.
- **CProductAop** (const std::string &fileName)
- **CProductAop** (const CStringList &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductAop** ()
Destructor.

Protected Member Functions

- virtual void **InitDateRef** ()

8.76.1 Detailed Description

Aop product management class.

Version

1.0

8.76.2 Constructor & Destructor Documentation

8.76.2.1 brathl::CProductAop::CProductAop (const std::string & fileName)

Creates new **CProductAop** (p. 245) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.76.2.2 brathl::CProductAop::CProductAop (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductAop** (p. 245) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

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

- ProductAop.h
- ProductAop.cpp

8.77 brathl::CProductCryosat Class Reference

```
#include <ProductCryosat.h>
```

Inherits brathl::CProduct.

Public Member Functions

- **CProductCryosat** ()
Empty CProductCryosat (p. 246) ctor.
- **CProductCryosat** (const std::string &fileName)
- **CProductCryosat** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductCryosat** ()
Destructor.

Protected Member Functions

- virtual bool **FindParentToRead** (**CField** *fromField, **CObList** *parentFieldList)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (**CField** *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()

8.77.1 Detailed Description

Cryosat product management class.

Version

1.0

8.77.2 Constructor & Destructor Documentation

8.77.2.1 brathl::CProductCryosat::CProductCryosat (const std::string & fileName)

Creates new **CProductCryosat** (p. 246) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.77.2.2 brathl::CProductCryosat::CProductCryosat (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductCryosat** (p. 246) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

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

- ProductCryosat.h
- ProductCryosat.cpp

8.78 brathl::CProductEnvisat Class Reference

```
#include <ProductEnvisat.h>
```

Inherits brathl::CProduct.

Public Member Functions

- **CProductEnvisat** ()
Empty CProductEnvisat (p. 247) ctor.
- **CProductEnvisat** (const std::string &fileName)
- **CProductEnvisat** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductEnvisat** ()
Destructor.

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (**CDataSet** *dataSet)
- void **ComputeHighResolutionFields** (**CDataSet** *dataSet, double deltaLat, double deltaLon)
- virtual bool **FindParentToRead** (**CField** *fromField, **CObList** *parentFieldList)
- virtual std::string **GetHighResolutionLatDiffFieldName** ()
- virtual std::string **GetHighResolutionLonDiffFieldName** ()
- virtual bool **HasHighResolutionFieldCalculation** ()
- bool **HasHighResolutionFieldCalculationValue** (**CDataSet** *dataset)
- bool **HasHighResolutionFieldCalculationValue** (**CDataSet** *dataset, **CFieldSetArrayDbf** *fieldSetArray↔
Dbf)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (**CField** *field)
- bool **IsParentHighResolutionField** (**CField** *field)
- virtual void **ProcessHighResolutionWithFieldCalculation** ()
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()
- virtual void **SetHighResolutionLatDiffFieldName** (const std::string &value)
- virtual void **SetHighResolutionLonDiffFieldName** (const std::string &value)

Protected Attributes

- **CStringArray** **m_arrayTimeStampFieldName**
- std::string **m_highResolutionLatDiffFieldName**
- std::string **m_highResolutionLonDiffFieldName**
- std::string **m_timeStampFieldName**

8.78.1 Detailed Description

Envisat product management class.

Version

1.0

8.78.2 Constructor & Destructor Documentation

8.78.2.1 `brathl::CProductEnvisat::CProductEnvisat (const std::string & fileName)`

Creates new **CProductEnvisat** (p. 247) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.78.2.2 `brathl::CProductEnvisat::CProductEnvisat (const CStringList & fileNameList, bool check_only_first_file)`

Creates new **CProductEnvisat** (p. 247) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.78.3 Member Function Documentation

8.78.3.1 `virtual std::string brathl::CProductEnvisat::GetHighResolutionLatDiffFieldName () [inline], [protected], [virtual]`

Get the "High resolution latitude differences" field name

8.78.3.2 `virtual std::string brathl::CProductEnvisat::GetHighResolutionLonDiffFieldName () [inline], [protected], [virtual]`

Get the "High resolution longitude differences" field name

8.78.3.3 `bool brathl::CProductEnvisat::IsHighResolutionField (CField * field) [protected], [virtual]`

Determines if a field object is a 'high resolution' array data For Envisat, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 20.
- the field name is different from the '18 Hz latitude differences from 1 Hz' field (1) and the '18 Hz longitude differences from 1 Hz' field (1)

(1) if this field are present in the record. Note that only off-line product (product type RA2_GDR_2P and RA2_MWS_2P have these fields
- the field name contains 'hz18' or '18hz'

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

References `brathl::CTools::StringToLower()`.

8.78.3.4 `virtual void bratl::CProductEnvisat::SetHighResolutionLatDiffFieldName (const std::string & value) [inline], [protected], [virtual]`

Set the "High resolution latitude differences" field name

8.78.3.5 `virtual void bratl::CProductEnvisat::SetHighResolutionLonDiffFieldName (const std::string & value) [inline], [protected], [virtual]`

Set the "High resolution longitude differences" field name

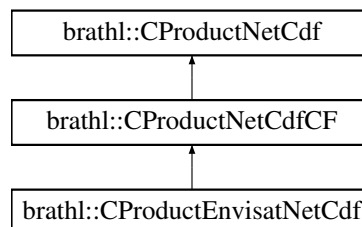
The documentation for this class was generated from the following files:

- ProductEnvisat.h
- ProductEnvisat.cpp

8.79 bratl::CProductEnvisatNetCdf Class Reference

```
#include <ProductEnvisatNetCdf.h>
```

Inheritance diagram for bratl::CProductEnvisatNetCdf:



Public Member Functions

- **CProductEnvisatNetCdf** ()
Empty CProductEnvisatNetCdf (p. 249) ctor.
- **CProductEnvisatNetCdf** (const std::string &path)
- **CProductEnvisatNetCdf** (const CStringList &paths, bool check_only_first_files)
- virtual void **InitDateRef** ()
- virtual ~**CProductEnvisatNetCdf** ()
Destructor.

Additional Inherited Members

8.79.1 Detailed Description

Reaper product management class.

Version

1.0

8.79.2 Constructor & Destructor Documentation

8.79.2.1 `bratl::CProductEnvisatNetCdf::CProductEnvisatNetCdf (const std::string & path) [inline]`

Creates new **CProductEnvisatNetCdf** (p. 249) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.79.2.2 **bratl::CProductEnvisatNetCdf::CProductEnvisatNetCdf** (**const CStringList &paths**, **bool check_only_first_files**)
[inline]

Creates new **CProductEnvisatNetCdf** (p. 249) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

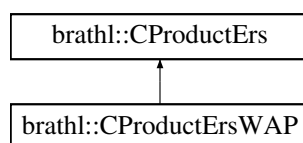
The documentation for this class was generated from the following file:

- ProductEnvisatNetCdf.h

8.80 **bratl::CProductErs** Class Reference

```
#include <ProductErs.h>
```

Inheritance diagram for **bratl::CProductErs**:



Public Member Functions

- **CProductErs** ()
Empty CProductErs (p. 250) ctor.
- **CProductErs** (const std::string &fileName)
- **CProductErs** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual **~CProductErs** ()
Destructor.

Static Public Attributes

- static const std::string **m_WAP** = "ALT.WAP"

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (**CDataSet** *dataSet, double deltaLat, double deltaLon)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (**CField** *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()

Protected Attributes

- std::string **m_timeStampMicrosecondFieldName**
- std::string **m_timeStampSecondFieldName**

8.80.1 Detailed Description

Ers product management class.

Version

1.0

8.80.2 Constructor & Destructor Documentation

8.80.2.1 brathl::CProductErs::CProductErs (const std::string & *fileName*)

Creates new **CProductErs** (p. 250) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.80.2.2 brathl::CProductErs::CProductErs (const CStringList & *fileNameList*, bool *check_only_first_file*)

Creates new **CProductErs** (p. 250) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.80.3 Member Function Documentation

8.80.3.1 bool brathl::CProductErs::IsHighResolutionField (CField * *field*) [protected], [virtual]

Determines if a field object is a 'high resolution' array data For Jason, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 10.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

Reimplemented in **brathl::CProductErsWAP** (p. 253).

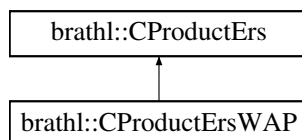
The documentation for this class was generated from the following files:

- ProductErs.h
- ProductErs.cpp

8.81 brathl::CProductErsWAP Class Reference

```
#include <ProductErsWAP.h>
```

Inheritance diagram for brathl::CProductErsWAP:



Public Member Functions

- **CProductErsWAP** ()
Empty CProductErsWAP (p. 252) ctor.
- **CProductErsWAP** (const std::string &fileName)
- **CProductErsWAP** (const CStringList &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductErsWAP** ()
Destructor.

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (CDataSet *dataSet)
- virtual bool **FindParentToRead** (CField *fromField, CObList *parentFieldList)
- virtual void **InitDateRef** ()
- virtual bool **IsDirectHighResolutionField** (CField *field)
- virtual bool **IsHighResolutionField** (CField *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()

Protected Attributes

- std::string **m_timeStampDayFieldName**
- std::string **m_timeStampMicrosecondFieldName**
- std::string **m_timeStampMillisecondFieldName**

Additional Inherited Members

8.81.1 Detailed Description

Ers product management class.

Version

1.0

8.81.2 Constructor & Destructor Documentation

8.81.2.1 brathl::CProductErsWAP::CProductErsWAP (const std::string & fileName)

Creates new **CProductErsWAP** (p. 252) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.81.2.2 brathl::CProductErsWAP::CProductErsWAP (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductErsWAP** (p. 252) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.81.3 Member Function Documentation

8.81.3.1 bool brathl::CProductErsWAP::IsHighResolutionField (CField * field) [protected], [virtual]

Determines if a field object is a 'high resolution' array data For Jason, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 10.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

Reimplemented from **brathl::CProductErs** (p. 251).

References **brathl::CTools::Format()**.

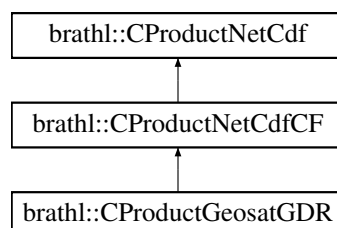
The documentation for this class was generated from the following files:

- ProductErsWAP.h
- ProductErsWAP.cpp

8.82 brathl::CProductGeosatGDR Class Reference

```
#include <ProductGeosatGDR.h>
```

Inheritance diagram for **brathl::CProductGeosatGDR**:



Public Member Functions

- **CProductGeosatGDR** ()
Empty CProductGeosatGDR (p. 253) ctor.
- **CProductGeosatGDR** (const std::string &path)
- **CProductGeosatGDR** (const CStringList &paths, bool check_only_first_file)
- virtual void **InitDateRef** ()
- virtual **~CProductGeosatGDR** ()
Destructor.

Additional Inherited Members

8.82.1 Detailed Description

Geosat GDR product management class.

Version

1.0

8.82.2 Constructor & Destructor Documentation

8.82.2.1 `brathl::CProductGeosatGDR::CProductGeosatGDR (const std::string & path) [inline]`

Creates new **CProductGeosatGDR** (p. 253) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.82.2.2 `brathl::CProductGeosatGDR::CProductGeosatGDR (const CStringList & paths, bool check_only_first_file) [inline]`

Creates new **CProductGeosatGDR** (p. 253) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

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

- ProductGeosatGDR.h

8.83 brathl::CProductGfo Class Reference

```
#include <ProductGfo.h>
```

Inherits `brathl::CProduct`.

Public Member Functions

- **CProductGfo** ()
Empty CProductGfo (p. 254) ctor.
- **CProductGfo** (const std::string &fileName)
- **CProductGfo** (const CStringList &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual **~CProductGfo** ()
Destructor.

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (CDataSet *dataSet, double deltaLat, double deltaLon)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (CField *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()

Protected Attributes

- std::string **m_timeStampMicrosecondFieldName**
- std::string **m_timeStampSecondFieldName**

8.83.1 Detailed Description

Ers product management class.

Version

1.0

8.83.2 Constructor & Destructor Documentation

8.83.2.1 brathl::CProductGfo::CProductGfo (const std::string & *fileName*)

Creates new **CProductGfo** (p. 254) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.83.2.2 brathl::CProductGfo::CProductGfo (const CStringList & *fileNameList*, bool *check_only_first_file*)

Creates new **CProductGfo** (p. 254) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.83.3 Member Function Documentation

8.83.3.1 bool brathl::CProductGfo::IsHighResolutionField (CField * *field*) [protected], [virtual]

Determines if a field object is a 'high resolution' array data For Jason, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 10.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

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

- ProductGfo.h
- ProductGfo.cpp

8.84 brathl::CProductJason Class Reference

```
#include <ProductJason.h>
```

Inherits brathl::CProduct.

Public Member Functions

- **CProductJason** ()
Empty CProductJason (p. 256) ctor.
- **CProductJason** (const std::string &fileName)
- **CProductJason** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductJason** ()
Destructor.

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (**CDataSet** *dataSet, double deltaLat, double deltaLon)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (**CField** *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()

Protected Attributes

- std::string **m_timeStampDayFieldName**
- std::string **m_timeStampMicrosecondFieldName**
- std::string **m_timeStampSecondFieldName**

8.84.1 Detailed Description

Jason product management class.

Version

1.0

8.84.2 Constructor & Destructor Documentation

8.84.2.1 brathl::CProductJason::CProductJason (const std::string & fileName)

Creates new **CProductJason** (p. 256) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.84.2.2 brathl::CProductJason::CProductJason (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductJason** (p. 256) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.84.3 Member Function Documentation

8.84.3.1 bool bratl::CProductJason::IsHighResolutionField (CField * *field*) [protected],[virtual]

Determines if a field object is a 'high resolution' array data For Jason, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 20.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

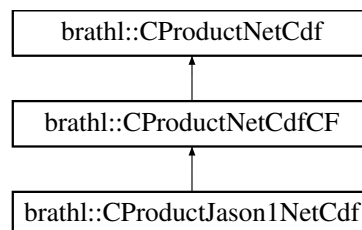
The documentation for this class was generated from the following files:

- ProductJason.h
- ProductJason.cpp

8.85 bratl::CProductJason1NetCdf Class Reference

```
#include <ProductJason1NetCdf.h>
```

Inheritance diagram for bratl::CProductJason1NetCdf:



Public Member Functions

- **CProductJason1NetCdf** ()
Empty CProductJason1NetCdf (p. 257) ctor.
- **CProductJason1NetCdf** (const std::string &path)
- **CProductJason1NetCdf** (const CStringList &paths, bool check_only_first_file)
- virtual void **InitDateRef** ()
- virtual ~**CProductJason1NetCdf** ()
Destructor.

Additional Inherited Members

8.85.1 Detailed Description

Jason-1 GDR (Native/Expertise) product management class.

Version

1.0

8.85.2 Constructor & Destructor Documentation

8.85.2.1 `brathl::CProductJason1NetCdf::CProductJason1NetCdf (const std::string & path)` `[inline]`

Creates new **CProductJason1NetCdf** (p. 257) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.85.2.2 bratl::CProductJason1NetCdf::CProductJason1NetCdf (const CStringList & paths, bool check_only_first_file)
[inline]

Creates new **CProductJason1NetCdf** (p. 257) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

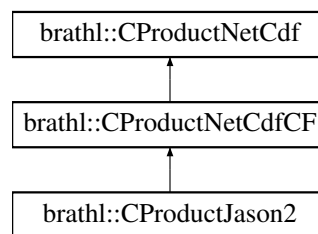
The documentation for this class was generated from the following file:

- ProductJason1NetCdf.h

8.86 bratl::CProductJason2 Class Reference

```
#include <ProductJason2.h>
```

Inheritance diagram for bratl::CProductJason2:



Public Member Functions

- **CProductJason2** ()
CIntMap (p. 234) *ctor.*
- **CProductJason2** (const std::string &fileName)
- **CProductJason2** (const CStringList &fileNameList, bool check_only_first_files)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual bool **HasCriteriaInfo** ()
- virtual void **InitCriteriaInfo** ()
- virtual void **InitDateRef** ()

Protected Member Functions

- void **Init** ()

Additional Inherited Members

8.86.1 Detailed Description

Mapping products management class.

Version

1.0

8.86.2 Constructor & Destructor Documentation

8.86.2.1 CProductJason2::CProductJason2 (const std::string & fileName)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.86.2.2 CProductJason2::CProductJason2 (const CStringList & fileNameList, bool check_only_first_files)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

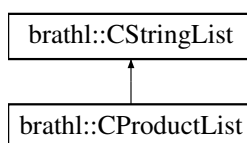
The documentation for this class was generated from the following files:

- ProductJason2.h
- ProductJason2.cpp

8.87 bratl::CProductList Class Reference

```
#include <Product.h>
```

Inheritance diagram for bratl::CProductList:



Public Member Functions

- bool **CheckFile** (const stringlist::iterator &it, bool netcdf_check)
- bool **CheckFiles** (bool onlyFirstFile=false, bool onlyFirstNetcdf=false)
- **CProductList** ()
Empty CProductList (p. 259) ctor.
- **CProductList** (const **CProductList** &o)
- **CProductList** (const std::string &fileName)
- **CProductList** (const **CStringList** &fileNameList)
- **CProductList** (const CStringArray &fileNameArray)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- bool **IsATP** () const
- bool **IsGenericNetCdf** () const
- bool **IsHdfOrNetcdfCodaFormat** ()
- bool **IsJason2** () const
- bool **IsNetCdfCFProduct** () const
- bool **IsNetCdfOrNetCdfCFProduct** () const
- bool **IsNetCdfProduct** () const
- bool **IsSameProduct** (const std::string &productClass, const std::string &productType)
- bool **IsYFX** () const

- bool **IsZFX** () const
 - **CProductList** & **operator=** (const **CProductList** &lst)
 - virtual ~**CProductList** ()
- Destructor.*

Static Public Member Functions

- static bool **IsHdfOrNetcdfCodaFormat** (coda_format format)

Public Attributes

- std::string **m_productClass**
- coda_format **m_productFormat**
- std::string **m_productType**
- std::string **mCodaProductClass**
- std::string **mCodaProductType**

8.87.1 Detailed Description

Product file list management class.

Version

1.0

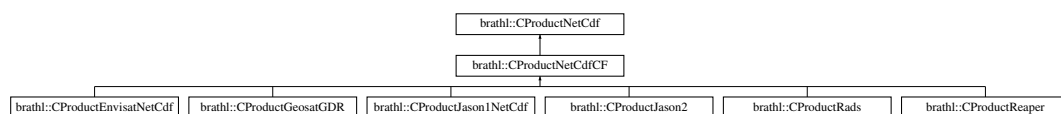
The documentation for this class was generated from the following files:

- Product.h
- Product.cpp

8.88 brathl::CProductNetCdf Class Reference

```
#include <ProductNetCdf.h>
```

Inheritance diagram for brathl::CProductNetCdf:



Public Member Functions

- void **AddDimsToReadOneByOne** (const CStringArray &value)
- virtual void **AddOffset** (double value, **CField** *field=NULL) override
- virtual bool **ApplyCriteria** (CStringList &filteredFileList, CProgressInterface *pi, const std::string &log_file="") override
- virtual bool **ApplyCriteriaCycle** (CCriterialInfo *criterialInfo) override
- virtual bool **ApplyCriteriaDatetime** (CCriterialInfo *criterialInfo) override
- virtual bool **ApplyCriteriaLatLon** (CCriterialInfo *criterialInfo) override
- virtual bool **ApplyCriteriaPass** (CCriterialInfo *criterialInfo) override
- virtual bool **ApplyCriteriaPassInt** (CCriterialInfo *criterialInfo) override
- virtual bool **ApplyCriteriaPassString** (CCriterialInfo *criterialInfo) override

- virtual void **CheckFileOpened** () override
- virtual CProduct * **Clone** () override
- virtual bool **Close** () override
- **CProductNetCdf** ()
 - Empty CProductNetCdf (p. 261) ctor.*
- **CProductNetCdf** (const std::string &fileName)
- **CProductNetCdf** (const CStringList &fileNameList, bool check_only_first_files)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
 - Dump fonction.*
- const CStringArray * **GetAxisDims** ()
- CStringArray * **GetComplementDims** ()
- virtual bool **GetDateMinMax** (CDatePeriod &datePeriodMinMax, CProgressInterface *pi=nullptr) override
- CStringArray * **GetDimsToReadOneByOne** ()
- **CExternalFilesNetCDF** * **GetExternalFile** ()
- virtual bool **GetForceReadDataOneByOne** () override
- virtual bool **GetLatLonMinMax** (CLatLonRect &latlonRectMinMax, CProgressInterface *pi=nullptr) override
- void **GetNetCdfDimensions** (const std::vector< CExpression > &expressions, CStringArray &commonDimNames)
- void **GetNetCdfDimensions** (const CExpression &expr, CStringArray &commonDimNames)
- void **GetNetCdfDimensions** (const CStringArray &fields, CStringArray &commonDimNames)
- void **GetNetCdfDimensions** (const std::vector< CExpression > &expressions, CStringArray &commonDimNames, const std::string &recordName)
- void **GetNetCdfDimensions** (const CExpression &expr, CStringArray &commonDimNames, const std::string &recordName)
- void **GetNetCdfDimensions** (const CStringArray &fields, CStringArray &commonDimNames, const std::string &recordName)
- void **GetNetCdfDimensionsWithoutAlgo** (const std::vector< CExpression > &expressions, CStringArray &commonDimNames, const std::string &recordName)
- void **GetNetCdfDimensionsWithoutAlgo** (const CExpression &expr, CStringArray &commonDimNames, const std::string &recordName)
- virtual int32_t **GetNumberOfRecords** (const std::string &dataSetName) override
- virtual int32_t **GetNumberOfRecords** () override
- virtual void **GetRecords** (CStringArray &array) override
- virtual bool **HasCriteriaInfo** () override
- virtual void **InitCriteriaInfo** () override
- void **InitDataset** ()
- virtual void **InitDateRef** () override
- void **InitLatLonFieldName** ()
- bool **IsApplyNetcdfProductInitialisation** ()
- bool **IsLatField** (CFieldNetCdf *field)
- bool **IsLonField** (CFieldNetCdf *field)
- virtual bool **IsOpened** () override
- virtual bool **IsOpened** (const std::string &fileName) override
- void **MustBeOpened** ()
- virtual void **NetCdfProductInitialization** (CProduct *from)
- virtual bool **NextRecord** ()
- virtual bool **PrevRecord** ()
- virtual void **ReadBratRecord** (int32_t iRecord) override
- **CFieldNetCdf** * **ReadDateCriteriaValue** (CFieldInfo &fieldInfo, CDate &date, bool wantMin=true)
- **CFieldNetCdf** * **ReadDoubleCriteriaValue** (CFieldInfo &fieldInfo, double &value, bool wantMin=true)
- virtual void **Rewind** () override
- void **SetApplyNetcdfProductInitialisation** (bool value)
- void **SetAxisDims** (const CStringArray &value)
- void **SetComplementDims** (const CStringArray &value)
- void **SetDimsToReadOneByOne** (const CStringArray &value)

- virtual void **SetForceReadDataOneByOne** (bool value) override
- virtual void **SetOffset** (double value) override
- virtual **~CProductNetCdf** ()

Destructor.

Static Public Member Functions

- static **CProductNetCdf * GetProductNetCdf** (CBratObject *ob, bool withExcept=true)
- static bool **IsProductNetCdf** (CBratObject *ob)

Static Public Attributes

- static const std::string **m_virtualRecordName** = "data"

Protected Member Functions

- virtual void **CreateFieldSets** ()
- void **DeleteExternalFile** ()
- void **DeleteFieldsToReadMap** ()
- virtual void **FillDescription** () override
- **CFieldNetCdf * FindCycleField** ()
- **CFieldNetCdf * FindLatField** ()
- **CFieldNetCdf * FindLonField** ()
- **CFieldNetCdf * FindPassField** ()
- **CFieldNetCdf * FindTimeField** ()
- void **Init** ()
- virtual void **InitInternalFieldName** (const std::string &dataSetName, **CStringList** &listField, bool convertDate=false) override
- virtual void **InitInternalFieldName** (**CStringList** &listField, bool convertDate=false) override
- virtual void **LoadFieldsInfo** () override
- virtual std::string **MakeInternalFieldName** (const std::string &dataSetName, const std::string &field) override
- virtual std::string **MakeInternalFieldName** (const std::string &field) override
- virtual bool **Open** () override
- virtual **CFieldNetCdf * Read** (CFieldInfo &fieldInfo, double &value, bool wantMin=true, const CAdjustValidMinMax &adjust_algo=CAdjustValidMinMax())
- virtual void **Read** (CFieldInfo &fieldInfo, std::string &value)
- virtual void **Read** (**CFieldNetCdf** *field, double &value)
- virtual void **Read** (**CFieldNetCdf** *field, CDoubleArray &vect)
- virtual void **Read** (**CFieldNetCdf** *field, **CExpressionValue** &value)
- virtual void **ReadAll** (**CFieldNetCdf** *field, const CAdjustValidMinMax &adjust_algo=CAdjustValidMinMax())
- virtual void **ReadAll** (**CFieldNetCdf** *field, **CExpressionValue** &value)
- virtual void **ReadBratFieldRecord** (const std::string &key)
- virtual void **ReadBratFieldRecord** (CField::CListField::iterator it) override
- virtual void **RewindEnd** () override
- virtual void **RewindInit** () override
- virtual void **RewindProcess** () override

Protected Attributes

- bool **m_applyNetcdfProductInitialisation**
- CStringArray **m_axisDims**
- CStringArray **m_complementDims**
- CStringArray **m_dimsToReadOneByOne**
- **CExternalFilesNetCDF * m_externalFile**
- **CObMap * m_fieldsToRead**
- bool **m_forceReadDataOneByOne**

8.88.1 Detailed Description

Netcdf product management class.

Version

1.0

8.88.2 Constructor & Destructor Documentation

8.88.2.1 brathl::CProductNetCdf::CProductNetCdf (const std::string & fileName)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.88.2.2 brathl::CProductNetCdf::CProductNetCdf (const CStringList & fileNameList, bool check_only_first_files)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.88.3 Member Data Documentation

8.88.3.1 CObMap* brathl::CProductNetCdf::m_fieldsToRead [protected]

Map of the fields to read (key : var name → **CFieldNetCdf** (p. 208) object) NB : **CFieldNetCdf** (p. 208) objects stored in this map have not to be delete (they are not a copy !!!)

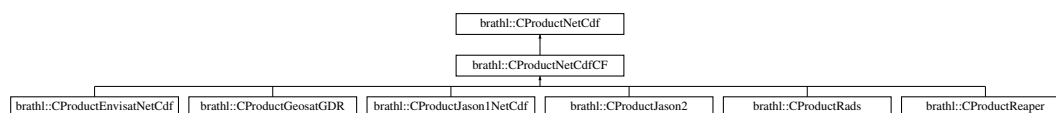
The documentation for this class was generated from the following files:

- ProductNetCdf.h
- ProductNetCdf.cpp

8.89 brathl::CProductNetCdfCF Class Reference

```
#include <ProductNetCdfCF.h>
```

Inheritance diagram for brathl::CProductNetCdfCF:



Public Member Functions

- virtual CProduct * **Clone** ()
- **CProductNetCdfCF** ()
Empty **CProductNetCdf** (p. 261) ctor.
- **CProductNetCdfCF** (const std::string &fileName)
- **CProductNetCdfCF** (const CStringList &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)

Dump fonction.

- virtual int32_t **GetNumberOfRecords** (const std::string &dataSetName)
- virtual int32_t **GetNumberOfRecords** ()
- virtual bool **NextRecord** ()
- virtual bool **PrevRecord** ()
- virtual void **Rewind** ()
- virtual ~**CProductNetCdfCF** ()

Destructor.

Static Public Member Functions

- static **CProductNetCdfCF** * **GetProductNetCdfCF** (CBratObject *ob, bool withExcept=true)
- static bool **IsProductNetCdfCF** (CBratObject *ob)

Protected Member Functions

- void **AdjustIndexesFromField** (CFieldNetCdf *field, bool next=true)
- void **AdjustIndexesToMin** (bool next=true)
- void **AdjustIndexesToMin** (CFieldNetCdf *field, bool next=true)
- bool **CheckEOF** ()
- void **Init** ()
- void **InitDimIndexes** (uint32_t value)
- virtual void **InitDimsIndexToMax** ()
- bool **IsAtBeginning** ()
- bool **NextFieldIndex** ()
- bool **PrevFieldIndex** ()
- virtual void **RewindEnd** ()
- virtual void **RewindInit** ()
- virtual void **RewindProcess** ()
- void **SetFieldIndex** ()
- void **SetFieldIndex** (CFieldNetCdf *field)

Protected Attributes

- bool **m_atBeginning**
- CIntMap **m_dimIds**
- CUIntMap **m_dimIndexes**
- CUIntMap **m_dimsCount**
- CUIntMap **m_dimValues**

Additional Inherited Members

8.89.1 Detailed Description

Netcdf product management class.

Version

1.0

8.89.2 Constructor & Destructor Documentation

8.89.2.1 brathl::CProductNetCdfCF::CProductNetCdfCF (const std::string & fileName)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.89.2.2 bratl::CProductNetCdfCF::CProductNetCdfCF (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductNetCdf** (p. 261) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.89.3 Member Data Documentation

8.89.3.1 bool bratl::CProductNetCdfCF::m_atBeginning [protected]

'At beginning" flag

Referenced by Dump().

8.89.3.2 CIntMap bratl::CProductNetCdfCF::m_dimIds [protected]

Map of the dimension's ids of the read fields (key : dim name → dim ids)

Referenced by Dump().

8.89.3.3 CUIntMap bratl::CProductNetCdfCF::m_dimsCount [protected]

Map of the dimension's ranges of the read fields (key : dim name → dim range)Array of the dimension count for reading (key : dim name → count)

Referenced by Dump().

8.89.3.4 CUIntMap bratl::CProductNetCdfCF::m_dimValues [protected]

Map of the dimension's values of the read fields (key : dim name → dim value)

Referenced by Dump().

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

- ProductNetCdfCF.h
- ProductNetCdfCF.cpp

8.90 bratl::CProductPodaac Class Reference

```
#include <ProductPodaac.h>
```

Inherits bratl::CProduct.

Public Member Functions

- **CProductPodaac** ()
Empty CProductPodaac (p. 266) ctor.
- **CProductPodaac** (const std::string &fileName)
- **CProductPodaac** (const CStringList &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual const std::string & **GetLabel** () const override
- virtual void **InitCriteriaInfo** ()

- virtual `~CProductPodaac ()`

Destructor.

Static Public Attributes

- static const std::string `m_J1SSHA_ATG_FILE` = "J1SSHA_ATG_FILE"
- static const std::string `m_J1SSHA_PASS_FILE` = "J1SSHA_PASS_FILE"
- static const std::string `m_TPSSHA_ATG_FILE` = "TPSSHA_ATG_FILE"
- static const std::string `m_TPSSHA_PASS_FILE` = "TPSSHA_PASS_FILE"

Protected Member Functions

- virtual void `InitDateRef ()`

8.90.1 Detailed Description

Ers product management class.

Version

1.0

8.90.2 Constructor & Destructor Documentation

8.90.2.1 brathl::CProductPodaac::CProductPodaac (const std::string & *fileName*)

Creates new **CProductPodaac** (p. 266) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.90.2.2 brathl::CProductPodaac::CProductPodaac (const CStringList & *fileNameList*, bool *check_only_first_file*)

Creates new **CProductPodaac** (p. 266) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

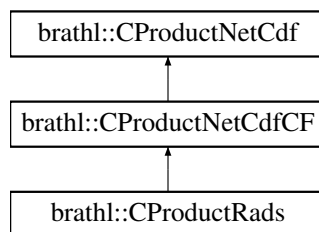
The documentation for this class was generated from the following files:

- ProductPodaac.h
- ProductPodaac.cpp

8.91 brathl::CProductRads Class Reference

```
#include <ProductRads.h>
```

Inheritance diagram for brathl::CProductRads:



Public Member Functions

- **CProductRads** ()
Empty CProductRads (p. 268) ctor.
- **CProductRads** (const std::string &fileName)
- **CProductRads** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr) override
Dump function.
- virtual std::string **GetLabelForCyclePass** () const override
- virtual ~**CProductRads** ()
Destructor.

Protected Member Functions

- virtual void **InitDateRef** () override

Additional Inherited Members

8.91.1 Detailed Description

RADS product management class.

Version

1.0

8.91.2 Constructor & Destructor Documentation

8.91.2.1 brathl::CProductRads::CProductRads (const std::string & fileName)

Creates new **CProductRads** (p. 268) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.91.2.2 brathl::CProductRads::CProductRads (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductRads** (p. 268) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

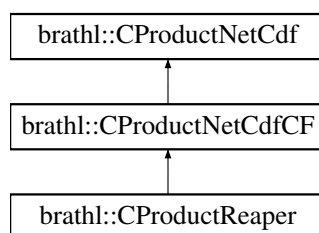
The documentation for this class was generated from the following files:

- ProductRads.h
- ProductRads.cpp

8.92 brathl::CProductReaper Class Reference

```
#include <ProductReaper.h>
```

Inheritance diagram for brathl::CProductReaper:



Public Member Functions

- **CProductReaper** ()
Empty CProductReaper (p. 269) ctor.
- **CProductReaper** (const std::string &path)
- **CProductReaper** (const CStringList &paths, bool check_only_first_files)
- virtual void **InitDateRef** ()
- virtual ~**CProductReaper** ()
Destructor.

Additional Inherited Members

8.92.1 Detailed Description

Reaper product management class.

Version

1.0

8.92.2 Constructor & Destructor Documentation

8.92.2.1 brathl::CProductReaper::CProductReaper (const std::string & path) [inline]

Creates new **CProductReaper** (p. 269) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.92.2.2 brathl::CProductReaper::CProductReaper (const CStringList & paths, bool check_only_first_files) [inline]

Creates new **CProductReaper** (p. 269) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

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

- ProductReaper.h

8.93 brathl::CProductRiverLake Class Reference

```
#include <ProductRiverLake.h>
```

Inherits brathl::CProduct.

Public Member Functions

- **CProductRiverLake** ()
Empty CProductRiverLake (p. 270) ctor.
- **CProductRiverLake** (const std::string &fileName)
- **CProductRiverLake** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual void **InitCriteriaInfo** ()
- virtual **~CProductRiverLake** ()
Destructor.

Static Public Attributes

- static const std::string **m_DAY_NAME** = "day"
- static const std::string **m_HOUR_NAME** = "hour"
- static const std::string **m_MINUTE_NAME** = "minute"
- static const std::string **m_MONTH_NAME** = "month"
- static const std::string **m_PROD_TYPE_RLA** = "RLA"
- static const std::string **m_PROD_TYPE_RLH** = "RLH"
- static const std::string **m_TIME_DESC** = "Time in seconds since 1950-01-01T00:00:00"
- static const std::string **m_TIME_NAME** = "time"
- static const std::string **m_TIME_UNIT** = "seconds since 1950-01-01T00:00:00"
- static const std::string **m_YEAR_NAME** = "year"

Protected Member Functions

- virtual void **InitDateRef** () override
- virtual void **InitInternalFieldNamesForCombinedVariable** (**CStringList** &listField, const std::string &record) override
- virtual bool **Open** () override
- virtual void **ReadBratFieldRecord** (CField::CListField::iterator it, bool &skipRecord) override

8.93.1 Detailed Description

River & Lake product management class.

Version

1.0

8.93.2 Constructor & Destructor Documentation

8.93.2.1 brathl::CProductRiverLake::CProductRiverLake (const std::string & fileName)

Creates new **CProductRiverLake** (p. 270) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.93.2.2 bratl::CProductRiverLake::CProductRiverLake (const CStringList & fileNameList, bool check_only_first_file)

Creates new **CProductRiverLake** (p. 270) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

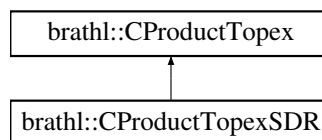
The documentation for this class was generated from the following files:

- ProductRiverLake.h
- ProductRiverLake.cpp

8.94 bratl::CProductTopex Class Reference

```
#include <ProductTopex.h>
```

Inheritance diagram for bratl::CProductTopex:



Public Member Functions

- **CProductTopex** ()
Empty CProductTopex (p. 271) ctor.
- **CProductTopex** (const std::string &fileName)
- **CProductTopex** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual const std::string & **GetLabel** () const override
- virtual void **InitCriteriaInfo** ()
- virtual ~**CProductTopex** ()
Destructor.

Static Public Attributes

- static const int32_t **m_ALTIMETER_POSEIDON** = 0
- static const int32_t **m_ALTIMETER_TOPEX** = 1
- static const std::string **m_PASS_FILE** = "MGDR_pass_file"
- static const std::string **m_SDR_PASS_FILE** = "SDR_pass_file"
- static const std::string **m_TOPEX_POSEIDON_HEADER** = "header"
- static const std::string **m_XNG_FILE** = "MGDR_crossover_point_file"

Protected Member Functions

- virtual void **AddInternalHighResolutionFieldCalculation** ()
- void **ComputeHighResolutionFields** (CDataSet *dataSet, double deltaLat, double deltaLon)
- virtual void **InitDateRef** ()
- virtual bool **IsHighResolutionField** (CField *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()
- virtual void **SetDeltaTimeHighResolution** (int32_t altimeterIndicator)

Protected Attributes

- std::string **m_altimeterIndicatorFieldName**
- std::string **m_timeStampDayFieldName**
- std::string **m_timeStampMicrosecondFieldName**
- std::string **m_timeStampMillisecondFieldName**

8.94.1 Detailed Description

Topex/Poseidon product management class.

Version

1.0

8.94.2 Constructor & Destructor Documentation

8.94.2.1 brathl::CProductTopex::CProductTopex (const std::string & *fileName*)

Creates new **CProductTopex** (p. 271) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.94.2.2 brathl::CProductTopex::CProductTopex (const CStringList & *fileNameList*, bool *check_only_first_file*)

Creates new **CProductTopex** (p. 271) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.94.3 Member Function Documentation

8.94.3.1 bool brathl::CProductTopex::IsHighResolutionField (CField * *field*) [protected], [virtual]

Determines if a field object is a 'high resolution' array data For Topex/Poseidon, to be a 'high resolution' field, all conditions below have to be true :

- the field object is not an instance of **CFieldBasic** (p. 206)
- the field has one dimension and the dimension is 10.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

Reimplemented in **brathl::CProductTopexSDR** (p. 274).

8.94.4 Member Data Documentation

8.94.4.1 `const int32_t brathl::CProductTopex::m_ALTIMETER_POSEIDON = 0` [static]

Altimeter Indicator. This element is computed for TOPEX and POSEIDON data. It indicates which altimeter is on at the time of the measurement. Value Definition: 0 = POSEIDON on, 1 = TOPEX on

8.94.4.2 `std::string brathl::CProductTopex::m_altimeterIndicatorFieldName` [protected]

Altimeter Indicator. This element is computed for TOPEX and POSEIDON data. It indicates which altimeter is on at the time of the measurement. Value Definition: 0 = POSEIDON on, 1 = TOPEX on

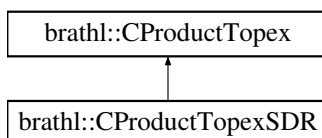
The documentation for this class was generated from the following files:

- ProductTopex.h
- ProductTopex.cpp

8.95 brathl::CProductTopexSDR Class Reference

```
#include <ProductTopexSDR.h>
```

Inheritance diagram for brathl::CProductTopexSDR:



Public Member Functions

- **CProductTopexSDR** ()
Empty CProductTopexSDR (p. 273) ctor.
- **CProductTopexSDR** (const std::string &fileName)
- **CProductTopexSDR** (const **CStringList** &fileNameList, bool check_only_first_file)
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- virtual const std::string & **GetLabel** () const override
- virtual ~**CProductTopexSDR** ()
Destructor.

Protected Member Functions

- virtual void **CheckConsistencyHighResolutionField** (**CFieldSetArrayDbI** *fieldSetArrayDbI)
- void **ComputeHighResolutionFields** (**CDataSet** *dataSet, double deltaLat, double deltaLon)
- virtual bool **IsHighResolutionField** (**CField** *field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()
- virtual void **PutFlatHighResolution** (**CDataSet** *dataSet, **CFieldSetArrayDbI** *fieldSetArrayDbI)
- virtual void **SetHighResolution** (**CField** *field)

Protected Attributes

- uint32_t **m_highRateNumHighResolutionMeasure**
- uint32_t **m_lowRateNumHighResolutionMeasure**

Additional Inherited Members

8.95.1 Detailed Description

Topex/Poseidon SDR product management class.

Version

1.0

8.95.2 Constructor & Destructor Documentation

8.95.2.1 `brathl::CProductTopexSDR::CProductTopexSDR (const std::string & fileName)`

Creates new **CProductTopexSDR** (p. 273) object

Parameters

<i>fileName</i>	[in] : file name to be connected
-----------------	----------------------------------

8.95.2.2 `brathl::CProductTopexSDR::CProductTopexSDR (const CStringList & fileNameList, bool check_only_first_file)`

Creates new **CProductTopexSDR** (p. 273) object

Parameters

<i>fileNameList</i>	[in] : list of file to be connected
---------------------	-------------------------------------

8.95.3 Member Function Documentation

8.95.3.1 `bool brathl::CProductTopexSDR::IsHighResolutionField (CField * field) [protected], [virtual]`

Determines if a field object is a 'high resolution' array data For Topex/Poseidon, to be a 'high resolution' field, all conditions below have to be true :

- **CProductTopex** (p. 271) rules (see **CProductTopex::IsHighResolutionField** (p. 273))
- the field has two dimensions and the first dimension is 10 or 5.

Parameters

<i>field</i>	[in] : field to be tested.
--------------	----------------------------

Reimplemented from **brathl::CProductTopex** (p. 273).

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

- ProductTopexSDR.h
- ProductTopexSDR.cpp

8.96 brathl::CPtrMap Class Reference

```
#include <List.h>
```

Inherits `mapptr`.

Public Member Functions

- **CPtrMap** (bool bDelete=true)
CPtrMap (p. 275) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump fonction.
- virtual bool **Erase** (CPtrMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual void * **Exists** (const std::string &key) const
- virtual void * **Insert** (const std::string &key, void *ptr, bool withExcept=true)
- virtual void **Insert** (const **CPtrMap** &ptrMap, bool withExcept=true)
- virtual void * **operator[]** (const std::string &key)
- virtual void **RemoveAll** ()
- virtual ~**CPtrMap** ()
CPtrMap (p. 275) dtor.

Protected Attributes

- bool **m_bDelete**

8.96.1 Detailed Description

a set of pointer management classes.

Version

1.0

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

- List.h
- List.cpp

8.97 brathl::CRecord Class Reference

```
#include <Field.h>
```

Inherits brathl::CBratObject.

Public Member Functions

- **CRecord** (**CRecordSet** *recordSet=NULL)
Ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- const std::string & **GetName** ()
- **CRecordSet** * **GetRecordSet** ()
- virtual ~**CRecord** ()
Dtor.

Protected Attributes

- **CRecordSet** * **m_recordSet**

8.97.1 Detailed Description

a set of record management classes.

Version

1.0

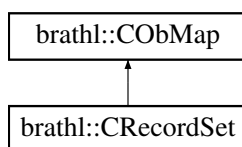
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.98 brathl::CRecordSet Class Reference

```
#include <Field.h>
```

Inheritance diagram for brathl::CRecordSet:



Public Member Functions

- **CRecordSet** (const std::string &name="", bool bDelete=true)
Ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump fonction.
- void **ExecuteExpression** (CExpression &expr, const std::string &recordName, **CExpressionValue** &expr←Value, CProduct *product=NULL)
- **CFieldSet** * **ExistsFieldSet** (const std::string &key)
- **CField** * **GetField** (CRecordSet::iterator it)
- **CFieldSet** * **GetFieldSet** (CRecordSet::iterator it)
- **CFieldSet** * **GetFieldSet** (const std::string &dataSetName, const std::string &fieldName)
- bool **IsFieldHasToBeExpanded** (CRecordSet::iterator it, const **CStringList** &listFieldExpandArray)
- bool **IsFieldHasToBeExpanded** (**CFieldSet** *fieldSet, const **CStringList** &listFieldExpandArray)
- virtual ~**CRecordSet** ()
Dtor.

Public Attributes

- std::string **m_name**

Additional Inherited Members

8.98.1 Detailed Description

a set of record fields value management classes.

Version

1.0

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

- Field.h
- Field.cpp

8.99 brathl::CRegisteredPass Class Reference

```
#include <ExternalFilesATP.h>
```

Inherits brathl::CBratObject.

Public Member Functions

- **CRegisteredPass** (**CRegisteredPass** &p)
- const **CRegisteredPass** & **operator=** (**CRegisteredPass** &p)
- void **Set** (**CRegisteredPass** &p)

Public Attributes

- double **m_beginDate**
- uint32_t **m_cycle**
- uint32_t **m_cycleIndex**
- uint32_t **m_nbData**
- uint32_t **m_pass**
- uint32_t **m_startPoint**

8.99.1 Detailed Description

External files access.

Version

1.0

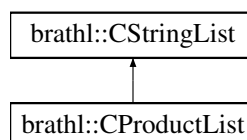
The documentation for this class was generated from the following file:

- ExternalFilesATP.h

8.100 brathl::CStringList Class Reference

```
#include <List.h>
```

Inheritance diagram for brathl::CStringList:



Public Member Functions

- virtual bool **Complement** (const **CStringList** &array, **CStringList** &complement) const
- **CStringList** ()
*Empty **CStringList** (p. 278) ctor.*
- **CStringList** (const **CStringList** &list)
- **CStringList** (const **stringlist** &list)
- **CStringList** (const CStringArray &vect)
- **CStringList** (const std::vector< std::string > &vect)
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump function.
- virtual void **Erase** (const std::string &str)
- virtual void **Erase** (CStringList::iterator it)
- virtual bool **Exists** (const std::string &str) const
- virtual bool **ExistsNoCase** (const std::string &str) const
- virtual void **ExtractKeys** (const std::string &str, const std::string &delim, bool bRemoveAll=true)
- virtual void **ExtractStrings** (const std::string &str, const char delim, bool bRemoveAll=true)
- virtual void **ExtractStrings** (const std::string &str, const std::string &delim, bool bRemoveAll=true)
- virtual int32_t **FindIndex** (const std::string &str, bool compareNoCase=false) const
- virtual void **Insert** (const **CStringList** &list, bool bEnd=true)
- virtual void **Insert** (const std::string &str, bool bEnd=true)
- virtual void **Insert** (const CStringArray &vect, bool bEnd=true)
- virtual void **Insert** (const std::vector< std::string > &vect, bool bEnd=true)
- virtual void **Insert** (const **stringlist** &lst, bool bEnd=true)
- virtual void **InsertUnique** (const std::string &str, bool bEnd=true)
- virtual void **InsertUnique** (const **CStringList** &lst, bool bEnd=true)
- virtual void **InsertUnique** (const CStringArray *vect, bool bEnd=true)
- virtual void **InsertUnique** (const CStringArray &vect, bool bEnd=true)
- virtual void **InsertUnique** (const std::vector< std::string > &vect, bool bEnd=true)
- virtual void **InsertUnique** (const **stringlist** &lst, bool bEnd=true)
- virtual bool **Intersect** (const **CStringList** &array, **CStringList** &intersect) const
- virtual const **CStringList** & **operator=** (const **CStringList** &lst)
- virtual const **CStringList** & **operator=** (const CStringArray &vect)
- virtual const **CStringList** & **operator=** (const std::vector< std::string > &vect)
- virtual const **CStringList** & **operator=** (const **stringlist** &lst)
- virtual void **RemoveAll** ()
- virtual std::string **Tostring** (const std::string &delim=",", bool useBracket=true) const
- virtual ~**CStringList** ()

Destructor.

8.100.1 Detailed Description

A std::list of strings management class.

Version

1.0

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

- List.h
- List.cpp

8.101 brathl::CStringMap Class Reference

```
#include <List.h>
```

Inherits mapstring.

Public Member Functions

- **CStringMap** ()
 - CStringMap* (p. 279) *ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
 - Dump fonction.*
- virtual bool **Erase** (CStringMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual std::string **Exists** (const std::string &key) const
- virtual void **GetKeys** (CStringArray &keys, bool bRemoveAll=true) const
- virtual std::string **Insert** (const std::string &key, const std::string &str, bool withExcept=true)
- virtual void **Insert** (const **CStringMap** &strmap, bool withExcept=true)
- virtual std::string **IsValue** (const std::string &value)
- virtual void **RemoveAll** ()
- virtual ~**CStringMap** ()
 - CStringMap* (p. 279) *dtor.*

8.101.1 Detailed Description

a set of std::string value management classes.

Version

1.0

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

- List.h
- List.cpp

8.102 brathl::CTools Class Reference

```
#include <Tools.h>
```

Static Public Member Functions

- static double **Abs** (double X)
- static std::string **AbsolutePath** (const std::string &partialPath)
- static double **ACos** (double X)
- static double **ACosD** (double X)
- static double **And** (double X, double Y)
- static bool **AreValidMercatorLatitude** (double lat)
- static std::string **BeforeFirst** (const std::string &str, const char ch)
- static double **BitwiseAnd** (double X, double Y)
- static double **BitwiseNot** (double X)
- static double **BitwiseOr** (double X, double Y)
- static bool **CastValue** (int32_t &Dest, const double Source)
- static double **Ceil** (double X)

- static int **Compare** (double X, double Y, double compareEpsilon=CTools::m_CompareEpsilon)
- static bool **Compare** (const char *str1, const char *str2)
- static double **Cos** (double X)
- static double **CosD** (double X)
- static double **Deg2Rad** (double X)
- static double **DistanceKmOnUnitSphere** (double lat1, double long1, double lat2, double long2)
- static double **DistanceOnUnitSphere** (double lat1, double long1, double lat2, double long2)
- static double **Divide** (double X, double Y)
- static void **DoIncrementalStats** (double NewValue, double &Count, double &Mean, double &StdDev, double &Min, double &Max)
- static std::string **DoubleToStr** (double d, int32_t precision=10)
- static double **Exp** (double X)
- static std::string **ExpandShellVar** (const std::string &value)
- static std::string **ExpandVariables** (const std::string &valueIn, const std::map< std::string, std::string > *varValues, bool recurse=false, char beginning= '%', uint32_t *numberVarsExpanded=NULL, bool withExcept=false, std::string *errorMsg=NULL)
- static std::string **ExpandVariables** (const std::string &valueIn, const std::map< std::string, std::string > *varValues, const std::map< std::string, std::string > *fieldAliases, bool recurse=false, char beginning= '%', uint32_t *numberVarsExpanded=NULL, bool withExcept=false, std::string *errorMsg=NULL)
- static void **ExtractVector** (const double *vectorIn, uint32_t *shape, uint32_t nDims, uint32_t *start, uint32_t *edges, double *vectorOut)
- static bool **FileExists** (const std::string &Name)
- static std::string **FileExtension** (const std::string &fileName)
- static void **FinalizeIncrementalStats** (double Count, double &Mean, double &StdDev, double &Min, double &Max, double DefaultValue=m_defaultValueDOUBLE)
- static void **Find** (const std::string &inText, const std::string ®exPattern, std::vector< std::string > &stringFound)
- static void **FindAliases** (const std::string &inText, std::vector< std::string > &aliasesFound, bool onlyName=false, const std::string &begining="%", bool recurse=false, const std::map< std::string, std::string > *varValues=NULL, const std::map< std::string, std::string > *fieldAliases=NULL, bool withExcept=false, std::string *errorMsg=NULL)
- static std::string **FindDataFile** (const std::string &Name)
- static std::string **FindFileInPath** (const std::string &filename, const std::string &path)
- static int32_t **FindNoCase** (const std::string &src, const std::string &findWhat, uint32_t pos=0)
- static int32_t **FindNoCase** (const char *src, const char *findWhat, uint32_t pos=0)
- static void **FindWord** (const std::string &inText, std::vector< std::string > &wordsFound)
- static std::string **FloatToStr** (float f, int32_t precision=10)
- static double **Floor** (double X)
- static int32_t static std::string **Format** (size_t size, const char *format,...) __attribute__((format(printf
- static int32_t static std::string static std::string **Format** (const char *format,...) __attribute__((format(printf
- static int32_t static std::string static std::string static std::string **Format** (size_t size, const char *format, va_list args)
- static double **Frac** (double value)
- static std::string **GetInternalDataDir** ()
- static uint32_t **GetProductValues** (uint32_t *shape, uint32_t nbDims)
- static double **Int** (double dValue)
- static std::string **IntToStr** (int32_t i)
- static double **IsBounded** (double Min, double X, double Max)
- static double **IsBoundedStrict** (double Min, double X, double Max)
- static double **IsDefaultFloat** (double X)
- static bool **IsEmpty** (const char *pstrString)
- static bool **IsEven** (uint32_t value)
- static bool **IsEven** (int32_t value)

- static int **IsInf** (double X)
- static bool **IsLongitudeCircular** (double min, double max, double compareEpsilon=CTools::m_CompareEpsilon)
- static int **IsNan** (double X)
- static bool **IsOdd** (uint32_t value)
- static bool **IsOdd** (int32_t value)
- static bool **LoadAndCheckUdUnitsSystem** (std::string &errorMsg)
- static double **Log** (double X)
- static double **Log10** (double X)
- static std::string **LongToStr** (int64_t i)
- static std::string **MakeCorrectPath** (const std::string &path)
- static double **Max** (double X1, double X2)
- static double **Min** (double X1, double X2)
- static double **Minus** (double X, double Y)
- static double **Mod** (double X, double Y)
- static double **Multiply** (double X, double Y)
- static double **NormalizeLongitude** (double Floor, double Longitude)
- static double **Or** (double X, double Y)
- static double **Plus** (double X, double Y)
- static double **Pow** (double X, double Y)
- static double **Rad2Deg** (double X)
- static char * **RemoveAllSpaces** (char *str)
- static std::string **RemoveCharSurroundingNumber** (const std::string &str, const char c1='(', const char c2=')')
- static std::string **Replace** (const std::string &inText, const std::string ®exPattern, const std::string &replaceString)
- static void **ReplaceAliases** (const std::string &in, std::string &out, std::vector< std::string > *aliases=NULL)
- static void **ReplaceAliases** (const std::string &in, const std::string &replacedBy, std::string &out, std::vector< std::string > *aliases=NULL)
- static std::string **ReplaceString** (const std::string &inText, const std::vector< std::string > &findString, const std::vector< std::string > &replaceWords)
- static std::string **ReplaceWord** (const std::string &inText, const std::vector< std::string > &findWords, const std::vector< std::string > &replaceWords)
- static std::string **ReplaceWord** (const std::string &inText, const std::string &findWords, const std::string &replaceWords)
- static int32_t **RFindNoCase** (const std::string &src, const std::string &findWhat, uint32_t pos=0)
- static int32_t **RFindNoCase** (const char *src, const char *findWhat, uint32_t pos=0)
- static double **Rnd** (double value, double precision)
- static double **Round** (double value)
- static void **SetInternalDataDir** (const std::string &DataDir)
- static double **Sign** (double X)
- static double **Sin** (double X)
- static double **Sinc** (double x)
- static double **SinD** (double X)
- static std::string **SlashesDecode** (const std::string &str, const std::string &exclude="", bool decodeLiterals=true)
- static std::string **SlashesEncode** (const std::string &str, const std::string &exclude="", const std::string &literals="", bool hexadecimal=true)
- static int32_t **snprintf** (char *str, size_t size, const char *format,...) __attribute__((format(printf, 3, 4)))
- static double **Sqr** (double X)
- static double **Sqrt** (double X)
- static int32_t **StrCaseCmp** (const char *str1, const char *str2)
- static bool **StringCompare** (const std::string &s1, const std::string &s2)
- static std::string **StringRemoveAllSpaces** (const std::string &str)
- static std::string **StringReplace** (const std::string &str, char c, char replaceBy)

- static std::string **StringReplace** (const std::string &str, const std::string &c, const std::string &replaceBy, bool compareNoCase=false)
- static void **StringToAlias** (const std::string &in, std::string &out, const char beginning)
- static std::string **StringToLower** (const std::string &str)
- static std::string **StringToUpper** (const std::string &str)
- static std::string **StringTrim** (const std::string &str)
- static double **StrToDouble** (const std::string &value)
- static float **StrToFloat** (const std::string &value)
- static int32_t **StrToInt32** (const std::string &s)
- static int64_t **StrToInt64** (const std::string &s)
- static int64_t **StrToLong** (const std::string &s)
- static uint64_t **StrToUInt64** (const std::string &s)
- static void **SwapValue** (int32_t &value)
- static void **SwapValue** (int16_t &value)
- static void **SwapValue** (float &value)
- static void **SwapValue** (double &value)
- static double **Tan** (double X)
- static double **TanD** (double X)
- static char * **ToLower** (char *str)
- static char **ToLower** (const char chr)
- static std::string **Tostring** (const char *s, size_t len=std::string::npos)
- static char * **ToUpper** (char *str)
- static char **ToUpper** (const char chr)
- static std::string **TrailingZeroesTrim** (const std::string &Text, bool dotTrim=true)
- static char * **Trim** (char *str)
- static double **UnaryMinus** (double X)
- static double **UnaryNot** (double X)
- static double **UnconvertLat** (const std::string &value)
- static double **UnconvertLon** (const std::string &value, bool normalize=true)
- static int32_t **VectorContiguousBlock** (uint32_t ndims, const uint32_t *const shape, const uint32_t *const edges, uint32_t *const countContinuousBlock)
- static uint32_t **VectorOffset** (uint32_t *shape, uint32_t ndims, const uint32_t *coord)
- static bool **Xor** (bool p, bool q)

Static Public Attributes

- static const double **m_CompareEpsilon** = 1.0E-70
- static const char **m_defaultValueCHAR** = '\0'
default values for chars
- static const double **m_defaultValueDOUBLE** = 18446744073709551616.0
default values for double
- static const float **m_defaultValueFLOAT** = 18446744073709551616.0F
default values for float
- static const int16_t **m_defaultValueINT16** = 0xFFFF
default values for int 16 bits
- static const int32_t **m_defaultValueINT32** = 0xFFFFFFFF
default values for int 32 bits
- static const int64_t **m_defaultValueINT64** = 0xFFFFFFFFFFFFFFFFLL
default values for unsigned int 64 bits
- static const int8_t **m_defaultValueINT8** = 0xFF
default values for int 8 bits
- static const char * **m_defaultValueString** = ""
default values for std::string

- static const uint16_t **m_defaultValueUINT16** = 0xFFFFU
default values for unsigned int 16 bits
- static const uint32_t **m_defaultValueUINT32** = 0xFFFFFFFFU
default values for unsigned int 32 bits
- static const uint64_t **m_defaultValueUINT64** = 0xFFFFFFFFFFFFFFFFULL
default values for unsigned int 64 bits
- static const uint8_t **m_defaultValueUINT8** = 0xFFU
default values for unsigned int 8 bits
- static const double **m_deltaLatitudeMercator** = 1.0E-7

8.102.1 Detailed Description

Tools management class.

This class provides various static utility methods

Version

1.0

8.102.2 Member Function Documentation

8.102.2.1 double brathl::CTools::Abs (double *X*) [static]

Find the absolute value of a number. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

8.102.2.2 std::string brathl::CTools::AbsolutePath (const std::string & *partialPath*) [static]

Creates an absolute or full path name for the specified relative path name.

- change path separator in a suitable path separator ('\' or '/' depending on the system)
- skip trailing "../", if any
- remove back references: translate dir1/../dir2 to dir2

Parameters

<i>in</i>	<i>partialPath</i>	: the relative path
-----------	--------------------	---------------------

Returns

the absolute path name, or empty std::string if there is an error (for example, if the value passed in relPath includes a drive letter that is not valid or cannot be found, or if the length of the created absolute path name is greater than the BRATHL_PATH_MAX defined in **brathl.h** (p. 315))

8.102.2.3 double brathl::CTools::ACos (double *X*) [static]

Do the arc cosine of a number expressed in radians. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

Referenced by ACosD().

8.102.2.4 double bratl::CTools::ACosD (double X) [static]

Do the arc cosine of a number expressed in degrees. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

References ACos(), and Deg2Rad().

8.102.2.5 double bratl::CTools::And (double X, double Y) [static]

Do a logical and on two numbers. Takes default values into account

Parameters

in	X	: Number involved
in	Y	: Number involved

Returns

Result of operation

8.102.2.6 double bratl::CTools::BitwiseAnd (double X, double Y) [static]

Do a bitwise AND operation an integer. The numbers are taken as signed integers (int32_t). Then a bitwise AND is computed and the integer is converted back to a float. If the parameters are default values or do not fall in integer range, a default value is returned.

Parameters

in	X	: Number involved
in	Y	: Number involved

Returns

Result of operation

8.102.2.7 double bratl::CTools::BitwiseNot (double X) [static]

Complement an integer. The number is taken as a signed integer (int32_t). Then a bitwise not is computed and the integer is converted back to a float. If the parameter is a default values or do not fall in integer range, a default value is returned.

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Complemented number

8.102.2.8 double brathl::CTools::BitwiseOr (double X, double Y) [static]

Do a bitwise OR operation an integer. The numbers are taken as signed integers (int32_t). Then a bitwise OR is computed and the integer is converted back to a float. If the parameters are default values or do not fall in integer range, a default value is returned.

Parameters

in	X	: Number involved
in	Y	: Number involved

Returns

Result of operation

8.102.2.9 double brathl::CTools::Ceil (double X) [static]

Find the integral value part over of a number. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.10 double brathl::CTools::Cos (double X) [static]

Do the cosine of a number expressed in radians. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.11 double brathl::CTools::CosD (double X) [static]

Do the cosine of a number expressed in degrees. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.12 double brathl::CTools::Deg2Rad (double X) [static]

Convert degrees to radians. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
----	----------	-------------------

Returns

Result of operation

Referenced by ACosD(), and TanD().

8.102.2.13 `double brathl::CTools::Divide (double X, double Y) [static]`

Divide two numbers. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
in	<i>Y</i>	: Number involved

Returns

Result of operation

8.102.2.14 `void brathl::CTools::DoIncrementalStats (double NewValue, double & Count, double & Mean, double & StdDev, double & Min, double & Max) [static]`

Do incremental statistics. Incremental statistics are done to avoid memory consumption needed when we do 'classical' stats: an array of all the values involved with statistics must be kept before computing them. After first call to this the parameters must not be modified until end of statistics or result will be unpredictable.

Parameters

in	<i>NewValue</i>	: New value to take into account for statistics. Only valid values are kept; valid values are those different from default value (#IsDefaultValue#)
	<i>in/out</i>	Count : number of valid data used for stats. Valid data is a number which is not a default value. On first call, this parameter must be 0 or a default value. And it is not modified since the first valid value.
	<i>in/out</i>	Mean : Incremental mean
	<i>in/out</i>	StdDev : Temporary value used to compute standard deviation
	<i>in/out</i>	Min : Minimum value
	<i>in/out</i>	Max : Maximum value

8.102.2.15 `std::string brathl::CTools::DoubleToStr (double d, int32_t precision = 10) [static]`

Convert an double to std::string

Parameters

in	<i>value</i>	: double to be converted
----	--------------	--------------------------

Returns

coanverted value or empty std::string if no possible conversion.

8.102.2.16 `double brathl::CTools::Exp (double X) [static]`

Find exponential of a number. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

References `IsInf()`.

8.102.2.17 `std::string brathl::CTools::ExpandShellVar (const std::string & value) [static]`

Expands shell variables (i.e. `${HOME}`). If the '\$' character is preceded by '\', it's taken into account as a common character and not as a shell variable identifier. Shell variables beginning by '+' are expanded in uppercase. Shell variables beginning by '-' are expanded in lowercase.

Parameters

<i>in</i>	<i>value</i>	: The <code>std::string</code> to expand
-----------	--------------	--

Returns

the newly expanded `std::string`.

References `ExpandVariables()`.

Referenced by `brathl::CParameter::AddValue()`.

8.102.2.18 `std::string brathl::CTools::ExpandVariables (const std::string & valueIn, const std::map< std::string, std::string > * varValues, bool recurse = false, char beginning = ' % ', uint32_t * numberVarsExpanded = NULL, bool withExcept = false, std::string * errorMsg = NULL) [static]`

Expand variables (i.e. `%{VAR}`). If the '%' character is preceded by '\', it's taken into account as a common character and not as a variable identifier. Variables beginning by '+' are expanded in uppercase. Variables beginning by '-' are expanded in lowercase.

Parameters

<i>in</i>	<i>value</i>	: The <code>std::string</code> to expand
<i>in</i>	<i>VarValues</i>	: The values of the variables. If NULL, the environment variables are taken.
<i>in</i>	<i>Beginning</i>	: Char identifying the beginning of a var reference
<i>in</i>	<i>Recurse</i>	: If true, variable expanded can contain references to other variables which are then expanded.

Returns

the newly expanded `std::string`.

Referenced by `ExpandShellVar()`, and `brathl::CParameter::SetAliases()`.

8.102.2.19 `bool brathl::CTools::FileExists (const std::string & Name) [static]`

Indicates if a file exists

Parameters

<i>in</i>	<i>Name</i>	: File name
-----------	-------------	-------------

Returns

Returns true if file exists and is readable

8.102.2.20 `std::string bratl::CTools::FileExtension (const std::string & fileName) [static]`

Gets a file name extension.

Parameters

in	<i>filename</i>	: file name
----	-----------------	-------------

Returns

the extension, or empty std::string if none

8.102.2.21 void brathl::CTools::FinalizeIncrementalStats (double *Count*, double & *Mean*, double & *StdDev*, double & *Min*, double & *Max*, double *DefaultValue* = **m_defaultValueDOUBLE**) [static]

Terminates incremental statistics. Computes the final value of standard deviation

Parameters

in	<i>Count</i>	: number of valid data used for stats. If count is 0 or default value, all other output parameters are set to default value.
	<i>in/out</i>	Mean : Computed mean or default value (see Count)
	<i>in/out</i>	StdDev : On output, actual value of standard deviation
	<i>in/out</i>	Min : Computed min or default value (see Count)
	<i>in/out</i>	Max : Computed max or default value (see Count)
in	<i>DefaultValue</i>	: Default value wanted Value to put in output parameters if no stats can be done (no valid data: count is 0 or default value m_defaultValueDOUBLE (p. 283)#).

8.102.2.22 std::string brathl::CTools::FindDataFile (const std::string & *Name*) [static]

Finds a file path known only by its name. The path is retrieved from compilation (intallation prefix) or by environment variable.

Parameters

in	<i>Name</i>	: File name
----	-------------	-------------

Returns

Returns the path of found file or an empty std::string if not found

8.102.2.23 std::string brathl::CTools::FindFileInPath (const std::string & *filename*, const std::string & *path*) [static]

Finds a file location known only by its name using the give path. The path should be similar to what can be used for the PATH environment variable on the current system.

Parameters

in	<i>filename</i>	: File name
in	<i>path</i>	: Search path

Returns

Returns the full path to the file or an empty std::string if not found

8.102.2.24 double brathl::CTools::Floor (double *X*) [static]

Find the integral value part below of a number. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

8.102.2.25 `std::string bratl::CTools::Format (size_t size, const char * format, ...) [static]`

Write formatted data to a `std::string`. WARNING : this method use `vsprintf` if `vsprintf` is defined, otherwise `vsprintf` is used and 'size' parameter is ignored

Parameters

<i>in</i>	<i>size</i>	: maximum number of characters to store
<i>in</i>	<i>format</i>	: format-control <code>std::string</code>
<i>in</i>	<i>...</i>	: optional arguments

Returns

formatted `std::string`

Referenced by `bratl::CDate::AsString()`, `bratl::BuildExistingInternalFileKind()`, `bratl::CFileParams::CheckCount()`, `bratl::CDate::CvDate()`, `bratl::CDoubleMap::Dump()`, `bratl::CObDoubleMap::Dump()`, `bratl::CDoublePtrDoubleMap::Dump()`, `bratl::CDataSet::EraseFieldSet()`, `Format()`, `bratl::CBratAlgorithmGeosVelGrid::GetInputParamDesc()`, `bratl::CBratAlgorithmGeosVelAtp::GetInputParamDesc()`, `bratl::CBratAlgoFilterMedian1D::GetInputParamDesc()`, `bratl::CBratAlgoFilterLoess1D::GetInputParamDesc()`, `bratl::CBratAlgoFilterLoess2D::GetInputParamDesc()`, `bratl::CBratAlgoFilterMedian2D::GetInputParamDesc()`, `bratl::CBratAlgorithmGeosVelGrid::GetInputParamFormat()`, `bratl::CBratAlgorithmGeosVelAtp::GetInputParamFormat()`, `bratl::CBratAlgoFilterMedian1D::GetInputParamFormat()`, `bratl::CBratAlgoFilterLoess2D::GetInputParamFormat()`, `bratl::CBratAlgoFilterLoess1D::GetInputParamFormat()`, `bratl::CBratAlgoFilterMedian2D::GetInputParamFormat()`, `bratl::CBratAlgorithmGeosVelGrid::GetInputParamUnit()`, `bratl::CBratAlgorithmGeosVelAtp::GetInputParamUnit()`, `bratl::CBratAlgoFilterMedian1D::GetInputParamUnit()`, `bratl::CBratAlgoFilterMedian2D::GetInputParamUnit()`, `bratl::CBratAlgoFilterLoess2D::GetInputParamUnit()`, `bratl::CBratAlgoFilterLoess1D::GetInputParamUnit()`, `bratl::CParameter::GetValue()`, `bratl::CUIntMap::Insert()`, `bratl::CDataSet::InsertFieldSet()`, `bratl::CProductErsWAP::IsHighResolutionField()`, `bratl::CFile::Open()`, `bratl::CFile::ReadToBuffer()`, `bratl::CBratAlgoFilterLanczos1D::Run()`, `bratl::CBratAlgoFilterGaussian1D::Run()`, `bratl::CBratAlgoFilterMedian1D::Run()`, `bratl::CBratAlgoFilterLoess1D::Run()`, `bratl::CDatePeriod::SetFrom()`, `bratl::CDatePeriod::SetTo()`, `SlashesDecode()`, `SlashesEncode()`, `bratl::CFile::WriteChar()`, `bratl::CFile::WriteFromBuffer()`, and `bratl::CFile::WriteString()`.

8.102.2.26 `std::string bratl::CTools::Format (const char * format, ...) [static]`

Write formatted data to a `std::string`. WARNING : this method use `vsprintf` if `vsprintf` is defined, otherwise `vsprintf` is used and 'size' parameter is ignored

Parameters

<i>in</i>	<i>format</i>	: format-control <code>std::string</code>
<i>in</i>	<i>...</i>	: optional arguments

Returns

formatted `std::string`

References `Format()`.

8.102.2.27 `std::string bratl::CTools::Format (size_t size, const char * format, va_list args) [static]`

Write formatted data to a `std::string`. WARNING : this method use `vsprintf` if `vsprintf` is defined, otherwise `vsprintf` is used and 'size' parameter is ignored

Parameters

in	<i>size</i>	: maximum number of characters to store
in	<i>format</i>	: format-control std::string
in	<i>args</i>	: optional arguments

Returns

formatted std::string

8.102.2.28 std::string brathl::CTools::GetInternalDataDir () [static]

Returns the constant data directory defined at compilation time, by environment variable, or set by application.

Returns

Returns the path of found file or an empty std::string if not found

8.102.2.29 std::string brathl::CTools::IntToStr (int32_t i) [static]

Convert an int to std::string

Parameters

in	<i>value</i>	: int to be converted
----	--------------	-----------------------

Returns

coconverted value or empty std::string if no possible conversion.

8.102.2.30 double brathl::CTools::IsBounded (double Min, double X, double Max) [static]

Indicates if a number is comprised between two others. Takes default values into account

Parameters

in	<i>Min</i>	: Lower bound
in	<i>X</i>	: Number involved
in	<i>Max</i>	: Upper bound

Returns

Result of operation: 0 if not $\text{Min} \leq X \leq \text{Max}$.

8.102.2.31 double brathl::CTools::IsBoundedStrict (double Min, double X, double Max) [static]

Indicates if a number is comprised between two others. Takes default values into account

Parameters

in	<i>Min</i>	: Lower bound
in	<i>X</i>	: Number involved
in	<i>Max</i>	: Upper bound

Returns

Result of operation: 0 if not $\text{Min} < X < \text{Max}$.

8.102.2.32 double brathl::CTools::IsDefaultFloat (double X) [static]

Checks a default value.

Parameters

in	X	: Number involved
----	---	-------------------

Returns

0.0 if X is not a default value, 1.0 otherwise

8.102.2.33 int32_t brathl::CTools::IsInf (double X) [static]

Indicates if a number is infinite.

Parameters

in	X	: Number involved
----	---	-------------------

Returns

0 if X in finite 1 if infinite

Referenced by Exp(), Pow(), Sqr(), and Tan().

8.102.2.34 int32_t brathl::CTools::IsNan (double X) [static]

Indicates if a value is a valid number.

Parameters

in	X	: Number involved
----	---	-------------------

Returns

0 if X is valid, 1 if X is not a number

Referenced by Tan().

8.102.2.35 double brathl::CTools::Log (double X) [static]

Find the natural logarithm of a number. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.36 double brathl::CTools::Log10 (double X) [static]

Find the decimal logarithm of a number. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.37 `std::string brathl::CTools::MakeCorrectPath (const std::string & path) [static]`

Cleans a path variable

- change path separator in a suitable path separator ('\' or '/' depending on the system)
- skip trailing "../.", if any
- remove back references: translate dir1/./dir2 to dir2

Parameters

<i>in</i>	<i>path</i>	: The std::string to clean
-----------	-------------	----------------------------

Returns

the newly cleaned std::string.

8.102.2.38 `double brathl::CTools::Max (double X1, double X2) [static]`

Find the maximum value of two numbers. Takes default values into account

Parameters

<i>in</i>	<i>X1</i>	: Number involved
<i>in</i>	<i>X2</i>	: Number involved

Returns

Result of operation

Referenced by brathl::CCriteriaLatLon::GetMinOrMaxLon().

8.102.2.39 `double brathl::CTools::Min (double X1, double X2) [static]`

Find the minimum value of two numbers. Takes default values into account

Parameters

<i>in</i>	<i>X1</i>	: Number involved
<i>in</i>	<i>X2</i>	: Number involved

Returns

Result of operation

Referenced by brathl::CCriteriaLatLon::GetMinOrMaxLon().

8.102.2.40 `double brathl::CTools::Minus (double X, double Y) [static]`

Subtracts one number from another. TAKES default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
<i>in</i>	<i>Y</i>	: Number involved

Returns

Result of operation

References m_defaultValueDOUBLE.

8.102.2.41 `double brathl::CTools::Mod (double X, double Y) [static]`

Find the modulus of a number divided by another. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
in	<i>Y</i>	: Divider

Returns

Result of operation

8.102.2.42 `double bratl::CTools::Multiply (double X, double Y) [static]`

Multiply two numbers. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
in	<i>Y</i>	: Number involved

Returns

Result of operation

8.102.2.43 `double bratl::CTools::NormalizeLongitude (double Floor, double Longitude) [static]`

Find a number satisfying the condition $Floor \leq Longitude < Floor+360$. Takes default values into account

Parameters

in	<i>Floor</i>	: Base longitude
in	<i>Longitude</i>	: Longitude to normalize

Returns

Result of operation

8.102.2.44 `double bratl::CTools::Or (double X, double Y) [static]`

Do a logical or on two numbers. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
in	<i>Y</i>	: Number involved

Returns

Result of operation

8.102.2.45 `double bratl::CTools::Plus (double X, double Y) [static]`

Add two numbers. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
in	<i>Y</i>	: Number involved

Returns

Result of operation

8.102.2.46 `double brathl::CTools::Pow (double X, double Y) [static]`

Find the power of a number by another. Takes default values into account

Parameters

in	X	: Number involved
in	Y	: Power. Can be a integral or decimal

Returns

Result of operation

References `IsInf()`.

8.102.2.47 `double bratl::CTools::Rad2Deg (double X) [static]`

Convert radians to degrees. Takes default values into account

Parameters

in	X	: Number involved
----	---	-------------------

Returns

Result of operation

8.102.2.48 `char * bratl::CTools::RemoveAllSpaces (char * str) [static]`

Remove all the blank characters in a `std::string`. Blank characters are identified by the function `isspace (3C)`.

Parameters

<i>str</i>	[in/out] : <code>std::string</code> to be modified
------------	--

Returns

a pointer to the `std::string`

Referenced by `StringRemoveAllSpaces()`.

8.102.2.49 `std::string bratl::CTools::RemoveCharSurroundingNumber (const std::string & str, const char c1 = ' (' , const char c2 = ') ') [static]`

Removes characters `c1` and `c2`, if these characters surround an number (integer or decimal). For example↵
: `RemoveCharSurroundingNumber("ABCD (125)", '(', ')')` will return "ABCD 125" `RemoveCharSurrounding↵
Number("ABCD (+125.63)", '(', ')')` will return "ABCD +125.63" `RemoveCharSurroundingNumber("ABCD (-45) (X↵
YZ*2)", '(', ')')` will return "ABCD -45 (XYZ*2)" `RemoveCharSurroundingNumber("(ABCD ((-45)))", '(', ')')` will return
"(ABCD (-45))"

Parameters

in	<i>str</i>	: The <code>std::string</code> to modify
in	<i>c1</i>	: the first surrounding char
in	<i>c2</i>	: the last surrounding char

Returns

the newly modified `std::string`.

8.102.2.50 `void bratl::CTools::SetInternalDataDir (const std::string & DataDir) [static]`

Explicitly set the Data Directory.

Parameters

<i>in</i>	<i>DataDir</i>	: Full path to data directory.
-----------	----------------	--------------------------------

8.102.2.51 double brathl::CTools::Sign (double *X*) [static]

Find the sign of a number (1 if positive or null, -1 if negative). Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

8.102.2.52 double brathl::CTools::Sin (double *X*) [static]

Do the sine of a number expressed in radians. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

8.102.2.53 double brathl::CTools::SinD (double *X*) [static]

Do the sine of a number expressed in degrees. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

8.102.2.54 std::string brathl::CTools::SlashesDecode (const std::string & *str*, const std::string & *exclude* = " ", bool *decodeliterals* = true) [static]

Takes a std::string with escaped charters including decimal and hexadecimal escapes and decodes them to the literal charter. This function supports only standard C/C++ escaped literals.

Parameters

<i>in</i>	<i>str</i>	: The std::string to decode.
<i>in</i>	<i>exclude</i>	: A list of charters to exclude from decoding.
<i>in</i>	<i>decodeliterals</i>	: Set if non standard escaped literals are to be decoded.

Returns

the newly encoded std::string.

References Format().

8.102.2.55 `std::string bratl::CTools::SlashesEncode (const std::string & str, const std::string & exclude = " ", const std::string & literals = " ", bool hexadecimal = true) [static]`

This encodes characters that are not printable or can be encode with one of the C/C++ standard escape sequences. The 'exclude' list is a list of chars to exclude from the encoding process. Since the '\0' is used to determine the end of the std::string and will not be encoded.

Parameters

in	<i>str</i>	: The std::string to encode.
in	<i>exclude</i>	: A list of charters to exclude from encoding.
in	<i>literals</i>	:A list of printable characters to be included in the encoding.
	<i>hexadecimal</i>	If true, non-standard, non-printable charecters will be encoded in hexadecimal. If false they will be encoded in octal format.

Returns

the newly encoded std::string.

References Format().

8.102.2.56 `int32_t brathl::CTools::snprintf(char * str, size_t size, const char * format, ...) [static]`

Write formatted data to a std::string. WARNING : this method use vsnprintf if vsnprintf is defined, otherwise vsprintf is used and 'size' parameter is ignored

Parameters

out	<i>str</i>	: storage location for output.
in	<i>size</i>	: maximum number of characters to store
in	<i>format</i>	: format-control std::string
in	<i>...</i>	: optional arguments

Returns

return value of the vsnprintf or vsprintf - see documentation of these functions

8.102.2.57 `double brathl::CTools::Sqr (double X) [static]`

Find the square value of a number. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
----	----------	-------------------

Returns

Result of operation

References IsInf().

8.102.2.58 `double brathl::CTools::Sqrt (double X) [static]`

Find the square root value of a number. Takes default values into account

Parameters

in	<i>X</i>	: Number involved
----	----------	-------------------

Returns

Result of operation

8.102.2.59 `int32_t brathl::CTools::StrCaseCmp (const char * str1, const char * str2) [static]`

Compare the two strings str1 and str2, while being unaware of the differences between upper-case and lower-case. This method is thus identical to the function strcasecmp (3C) with the following difference : str1, str2 can be NULL, in this case, the std::string concerned is regarded as a null std::string.

Parameters

in	<i>str1</i>	: std::string 1
in	<i>str2</i>	: std::string 2

Returns

: negative, null (= 0) or positive value if the str1 is respectively lower, equal or higher than str2.

Referenced by brathl::CParameter::GetValue().

8.102.2.60 `std::string brathl::CTools::StringRemoveAllSpaces (const std::string & str) [static]`

Remove all the blank characters in a std::string. Blank characters are identified by the function isspace (3C).

Parameters

in	<i>str</i>	: std::string to be modified
----	------------	------------------------------

Returns

the modified std::string

References RemoveAllSpaces().

8.102.2.61 `std::string brathl::CTools::StringReplace (const std::string & str, char c, char replaceBy) [static]`

Replace all tokens of char c by char replaceBy in a std::string.

Parameters

in	<i>str</i>	: std::string to be modified
in	<i>c</i>	: char to replace
in	<i>replaceBy</i>	: char replaced

Returns

the modified std::string

8.102.2.62 `std::string brathl::CTools::StringReplace (const std::string & str, const std::string & c, const std::string & replaceBy, bool compareNoCase = false) [static]`

Replace all tokens of std::string c by std::string replaceBy in a std::string.

Parameters

in	<i>str</i>	: std::string to be modified
in	<i>c</i>	: std::string to replace
in	<i>replaceBy</i>	: std::string replaced

Returns

the modified std::string

8.102.2.63 `std::string brathl::CTools::StringToLower (const std::string & str) [static]`

Set a std::string object in lowercase

Parameters

<i>str</i>	[in/out] : std::string to be modified
------------	---------------------------------------

Returns

a new std::string object in lowercase

References ToLower().

Referenced by brathl::CProductEnvisat::IsHighResolutionField().

8.102.2.64 `std::string brathl::CTools::StringToUpper (const std::string & str) [static]`

Set a std::string object in uppercase

Parameters

<i>in</i>	<i>str</i>	: character
-----------	------------	-------------

Returns

a new std::string object in uppercase

References ToUpper().

8.102.2.65 `std::string brathl::CTools::StringTrim (const std::string & str) [static]`

Remove all the blank characters at the beginning and the end of a std::string. Blank characters are identified by the function isspace (3C).

Parameters

<i>str</i>	[in/out] : std::string to be modified
------------	---------------------------------------

Returns

a trimmed std::string

Referenced by StrToDouble(), Trim(), UnconvertLat(), and UnconvertLon().

8.102.2.66 `double brathl::CTools::StrToDouble (const std::string & value) [static]`

Convert an std::string to double

Parameters

<i>in</i>	<i>value</i>	: std::string to be converted
-----------	--------------	-------------------------------

Returns

coconverted value or CTool::m_defaultValueDOUBLE if no possible conversion.

References m_defaultValueDOUBLE, and StringTrim().

Referenced by UnconvertLat(), and UnconvertLon().

8.102.2.67 `int32_t brathl::CTools::StrToInt32 (const std::string & s) [static]`

Convert an std::string to int

Parameters

<i>in</i>	<i>value</i>	: std::string to be converted
-----------	--------------	-------------------------------

Returns

coconverted value or CTool::m_defaultValueINT if no possible conversion.

Referenced by bratl::CCriteriaCycle::Set(), bratl::CCriteriaPassInt::Set(), bratl::CCriteriaCycle::SetFrom(), bratl::CCriteriaPassInt::SetFrom(), bratl::CCriteriaCycle::SetTo(), and bratl::CCriteriaPassInt::SetTo().

8.102.2.68 `int64_t bratl::CTools::StrToInt64 (const std::string & s) [static]`

Convert an std::string to int64

Parameters

<i>in</i>	<i>value</i>	: std::string to be converted
-----------	--------------	-------------------------------

Returns

coconverted value or CTool::m_defaultValueINT if no possible conversion.

8.102.2.69 `uint64_t bratl::CTools::StrToUInt64 (const std::string & s) [static]`

Convert an std::string to uint64

Parameters

<i>in</i>	<i>value</i>	: std::string to be converted
-----------	--------------	-------------------------------

Returns

coconverted value or CTool::m_defaultValueINT if no possible conversion.

8.102.2.70 `double bratl::CTools::Tan (double X) [static]`

Do the tangent of a number expressed in radians. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

References IsInf(), and IsNan().

Referenced by TanD().

8.102.2.71 `double bratl::CTools::TanD (double X) [static]`

Do the tangent of a number expressed in degrees. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Result of operation

References Deg2Rad(), and Tan().

8.102.2.72 `char * brathl::CTools::ToLower (char * str)` `[static]`

Set a std::string in lowercase

Parameters

<i>str</i>	[in/out] : std::string to be modified
------------	---------------------------------------

Returns

a pointer to the std::string

Referenced by StringToLower().

8.102.2.73 `char brathl::CTools::ToLower (const char chr) [static]`

Set a std::string in lowercase

Parameters

<i>in</i>	<i>chr</i>	: character
-----------	------------	-------------

Returns

the lowercase character

8.102.2.74 `char * brathl::CTools::ToUpper (char * str) [static]`

Set a std::string in uppercase

Parameters

<i>str</i>	[in/out] : std::string to be modified
------------	---------------------------------------

Returns

a pointer to the std::string

Referenced by StringToUpper().

8.102.2.75 `char brathl::CTools::ToUpper (const char chr) [static]`

Set a character in uppercase

Parameters

<i>in</i>	<i>chr</i>	: character
-----------	------------	-------------

Returns

the uppercase character

8.102.2.76 `std::string brathl::CTools::TrailingZeroesTrim (const std::string & Text, bool dotTrim = true) [static]`

Removes trailing zeroes from a number: 2.30000 is transformed into 2.3.

Parameters

<i>in</i>	<i>Text</i>	: String
<i>in</i>	<i>dotTrim</i>	: if true, remove dot at the end : 2.000 -> 2, if false, leave dot : 2.000 -> 2.

Returns

Returns modified std::string

8.102.2.77 `char * brathl::CTools::Trim (char * str) [static]`

Remove all the blank characters at the beginning and the end of a `std::string`. Blank characters are identified by the function `isspace (3C)`.

Parameters

<i>str</i>	[in/out] : std::string to be modified
------------	---------------------------------------

Returns

a pointer to the std::string

References StringTrim().

Referenced by brathl::CFile::ReadLineData().

8.102.2.78 double brathl::CTools::UnaryMinus (double *X*) [static]

Negates a number. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Negated number

8.102.2.79 double brathl::CTools::UnaryNot (double *X*) [static]

Negates a logical value (0 is false, other (except default value) is true. Takes default values into account

Parameters

<i>in</i>	<i>X</i>	: Number involved
-----------	----------	-------------------

Returns

Negated value

8.102.2.80 double brathl::CTools::UnconvertLat (const std::string & *value*) [static]

Converts and normalize a latitude std::string representation (eg 60 N, 75.56 W, 60, -75.56) Normalize +/-90.

Parameters

<i>value</i>	latitude std::string representation
--------------	-------------------------------------

References m_defaultValueDOUBLE, StringTrim(), and StrToDouble().

8.102.2.81 double brathl::CTools::UnconvertLon (const std::string & *value*, bool *normalize* = true) [static]

Converts and eventually normalize a longitude std::string representation (eg 60 E, 120.23 W, 60, -120.23) Normalize +/-180.

Parameters

<i>normalize</i>	set to true to normalize longitude value
<i>value</i>	longitude std::string representation

Returns

converted longitude.

References m_defaultValueDOUBLE, StringTrim(), and StrToDouble().

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

- Tools.h
- Tools.cpp

8.103 brathl::CTreeField Class Reference

```
#include <TreeField.h>
```

Inherits brathl::ObjectTree.

Public Member Functions

- virtual ObjectTreeIterator **AddChild** (ObjectTreeNode *parent, const std::string &nm, **CField** *x, bool go↵
Current=false)
- virtual ObjectTreeIterator **AddChild** (ObjectTreeIterator &parent, const std::string &nm, **CField** *x, bool
goCurrent=false)
- virtual ObjectTreeIterator **AddChild** (const std::string &nm, **CField** *x, bool goCurrent=false)
- **CTreeField** ()
Empty CTreeField (p. 309) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr)
Dump function.
- void **DumpDictionary** (std::ostream &fOut=std::cout)
- void **DumpDictionary** (const std::string &outputFileName)
- **CField** * **FindParent** (**CField** *field)
- **CField** * **GetCurrentData** (bool withExcept=true)
- **CField** * **GetParentData** (bool withExcept=true)
- **CField** * **GetRootData** ()
- void **ResetHiddenFlag** ()
- virtual ~**CTreeField** ()
Destructor.

Static Public Member Functions

- static **CField** * **GetDataAsFieldObject** (ObjectTreeNode *node, bool withExcept=true)
- static **CFieldRecord** * **GetDataAsFieldRecordObject** (ObjectTreeNode *node, bool withExcept=true)

Static Public Attributes

- static const std::string **m_keyDelimiter** = "."

8.103.1 Detailed Description

Tree fields management class.

Version

1.0

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

- TreeField.h
- TreeField.cpp

8.104 brathl::CUIntMap Class Reference

```
#include <List.h>
```

Inherits mapuint.

Public Member Functions

- **CUIntMap** ()
CUIntMap (p. 309) ctor.
- virtual void **Dump** (std::ostream &fOut=std::cerr) const
Dump function.
- virtual bool **Erase** (CUIntMap::iterator it)
- virtual bool **Erase** (const std::string &key)
- virtual uint32_t **Exists** (const std::string &key) const
- virtual void **GetKeys** (CStringArray &keys, bool bRemoveAll=true)
- virtual uint32_t **Insert** (const std::string &key, uint32_t value, bool withExcept=true)
- virtual void **Insert** (const **CUIntMap** &m, bool bRemoveAll=true, bool withExcept=true)
- virtual void **Insert** (const CStringArray &keys, uint32_t initValue, bool bRemoveAll=true, bool withExcept=true)
- virtual void **Insert** (const CStringArray &keys, const CUIntArray &values, bool bRemoveAll=true, bool withExcept=true)
- virtual void **Insert** (const CStringArray &keys, bool bRemoveAll=true, bool withExcept=true)
- virtual uint32_t **operator[]** (const std::string &key)
- virtual void **RemoveAll** ()
- virtual ~**CUIntMap** ()
CUIntMap (p. 309) dtor.

8.104.1 Detailed Description

a set of unsigned integer value management classes.

Version

1.0

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

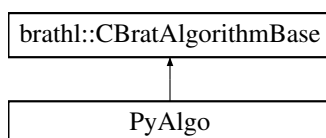
- List.h
- List.cpp

8.105 PyAlgo Class Reference

Definition of the object to hold each Python Algorithm and respective variables/methods.

```
#include <PythonEngine.hpp>
```

Inheritance diagram for PyAlgo:



Public Member Functions

- void **CreateAlgorithmParamVector** (CVectorBratAlgorithmParam &brat_args, char **args, size_t argcount)
- virtual std::string **GetDescription** () const override
- virtual std::string **GetInputParamDesc** (uint32_t indexParam) const override
- virtual
CBratAlgorithmParam::bratAlgoParamTypeVal **GetInputParamFormat** (uint32_t indexParam) const override

- virtual std::string **GetInputParamUnit** (uint32_t indexParam) const override
- virtual std::string **GetName** () const override
- virtual uint32_t **GetNumInputParam** () const override
- virtual std::string **GetOutputUnit** () const override
- virtual std::string **GetParamName** (uint32_t indexParam) const override
- **PyAlgo** (const std::string file_path, const std::string &class_name)
*User defined constructor for **PyAlgo** (p. 310).*
- virtual double **Run** (CVectorBratAlgorithmParam &args) override
- virtual ~**PyAlgo** ()
*Default destructor for **PyAlgo** (p. 310).*

Static Protected Member Functions

- static PyObject * **createPyArguments** (CVectorBratAlgorithmParam &args)
Method to create a list of Python arguments.
- template<typename T >
static T & **processCall** (PyObject *py_result, T &result)
Method to process the result of a method call.

Additional Inherited Members

8.105.1 Detailed Description

Definition of the object to hold each Python Algorithm and respective variables/methods.

8.105.2 Constructor & Destructor Documentation

8.105.2.1 PyAlgo::PyAlgo (const std::string file_path, const std::string & class_name) [inline]

User defined constructor for **PyAlgo** (p. 310).

Parameters

in	file_path	The path of the algorithm python script/module.
in	class_name	Name of the algorithm class.

8.105.3 Member Function Documentation

8.105.3.1 static PyObject* PyAlgo::createPyArguments (CVectorBratAlgorithmParam & args) [inline],[static],[protected]

Method to create a list of Python arguments.

Returns

pArgs Python List with python objects/arguments.

Referenced by Run().

8.105.3.2 virtual std::string PyAlgo::GetDescription () const [inline],[override],[virtual]

Gets the description of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 130).

References processCall().

8.105.3.3 `virtual std::string PyAlgo::GetInputParamDesc (uint32_t indexParam) const` `[inline],[override],`
`[virtual]`

Gets the description of an input parameter.

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References processCall().

8.105.3.4 `virtual CBratAlgorithmParam::bratAlgoParamTypeVal PyAlgo::GetInputParamFormat (uint32_t indexParam) const` `[inline], [override], [virtual]`

Gets the format of an input parameter : CBratAlgorithmParam::T_DOUBLE for double CBratAlgorithmParam::T_↵
 FLOAT for float CBratAlgorithmParam::T_INT for integer CBratAlgorithmParam::T_LONG for long integer CBrat↵
 AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References processCall().

8.105.3.5 `virtual std::string PyAlgo::GetInputParamUnit (uint32_t indexParam) const` `[inline], [override], [virtual]`

Gets the unit of an input parameter :

Parameters

<i>indexParam</i>	[in] : parameter index. First parameter index is 0, last one is 'number of parameters - 1'.
-------------------	---

Implements **brathl::CBratAlgorithmBase** (p. 130).

References processCall().

8.105.3.6 `virtual std::string PyAlgo::GetName () const` `[inline], [override], [virtual]`

Gets the name of the algorithm

Implements **brathl::CBratAlgorithmBase** (p. 131).

References processCall().

8.105.3.7 `virtual uint32_t PyAlgo::GetNumInputParam () const` `[inline], [override], [virtual]`

Gets the number of input parameters to pass to the 'Run' function

Implements **brathl::CBratAlgorithmBase** (p. 131).

References processCall().

8.105.3.8 `virtual std::string PyAlgo::GetOutputUnit () const` `[inline], [override], [virtual]`

Gets the unit of an output value returned by the 'Run' function.

Implements **brathl::CBratAlgorithmBase** (p. 131).

References processCall().

8.105.3.9 `template<typename T> static T& PyAlgo::processCall (PyObject * py_result, T & result)` `[inline], [static], [protected]`

Method to process the result of a method call.

Returns

result Result after conversion to proper data type.

Referenced by GetDescription(), GetInputParamDesc(), GetInputParamFormat(), GetInputParamUnit(), GetName(), GetNumInputParam(), GetOutputUnit(), and Run().

8.105.3.10 `virtual double PyAlgo::Run (CVectorBratAlgorithmParam & args) [inline],[override],
[virtual]`

Runs the algorithm

Parameters

<i>fmt</i>	[in] : a std::string that indicates the format of each value of input parameters (number, std::string) : d for integer l for long integer f for double s for std::string
<i>args</i>	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implements **brathl::CBratAlgorithmBase** (p. 131).

References createPyArguments(), and processCall().

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

- PythonEngine.hpp

8.106 PythonEngine Class Reference

Definition of the object to hold the Python Interpreter and respective methods.

```
#include <PythonEngine.hpp>
```

Public Member Functions

- bool **evaluate** (const std::string &expression) const
- PyObject * **getObject** (const std::string &name) const

Static Public Member Functions

- static PyObject * **convert** (PyObject *py_result, std::string &result)
- static PyObject * **convert** (PyObject *py_result, uint32_t &result)
- static PyObject * **convert** (PyObject *py_result, double &result)
- static **PythonEngine** * **CreateInstance** (wchar_t *pypath)
- static **PythonEngine** * **Instance** ()

Protected Member Functions

- **PythonEngine** (wchar_t *pypath)

Protected Attributes

- PyObject * **m_global_dict**

8.106.1 Detailed Description

Definition of the object to hold the Python Interpreter and respective methods.

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

- PythonEngine.hpp

9 File Documentation

9.1 brathl.h File Reference

```
#include <stdio.h>
#include <inttypes.h>
#include <sys/stat.h>
```

Classes

- struct **_structDateDSM**
- struct **_structDateJulian**
- struct **_structDateSecond**
- struct **_structDateYMDHMSM**

Macros

- #define **BRATHL_CYCLE_LEN** 60
- #define **BRATHL_MAX_ERRMSG_LEN** 255
- #define **BRATHL_PATH_MAX** PATH_MAX
- #define **BRATHL_REF_DATE_USER_LEN** 28
- #define **BRATHL_UNITFILE** "brathl_units.dat"
- #define **HAVE_INTTYPES_H** 1
- #define **HAVE_ISINF** 1
- #define **HAVE_ISNAN** 1
- #define **HAVE_REALPATH** 1
- #define **HAVE_STAT** 1
- #define **HAVE_STDINT_H** 1
- #define **HAVE_STRCASECMP** 1
- #define **HAVE_SYS_STAT_H** 1
- #define **HAVE_SYS_TYPES_H** 1
- #define **HAVE_UNISTD_H** 1
- #define **HAVE_VSNPRINTF** 1
- #define **LIBRATHL_API**
- #define **M_PI** 3.14159265358979323846
- #define **M_PI_2** 1.57079632679489661923 /* pi/2 */
- #define **M_PI_4** 0.78539816339744830962 /* pi/4 */
- #define **PATH_LIST_SEPARATOR** ":"
- #define **PATH_LIST_SEPARATOR_CHAR** ':'
- #define **PATH_SEPARATOR** "/"
- #define **PATH_SEPARATOR_CHAR** '/'

Typedefs

- typedef struct **_structDateDSM** brathl_DateDSM
- typedef struct **_structDateJulian** brathl_DateJulian
- typedef struct **_structDateSecond** brathl_DateSecond
- typedef struct **_structDateYMDHMSM** brathl_DateYMDHMSM
- typedef int **brathl_mission**

Enumerations

- enum **brathl_FileMode** { **ReadOnly**, **Write**, **Replace**, **New** }
- enum **brathl_refDate** {
REF19500101, **REF19580101**, **REF19850101**, **REF19900101**,
REF20000101, **REFUSER1**, **REFUSER2** }

Variables

- LIBRATHL_API char **brathl_refDateUser1** [BRATHL_REF_DATE_USER_LEN]
- LIBRATHL_API char **brathl_refDateUser2** [BRATHL_REF_DATE_USER_LEN]

9.1.1 Detailed Description

C/C++ general interface of BRATHL

9.1.2 Macro Definition Documentation

9.1.2.1 #define BRATHL_CYCLE_LEN 60

Maximum length of date reference string

9.1.2.2 #define BRATHL_MAX_ERRMSG_LEN 255

Maximum length of error message string

9.1.2.3 #define BRATHL_REF_DATE_USER_LEN 28

Maximum length of date reference string

Referenced by FTN_NAME().

9.1.3 Typedef Documentation

9.1.3.1 typedef struct **_structDateDSM** brathl_DateDSM

Day/seconds/microseconds date structureCreates a type name for **_structDateDSM** (p. 101)

9.1.3.2 typedef struct **_structDateJulian** brathl_DateJulian

Decimal julian date structureCreates a type name for **_structDateJulian** (p. 102)

9.1.3.3 typedef struct **_structDateSecond** brathl_DateSecond

Decimal seconds date structureCreates a type name for **_structDateSecond** (p. 102)

9.1.3.4 typedef struct **_structDateYMDHMSM** brathl_DateYMDHMSM

YYYY-MM-DD HH:MN:SS:MS date structureCreates a type name for **_structDateYMDHMSM** (p. 103)

9.1.3.5 typedef int brathl_mission

Satellite (mission) ID -> On Brat V.4, mission ID is defined on txt file CMission::m_refFileName

9.1.4 Enumeration Type Documentation

9.1.4.1 enum brathl_FileMode

Enumerator

Write file exists, open read-only

Replace file exists, open for writing

New create new file, even if it already exists create new file, fail if it already exists

9.1.4.2 enum brathl_refDate

date reference enumeration Used to give a date a a start reference User can defined its own reference by using REFUSER1 and/or REFUSER2

Enumerator

REF19500101 reference to the 1950-01-01 00:00:00:00

REF19580101 reference to the 1958-01-01 00:00:00:00

REF19850101 reference to the 1985-01-01 00:00:00:00

REF19900101 reference to the 1990-01-01 00:00:00:00

REF20000101 reference to the 2000-01-01 00:00:00:00

REFUSER1 reference to a user-defined date **brathl_refDateUser1** (p. 317)

REFUSER2 reference to a second user-defined date **brathl_refDateUser2** (p. 317)

9.1.5 Variable Documentation

9.1.5.1 LIBRATHL_API char brathl_refDateUser1[BRATHL_REF_DATE_USER_LEN]

Global variable to define REFUSER1 date (see **brathl_refDate** (p. 317))

Referenced by brathl::CDate::ConstructDate(), and FTN_NAME().

9.1.5.2 LIBRATHL_API char brathl_refDateUser2[BRATHL_REF_DATE_USER_LEN]

Global variable to define REFUSER2 date (see **brathl_refDate** (p. 317))

Referenced by brathl::CDate::ConstructDate(), and FTN_NAME().

9.2 brathl_fortran.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "brathlc.h"
```

Macros

- #define _(A, B) AB

- `#define _2(A, B) AB`
- `#define _3(A, B, C) ABC`
- `#define _cleft 1`
- `#define _cleft_cfright 0`
- `#define _cfright`
- `#define ANSI_C_preprocessor _cleft_cfright`
- `#define FTN_NAME(Low, Up) ADD_UNDESCORE(Low)`
- `#define INTEGER4 int32_t`
- `#define MIN(A, B) ((A) < (B) ? (A) : (B))`
- `#define REAL8 double`
- `#define SECOND_UNDESCORE(X) X`

Functions

- void **FTN_NAME** (brathlf_setrefuser1, BRATHLF_SETREFUSER1)
- void **FTN_NAME** (brathlf_setrefuser2, BRATHLF_SETREFUSER2)
- INTEGER4 **FTN_NAME** (brathlf_geterrno, BRATHLF_GETERRNO)
- void **FTN_NAME** (brathlf_errno2string, BRATHLF_ERRNO2STRING)
- INTEGER4 **FTN_NAME** (brathlf_seconds2dsm, BRATHLF_SECONDS2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2seconds, BRATHLF_DSM2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_julian2dsm, BRATHLF_JULIAN2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2julian, BRATHLF_DSM2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2dsm, BRATHLF_YMDHMSM2DSM)
- INTEGER4 **FTN_NAME** (brathlf_dsm2ymdhmsm, BRATHLF_DSM2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_seconds2julian, BRATHLF_SECONDS2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_julian2seconds, BRATHLF_JULIAN2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_seconds2ymdhmsm, BRATHLF_SECONDS2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2seconds, BRATHLF_YMDHMSM2SECONDS)
- INTEGER4 **FTN_NAME** (brathlf_julian2ymdhmsm, BRATHLF_JULIAN2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2julian, BRATHLF_YMDHMSM2JULIAN)
- INTEGER4 **FTN_NAME** (brathlf_nowymdhmsm, BRATHLF_NOWYMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_dayofyear, BRATHLF_DAYOFYEAR)
- INTEGER4 **FTN_NAME** (brathlf_diffymdhmsm, BRATHLF_DIFFYMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_diffdsm, BRATHLF_DIFFDSM)
- INTEGER4 **FTN_NAME** (brathlf_diffjulian, BRATHLF_DIFFJULIAN)
- INTEGER4 **FTN_NAME** (brathlf_cycle2ymdhmsm, BRATHLF_CYCLE2YMDHMSM)
- INTEGER4 **FTN_NAME** (brathlf_ymdhmsm2cycle, BRATHLF_YMDHMSM2CYCLE)
- INTEGER4 **FTN_NAME** (brathlf_readdata, BRATHLF_READDATA)
- static char * **GetFtnString** (const char *FtnString, int32_t FtnLength)
- static char ** **GetFtnStringArray** (const char *FtnString, int32_t FtnLength, int32_t ArraySize)
- static int32_t **GetFtnStringLen** (const char *FtnString, int32_t FtnLength)
- static void **PutFtnString** (char *FtnString, int32_t FtnLength, const char *CString)
- static void **PutFtnStringArray** (char *FtnString, int32_t FtnLength, int32_t ArraySize, const char **CStrings)

9.2.1 Detailed Description

Fortran general interface of BRATHL no .h file since it is only called from fortran

9.3 brathlc.h File Reference

```
#include "brathl.h"
#include "common/ccore-types.h"
#include "common/tools/brathl_error.h"
```


Functions

- LIBRATHL_API int32_t **brathl_Cycle2YMDHMSM** (**brathl_mission** mission, int32_t cycle, int32_t pass, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_DayOfYear** (**brathl_DateYMDHMSM** *dateYMDHMSM, uint32_t *dayOfYear)
- LIBRATHL_API int32_t **brathl_DiffDSM** (**brathl_DateDSM** *dateDSM1, **brathl_DateDSM** *dateDSM2, double *diff)
- LIBRATHL_API int32_t **brathl_DiffJulian** (**brathl_DateJulian** *dateJulian1, **brathl_DateJulian** *dateJulian2, double *diff)
- LIBRATHL_API int32_t **brathl_DiffYMDHMSM** (**brathl_DateYMDHMSM** *dateYMDHMSM1, **brathl_DateYMDHMSM** *dateYMDHMSM2, double *diff)
- LIBRATHL_API int32_t **brathl_DSM2Julian** (**brathl_DateDSM** *dateDSM, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_DSM2Seconds** (**brathl_DateDSM** *dateDSM, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)
- LIBRATHL_API int32_t **brathl_DSM2YMDHMSM** (**brathl_DateDSM** *dateDSM, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API const char * **brathl_Errno2String** (const int32_t err)
- LIBRATHL_API int32_t **brathl_Julian2DSM** (**brathl_DateJulian** *dateJulian, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_Julian2Seconds** (**brathl_DateJulian** *dateJulian, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)
- LIBRATHL_API int32_t **brathl_Julian2YMDHMSM** (**brathl_DateJulian** *dateJulian, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API void **brathl_LoadAliasesDictionary** ()
- LIBRATHL_API int32_t **brathl_NowYMDHMSM** (**brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_ReadData** (int32_t nbFiles, char **fileNames, const char *recordName, const char *selection, int32_t nbData, char **dataExpressions, char **units, double **results, int32_t sizes[], size_t *actualSize, int ignoreOutOfRange, int statistics, double defaultValue)
- LIBRATHL_API void **brathl_RegisterAlgorithms** ()
- LIBRATHL_API int32_t **brathl_Seconds2DSM** (**brathl_DateSecond** *dateSeconds, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_Seconds2Julian** (**brathl_DateSecond** *dateSeconds, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_Seconds2YMDHMSM** (**brathl_DateSecond** *dateSeconds, **brathl_DateYMDHMSM** *dateYMDHMSM)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Cycle** (**brathl_mission** mission, **brathl_DateYMDHMSM** *dateYMDHMSM, int32_t *cycle, int32_t *pass)
- LIBRATHL_API int32_t **brathl_YMDHMSM2DSM** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateDSM** *dateDSM)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Julian** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateJulian** *dateJulian)
- LIBRATHL_API int32_t **brathl_YMDHMSM2Seconds** (**brathl_DateYMDHMSM** *dateYMDHMSM, **brathl_refDate** refDate, **brathl_DateSecond** *dateSeconds)

Variables

- LIBRATHL_API int **brathl_errno**

9.3.1 Detailed Description

C general interface of BRATHL

9.3.2 Function Documentation

9.3.2.1 LIBRATHL_API const char* brathl_Errno2String (const int32_t *err*)

Retrieve a string with the error description

With a few exceptions almost all BRATHL functions return an integer that indicate whether the function was able to perform its operations successfully. The return value will be 0 on success and < 0 otherwise. The result is also save in the global variable **brathl_errno** (p. 320) In case you get a negative value.

Parameters

<i>in</i>	<i>err</i>	: error code
-----------	------------	--------------

Returns

string error description

Referenced by FTN_NAME().

9.3.3 Variable Documentation

9.3.3.1 LIBRATHL_API int brathl_errno

Global variable to save error code

Referenced by brathl_Cycle2YMDHMSM(), brathl_DayOfYear(), brathl_DiffDSM(), brathl_DiffJulian(), brathl_DiffYMDHMSM(), brathl_DSM2Julian(), brathl_DSM2Seconds(), brathl_DSM2YMDHMSM(), brathl_Julian2DSM(), brathl_Julian2Seconds(), brathl_Julian2YMDHMSM(), brathl_NowYMDHMSM(), brathl_ReadData(), brathl_Seconds2DSM(), brathl_Seconds2Julian(), brathl_Seconds2YMDHMSM(), brathl_YMDHMSM2Cycle(), brathl_YMDHMSM2DSM(), brathl_YMDHMSM2Julian(), brathl_YMDHMSM2Seconds(), and FTN_NAME().

9.4 MapParameter.h File Reference

```
#include <string>
#include "Parameter.h"
```

Classes

- class **brathl::CMapParameter**

Namespaces

- **brathl**

Typedefs

- typedef std::map< std::string, **CParameter** * > **brathl::map_parameter**

9.4.1 Detailed Description

Class definition file