BRAT

4.2.0

Generated by Doxygen 1.8.8

Thu Sep 28 2017 16:29:48

Contents

1 Module Index

1.1 Modules

Here is a list of all modules:

2 Namespace Index

2.1 Namespace List

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

3 Hierarchical Index

3.1 Class Hierarchy

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

4 Class Index

4.1 Class List

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

5 File Index

5.1 File List

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

6 Module Documentation

6.1 Algorithms classes

Classes

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

Macros

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

Typedefs

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

Functions

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

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

Variables

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

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 brathl::CBratAlgorithmGeosVelGrid::CBratAlgorithmGeosVelGrid ( )

Default contructor

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

Copy contructor

6.1.2.3 brathl::CBratAlgorithmGeosVelGridU::CBratAlgorithmGeosVelGridU ( )

Default contructor

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

Copy contructor

6.1.2.5 brathl::CBratAlgorithmGeosVelGridV::CBratAlgorithmGeosVelGridV ( )

Default contructor

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

Copy contructor

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

indexParam	[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

indexParam	[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().

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

Gets the unit of an input parameter :

Parameters

indexParam	[in] : parameter index.

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

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

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

Gets the name of the algorithm

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

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

Gets the name of the algorithm

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

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

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

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

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

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

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

Parameters

indexParam	[in] : parameter index.

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

6.1.2.19 CBratAlgorithmGeosVelGrid & brathl::CBratAlgorithmGeosVelGrid::operator= (const CBratAlgorithmGeosVelGrid & *copy*)

Overloads operator '='

6.1.2.20 double brathl::CBratAlgorithmGeosVelGrid::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

6.1.2.21 brathl::CBratAlgorithmGeosVelGrid::~CBratAlgorithmGeosVelGrid ( ) [virtual]

Destructor

6.1.2.22 brathl::CBratAlgorithmGeosVelGridU::~CBratAlgorithmGeosVelGridU ( ) [virtual]

Destructor

6.1.2.23 brathl::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::CInternalFilesZFXY**
- 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 > **brathl::maparraystring**
- typedef std::map< std::string, CObjectTreeNode * > **brathl::mapTreeNode**

Functions

- void **brathl::CArrayDoublePtrArray::AdjustValidMinMax** (double value)
- void **brathl::CArrayDoubleArray::AdjustValidMinMax** (double value)
- double * **brathl::CMatrix::At** (size_t i, size_t j)
- CExternalFiles * **brathl::BuildExistingExternalFileKind** (const std::string &path)
- **CInternalFiles** * **brathl::BuildExistingInternalFileKind** (const std::string &name, const CStringArray *fieldNames)
- **brathl::CArrayDoubleArray::CArrayDoubleArray** ()

  Empty CDoubleArray ctor.
- **brathl::CArrayDoubleArray::CArrayDoubleArray** (const **CArrayDoubleArray** &a)
- **brathl::CArrayDoublePtrArray::CArrayDoublePtrArray** (bool bDelete=true)

  Empty CDoubleArray ctor.
- **brathl::CArrayDoublePtrArray::CArrayDoublePtrArray** (const **CArrayDoublePtrArray** &a)
- **brathl::CArrayStringMap::CArrayStringMap** ()

  CStringMap *(*p. *279) ctor.*
- **brathl::CArrayStringMap::CArrayStringMap** (const CArrayStringMap &a)
- **brathl::CDoubleArrayOb::CDoubleArrayOb** (const CDoubleArrayOb &vect)
- **brathl::CDoubleMap::CDoubleMap** ()

  CDoubleMap *(*p. *192) ctor.*
- **brathl::CDoublePtrArray::CDoublePtrArray** (bool bDelete=true)

  Empty ***CDoublePtrArray*** *(*p. *193) ctor.*
- **brathl::CDoublePtrDoubleMap::CDoublePtrDoubleMap** (bool bDelete=true)

  CDoublePtrDoubleMap *(*p. *194) ctor.*
- **brathl::CDoublePtrDoubleMap::CDoublePtrDoubleMap** (const CUIntArray &matrixDims, bool bDelete=true)
- **brathl::CIntList::CIntList** ()

  Empty ***CIntList*** *(*p. *233) ctor.*
- **brathl::CIntList::CIntList** (const **CIntList** &list)
- **brathl::CIntMap::CIntMap** ()

  CIntMap *(*p. *234) ctor.*
- virtual CBratObject * **brathl::CDoubleArrayOb::Clone** ()
- virtual CBratObject * **brathl::CObArrayOb::Clone** ()
- **brathl::CMatrix::CMatrix** (const CMatrix &m)
- **brathl::CMatrixDouble::CMatrixDouble** (size_t nrows, size_t ncols)
- **brathl::CMatrixDouble::CMatrixDouble** (const CMatrixDouble &m)
- **brathl::CMatrixDoublePtr::CMatrixDoublePtr** (size_t nrows, size_t ncols)
- **brathl::CMatrixDoublePtr::CMatrixDoublePtr** (const CMatrixDoublePtr &m)
- **brathl::CObArray::CObArray** (bool bDelete=true)

  Empty ***CObArray*** *(*p. *237) ctor.*
- **brathl::CObArray::CObArray** (const **CObArray** &vect)
- **brathl::CObArrayOb::CObArrayOb** (bool bDelete=true)
- **brathl::CObArrayOb::CObArrayOb** (const CObArrayOb &vect)
- **brathl::CObDoubleMap::CObDoubleMap** (bool bDelete=true)

  CObMap *(*p. *241) ctor.*
- **brathl::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 fonction.

- virtual void **brathl::CDoubleMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CObMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CObIntMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CObDoubleMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CDoublePtrDoubleMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CPtrMap::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CMatrix::Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual void **brathl::CMatrixDoublePtr::Dump** (std::ostream &fOut=std::cerr) const override

  Dump fonction.
- virtual void **brathl::CMatrixDouble::Dump** (std::ostream &fOut=std::cerr) const override

  Dump fonction.
- 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 **brathl::CDoubleMap::Exists** (const std::string &key) const
- virtual CBratObject * **brathl::CObMap::Exists** (const std::string &key) const
- virtual CBratObject * **brathl::CObIntMap::Exists** (int32_t key) const
- virtual CBratObject * **brathl::CObDoubleMap::Exists** (double key) const
- virtual DoublePtr * **brathl::CDoublePtrDoubleMap::Exists** (double key) const
- virtual void * **brathl::CPtrMap::Exists** (const std::string &key) const
- virtual bool **brathl::CStringList::ExistsNoCase** (const std::string &str) const
- virtual void **brathl::CStringList::ExtractKeys** (const std::string &str, const std::string &delim, bool b↩ RemoveAll=true)
- virtual void **brathl::CStringList::ExtractStrings** (const std::string &str, const char delim, bool bRemove↩ All=true)
- virtual void **brathl::CStringList::ExtractStrings** (const std::string &str, const std::string &delim, bool b↩ RemoveAll=true)
- virtual int32_t **brathl::CStringList::FindIndex** (const std::string &str, bool compareNoCase=false) const
- const CArrayDoublePtrArray & **brathl::CMatrixDoublePtr::GetData** ()
- const CArrayDoubleArray & **brathl::CMatrixDouble::GetData** ()
- bool **brathl::CObList::GetDelete** ()
- bool **brathl::CDoublePtrArray::GetDelete** ()
- bool **brathl::CArrayDoublePtrArray::GetDelete** ()
- bool **brathl::CObStack::GetDelete** ()
- bool **brathl::CObArray::GetDelete** ()
- bool **brathl::CObMap::GetDelete** ()
- bool **brathl::CObIntMap::GetDelete** ()
- bool **brathl::CObDoubleMap::GetDelete** ()
- bool **brathl::CDoublePtrDoubleMap::GetDelete** ()
- virtual void **brathl::CStringMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true) const
- virtual void **brathl::CUIntMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true)
- virtual void **brathl::CObMap::GetKeys** (CStringArray &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual void **brathl::CObMap::GetKeys** (**CStringList** &keys, bool bRemoveAll=true, bool bUnique=false)
- virtual void **brathl::CObIntMap::GetKeys** (CIntArray &keys, bool bRemoveAll=true)
- virtual void **brathl::CObDoubleMap::GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- virtual void **brathl::CDoublePtrDoubleMap::GetKeys** (CDoubleArray &keys, bool bRemoveAll=true)
- uint32_t **brathl::CDoublePtrDoubleMap::GetMatrixColDim** (uint32_t row)
- CStringArray * **brathl::CMatrixDoublePtr::GetMatrixDataDimIndexes** ()
- uint32_t **brathl::CDoublePtrArray::GetMatrixDim** (uint32_t row)
- size_t **brathl::CArrayDoublePtrArray::GetMatrixDim** (size_t row)
- size_t **brathl::CMatrixDoublePtr::GetMatrixDimData** (size_t row)
- CUIntArray * **brathl::CDoublePtrArray::GetMatrixDims** ()
- CUIntArray * **brathl::CArrayDoublePtrArray::GetMatrixDims** ()
- const CUIntArray * **brathl::CArrayDoublePtrArray::GetMatrixDims** () const
- CUIntArray * **brathl::CDoublePtrDoubleMap::GetMatrixDims** ()
- const CUIntArray * **brathl::CMatrixDoublePtr::GetMatrixDimsData** () const
- size_t **brathl::CDoublePtrArray::GetMatrixNumberOfDims** ()
- size_t **brathl::CArrayDoublePtrArray::GetMatrixNumberOfDims** ()
- size_t **brathl::CMatrixDoublePtr::GetMatrixNumberOfDimsData** ()
- size_t **brathl::CDoublePtrDoubleMap::GetMatrixNumberOfRows** () const
- virtual size_t **brathl::CMatrix::GetMatrixNumberOfValuesData** () const
- virtual size_t **brathl::CMatrixDoublePtr::GetMatrixNumberOfValuesData** () const override
- virtual size_t **brathl::CMatrixDouble::GetMatrixNumberOfValuesData** () const override
- void **brathl::CArrayDoublePtrArray::GetMinMaxValues** (double &min, double &max, bool recalc=true)
- void **brathl::CArrayDoubleArray::GetMinMaxValues** (double &min, double &max, bool recalc=true)
- virtual void **brathl::CMatrix::GetMinMaxValues** (double &min, double &max)=0
- virtual void **brathl::CMatrixDoublePtr::GetMinMaxValues** (double &min, double &max) override
- virtual void **brathl::CMatrixDouble::GetMinMaxValues** (double &min, double &max) override
- std::string **brathl::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 initial↩ Value=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< B↩ RAT_OBJECT > &o)
- virtual std::string **brathl::CStringMap::Insert** (const std::string &key, const std::string &str, bool with↩ Except=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 bRemove↩ All=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 bRemove↩ All=true, bool withExcept=true)
- virtual void **brathl::CUIntMap::Insert** (const CStringArray &keys, bool bRemoveAll=true, bool with↩ Except=true)
- virtual double **brathl::CDoubleMap::Insert** (const std::string &key, double value, bool withExcept=true)

- virtual CBratObject * **brathl::CObMap::Insert** (const std::string &key, CBratObject *ob, bool with↩ Except=true)
- virtual void **brathl::CObMap::Insert** (const **CObMap** &obMap, bool withExcept=true)
- virtual CBratObject * **brathl::CObIntMap::Insert** (int32_t key, CBratObject *ob, bool withExcept=true)
- virtual void **brathl::CObIntMap::Insert** (const **CObIntMap** &obMap, bool withExcept=true)
- virtual CBratObject * **brathl::CObDoubleMap::Insert** (double key, CBratObject *ob, bool withExcept=true)
- virtual void **brathl::CObDoubleMap::Insert** (const **CObDoubleMap** &obMap, bool withExcept=true)
- virtual DoublePtr * **brathl::CDoublePtrDoubleMap::Insert** (double key, DoublePtr *ob, bool with↩ Except=true)
- virtual DoublePtr * **brathl::CDoublePtrDoubleMap::Insert** (double key, double initialValue=CTools::m_↩ defaultValueDOUBLE)
- virtual void * **brathl::CPtrMap::Insert** (const std::string &key, void *ptr, bool withExcept=true)
- virtual void **brathl::CPtrMap::Insert** (const **CPtrMap** &ptrMap, bool withExcept=true)
- virtual CDoublePtrArray::iterator **brathl::CDoublePtrArray::InsertAt** (CDoublePtrArray::iterator where, DoublePtr ob)
- virtual CObArray::iterator **brathl::CObArray::InsertAt** (CObArray::iterator where, CBratObject *ob)
- virtual void **brathl::CStringList::InsertUnique** (const std::string &str, bool bEnd=true)
- virtual void **brathl::CStringList::InsertUnique** (const **CStringList** &lst, bool bEnd=true)
- virtual void **brathl::CStringList::InsertUnique** (const CStringArray *vect, bool bEnd=true)
- virtual void **brathl::CStringList::InsertUnique** (const CStringArray &vect, bool bEnd=true)
- virtual void **brathl::CStringList::InsertUnique** (const std::vector< std::string > &vect, bool bEnd=true)
- virtual void **brathl::CStringList::InsertUnique** (const **stringlist** &lst, bool bEnd=true)
- virtual bool **brathl::CStringList::Intersect** (const **CStringList** &array, **CStringList** &intersect) const
- virtual bool **brathl::CMatrix::IsMatrixDataSet** ()
- bool **brathl::CMatrixDoublePtr::IsMatrixDataSet** () override
- virtual std::string **brathl::CStringMap::IsValue** (const std::string &value)
- DoublePtr **brathl::CDoublePtrArray::NewMatrix** (double initialValue=CTools::m_defaultValueDOUBLE)
- DoublePtr **brathl::CArrayDoublePtrArray::NewMatrix** (double initialValue=CTools::m_defaultValueDOUB↩ LE)
- DoublePtr * **brathl::CDoublePtrDoubleMap::NewMatrix** (double initialValue=CTools::m_defaultValueDO↩ UBLE)
- DoublePtr **brathl::CMatrixDoublePtr::NewMatrixData** (double initialValue=CTools::m_defaultValueDOUB↩ LE)
- virtual double * **brathl::CMatrix::operator()** (size_t i, size_t j)=0
- virtual double * **brathl::CMatrix::operator()** (size_t i, size_t j) const =0
- virtual double * **brathl::CMatrixDoublePtr::operator()** (size_t i, size_t j) override
- virtual double * **brathl::CMatrixDoublePtr::operator()** (size_t i, size_t j) const override
- virtual double * **brathl::CMatrixDouble::operator()** (size_t i, size_t j) override
- virtual double * **brathl::CMatrixDouble::operator()** (size_t i, size_t j) const override
- virtual const **CStringList** & **brathl::CStringList::operator=** (const **CStringList** &lst)
- virtual const **CStringList** & **brathl::CStringList::operator=** (const CStringArray &vect)
- virtual const **CStringList** & **brathl::CStringList::operator=** (const std::vector< std::string > &vect)
- virtual const **CStringList** & **brathl::CStringList::operator=** (const **stringlist** &lst)
- const **CIntList** & **brathl::CIntList::operator=** (const **CIntList** &lst)
- virtual const **CObList** & **brathl::CObList::operator=** (const **CObList** &lst)
- virtual const **CArrayDoublePtrArray** & **brathl::CArrayDoublePtrArray::operator=** (const **CArrayDoublePtrArray** &m)
- virtual const **CArrayDoubleArray** & **brathl::CArrayDoubleArray::operator=** (const **CArrayDoubleArray** &m)
- virtual const CArrayStringMap & **brathl::CArrayStringMap::operator=** (const CArrayStringMap &a)
- virtual const CDoubleArrayOb & **brathl::CDoubleArrayOb::operator=** (const CDoubleArrayOb &vect)
- virtual const **CObArray** & **brathl::CObArray::operator=** (const **CObArray** &lst)
- CObjectPointersArray < BRAT_OBJECT > & **brathl::CObjectPointersArray< BRAT_OBJECT >::operator=** (const CObject↩ PointersArray< 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 **brathl::CMatrix::ScaleDownData** (double scaleFactor, double addOffset, double default↩ Value=CTools::m_defaultValueDOUBLE)=0
- virtual void **brathl::CMatrixDoublePtr::ScaleDownData** (double scaleFactor, double addOffset, double defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **brathl::CMatrixDouble::ScaleDownData** (double scaleFactor, double addOffset, double defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **brathl::CMatrix::ScaleUpData** (double scaleFactor, double addOffset, double defaultValue=C↩ Tools::m_defaultValueDOUBLE)=0
- virtual void **brathl::CMatrixDoublePtr::ScaleUpData** (double scaleFactor, double addOffset, double defaultValue=CTools::m_defaultValueDOUBLE) override
- virtual void **brathl::CMatrixDouble::ScaleUpData** (double scaleFactor, double addOffset, double default↩ Value=CTools::m_defaultValueDOUBLE) override
- void **brathl::CArrayDoublePtrArray::Set** (const **CArrayDoublePtrArray** &m)
- void **brathl::CArrayDoubleArray::Set** (const **CArrayDoubleArray** &m)
- virtual void **brathl::CArrayStringMap::Set** (const CArrayStringMap &a)
- virtual void **brathl::CMatrix::Set** (const CMatrix &m)
- virtual void **brathl::CMatrix::Set** (size_t &row, size_t &col, double ∗x)=0
- void **brathl::CMatrixDoublePtr::Set** (size_t &row, size_t &col, double ∗x) override
- void **brathl::CMatrixDoublePtr::Set** (const CMatrixDoublePtr &m)
- void **brathl::CMatrixDouble::Set** (size_t &row, size_t &col, double ∗x) override
- void **brathl::CMatrixDouble::Set** (const CMatrixDouble &m)
- void **brathl::CObList::SetDelete** (bool value)
- void **brathl::CDoublePtrArray::SetDelete** (bool value)
- void **brathl::CArrayDoublePtrArray::SetDelete** (bool value)
- void **brathl::CObStack::SetDelete** (bool value)
- void **brathl::CObArray::SetDelete** (bool value)
- void **brathl::CObMap::SetDelete** (bool value)
- void **brathl::CObIntMap::SetDelete** (bool value)
- void **brathl::CObDoubleMap::SetDelete** (bool value)
- void **brathl::CDoublePtrDoubleMap::SetDelete** (bool value)
- void **brathl::CMatrixDoublePtr::SetMatrixDataDimIndexes** (const CStringArray &m)
- void **brathl::CDoublePtrArray::SetMatrixDims** (const CUIntArray &matrixDims)
- void **brathl::CArrayDoublePtrArray::SetMatrixDims** (const CUIntArray &matrixDims)
- void **brathl::CDoublePtrDoubleMap::SetMatrixDims** (const CUIntArray &matrixDims)
- void **brathl::CMatrixDoublePtr::SetMatrixDimsData** (const CUIntArray &matrixDims)
- void **brathl::CMatrixDoublePtr::SetMatrixDimsData** (size_t nbValues)
- void **brathl::CMatrix::SetName** (const std::string &value)
- void **brathl::CMatrix::SetXName** (const std::string &value)
- void **brathl::CMatrix::SetYName** (const std::string &value)
- virtual void **brathl::CObMap::ToArray** (**CObArray** &obArray)
- virtual CBratObject ∗ **brathl::CObStack::Top** ()
- virtual std::string **brathl::CStringList::ToString** (const std::string &delim=",", bool useBracket=true) const
- virtual **brathl::CArrayDoubleArray::∼CArrayDoubleArray** ()

  Destructor.
- virtual **brathl::CArrayDoublePtrArray::∼CArrayDoublePtrArray** ()

  Destructor.
- virtual **brathl::CArrayStringMap::∼CArrayStringMap** ()

  CStringMap *(*p. *279) dtor.*
- virtual **brathl::CDoubleMap::∼CDoubleMap** ()

  CDoubleMap *(*p. *192) dtor.*
- virtual **brathl::CDoublePtrArray::∼CDoublePtrArray** ()

  Destructor.
- virtual **brathl::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_STANDARD** = "Generic NetCdf Standard"
- const std::string **brathl::GENERIC_NETCDF_TYPE_VARIANT_1** = "Generic NetCdf Variant 1"
- 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::ZFXY_NETCDF_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

std::list	[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

vect	[in] : std::list to be copied

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

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

Parameters

lst	[in] : std::list to be copied

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

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

Parameters

std::list	[in] : std::list to be copied

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

Delete an element referenced by ob

Returns

true if no error, otherwise false

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

Delete an element referenced by it

Returns

true if no error, otherwise false

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

Delete an element referenced by it

Returns

true if no error, otherwise false

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

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

Delete an element referenced by index

Returns

true if no error, otherwise false

References brathl::CDoublePtrArray::Erase().

6.2.4.11 bool brathl::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::CArrayStringMap::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 brathl::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 brathl::CTools::m_defaultValueUINT32.

Referenced by brathl::CUIntMap::operator[]().

6.2.4.38 double brathl::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 brathl::CTools::m_defaultValueDOUBLE.

Referenced by brathl::CDoubleMap::operator[]().

6.2.4.39 CBratObject * brathl::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 * brathl::CObIntMap::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 * brathl::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 * brathl::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

keys	[out] : the keys of the std::map
bRemoveAll	[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

keys	[out] : the keys of the std::map
bRemoveAll	[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

keys	[out] : the keys of the std::map
bRemoveAll	[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

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

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

Gets keys of the std::map

Parameters

keys	[out] : the keys of the std::map
bRemoveAll	[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

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

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

Gets keys of the std::map

Parameters

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

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

Inserts a std::string

Parameters

key	: std::map key
str	: std::string value

Returns

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

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

Inserts a std::string

Parameters

key	: std::map key
str	: std::string value

Returns

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

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

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

Inserts a std::string std::map

Parameters

strmap	: std::map to insert
withExcept	: true for exception handling, flse otherwise

Returns

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

References brathl::CStringMap::Insert().

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

Inserts an integer

Parameters

key	: std::map key
value	: 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

std::map	[in]: std::map
bRemoveAll	[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

key	: std::map key
value	: 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

std::map	[in]: std::map
bRemoveAll	[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

keys	[in]: std::map keys to insert
initValue	[in]: value of the keys

bRemoveAll	[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

keys	[in]: keys to insert
values	[in]: values to insert
bRemoveAll	[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

key	: std::map key
value	: 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

key	: CBratObject name (std::map key)
value	: CBratObject value
withExcept	: 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

obMap	: **CObMap** (p. 241) to insert
withExcept	: true for exception handling, flse otherwise

References brathl::CObMap::Insert().

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

Inserts a CBratObject object

Parameters

key	: CBratObject name (std::map key)
value	: CBratObject value
withExcept	: 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

obMap	: **CObMap** (p. 241) to insert
withExcept	: 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

key	: CBratObject name (std::map key)
value	: CBratObject value
withExcept	: 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

obMap	: **CObMap** (p. 241) to insert
withExcept	: true for exception handling, flse otherwise

References brathl::CObDoubleMap::Insert().

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

Inserts a DoublePtr* object

Parameters

key	: DoublePtr* name (std::map key)

value	: DoublePtr* value
withExcept	: 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

key	: keymap
value	: pointer value
withExcept	: 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

obMap	: **CPtrMap** (p. 275) to insert
withExcept	: 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 identifiy by 'key'.

Parameters

key	: std::string keyword

Returns

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

References brathl::CIntMap::Exists().

6.2.4.76 uint32_t brathl::CUIntMap::operator[] (const std::string & *key*) [virtual]

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

Parameters

key	: std::string keyword

Returns

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

References brathl::CUIntMap::Exists().

6.2.4.77 double brathl::CDoubleMap::operator[] (const std::string & *key*) [virtual]

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

Parameters

key	: std::string keyword

Returns

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

References brathl::CDoubleMap::Exists().

6.2.4.78 CBratObject * brathl::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

key	: CBratObject keyword

Returns

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

6.2.4.79 CBratObject * brathl::CObIntMap::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

key	: 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 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

key	: 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 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

key	: 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 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 void *p = myMap[key] –> use Exists method instead ;

Parameters

key	: 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::CObIntMap::RemoveAll ( ) [virtual]

Remove all elements and clear the std::map

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

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

oldKey	: old key
newKey	: new key

Returns

true if key is renamed, otherwise false

References brathl::CObMap::Insert().

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

Rename a key

Parameters

oldKey	: old key
newKey	: new key

Returns

true if key is renamed, otherwise false

References brathl::CObIntMap::Insert().

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

Rename a key

Parameters

oldKey	: old key
newKey	: new key

Returns

true if key is renamed, otherwise false

References brathl::CObDoubleMap::Insert().

6.2.4.102 bool brathl::CDoublePtrDoubleMap::RenameKey (double *oldKey,* double *newKey*)

Rename a key

Parameters

oldKey	: old key
newKey	: new key

Returns

true if key is renamed, otherwise false

References brathl::CDoublePtrDoubleMap::Insert().

6.2.4.103 void brathl::CArrayStringMap::Set (const CArrayStringMap & *a*) [virtual]

Inserts a std::string std::map

Parameters

strmap	: std::map to insert
withExcept	: 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 brathl::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::CCriteriaInfo**
- 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 **brathl::CCriteria::CriteriaKind** {
  brathl::CCriteria::UNKNOWN, **brathl::CCriteria::LATLON**, **brathl::CCriteria::DATETIME**, **brathl::C↩ Criteria::PASS**,
  brathl::CCriteria::CYCLE }

Functions

- void **brathl::CProduct::AddCriteria** (bool force=false)
- void **brathl::CProduct::AddCriteria** (**CCriteria** ∗criteria, bool erase=true)
- void **brathl::CProduct::AddCriteria** (CProduct ∗product)
- void **brathl::CMapProduct::AddCriteriaToProducts** ()
- virtual void **brathl::CProduct::AddInternalHighResolutionFieldCalculation** ()
- CInfo ∗ **brathl::CProduct::CListInfo::AddNew** ()
- virtual void **brathl::CProduct::AddOffset** (double value, **CField** ∗field=NULL)
- bool **brathl::CProduct::AddRecordNameToField** (const CExpression &expr, const std::string &dataSet↩ Name, CExpression &exprOut, std::string &errorMsg)
- bool **brathl::CProduct::AddRecordNameToField** (const std::string &in, const std::string &dataSetName, std::string &out, std::string &errorMsg)
- bool **brathl::CProduct::AddRecordNameToField** (const std::string &in, const std::string &dataSetName, const CStringArray &fieldsIn, std::string &out, std::string &errorMsg)
- bool **brathl::CProduct::AddRecordNameToField** (CProductAliases ∗productAliases, std::string &errorMsg)
- virtual void **brathl::CProduct::AddSameFieldName** (const std::string &fieldNameToSearch, CStringArray &arrayFieldsAdded)
- void **brathl::CCriteriaPassInt::Adjust** ()
- virtual bool **brathl::CProduct::ApplyCriteria** (**CStringList** &filteredFileList, CProgressInterface ∗pi, const std::string &log_file="")
- virtual bool **brathl::CProduct::ApplyCriteriaCycle** (**CCriteriaInfo** ∗criteriaInfo)
- virtual bool **brathl::CProduct::ApplyCriteriaDatetime** (**CCriteriaInfo** ∗criteriaInfo)
- virtual bool **brathl::CProduct::ApplyCriteriaLatLon** (**CCriteriaInfo** ∗criteriaInfo)
- virtual bool **brathl::CProduct::ApplyCriteriaPass** (**CCriteriaInfo** ∗criteriaInfo)
- virtual bool **brathl::CProduct::ApplyCriteriaPassInt** (**CCriteriaInfo** ∗criteriaInfo)
- virtual bool **brathl::CProduct::ApplyCriteriaPassString** (**CCriteriaInfo** ∗criteriaInfo)
- CInfo ∗ **brathl::CProduct::CListInfo::Back** (bool withExcept=true)
- void **brathl::CProduct::BuildCriteriaFieldsToRead** (CRecordDataMap &listRecord)
- **brathl::CCriteriaPass::CCriteriaPass** ()

  *Empty **CCriteriaPass*** (p. *162*) *ctor.*
- **brathl::CCriteriaPassInt::CCriteriaPassInt** ()

  *Empty **CCriteriaPassInt*** (p. *164*) *ctor.*
- **brathl::CCriteriaPassInt::CCriteriaPassInt** (**CCriteriaPassInt** &c)
- **brathl::CCriteriaPassInt::CCriteriaPassInt** (**CCriteriaPassInt** ∗c)
- **brathl::CCriteriaPassInt::CCriteriaPassInt** (int32_t from, int32_t to)
- **brathl::CCriteriaPassInt::CCriteriaPassInt** (const std::string &from, const std::string &to)
- **brathl::CCriteriaPassInt::CCriteriaPassInt** (const CStringArray &array)
- **brathl::CCriteriaPassString::CCriteriaPassString** ()

  *Empty **CCriteriaPassString*** (p. *166*) *ctor.*
- **brathl::CCriteriaPassString::CCriteriaPassString** (**CCriteriaPassString** &c)
- **brathl::CCriteriaPassString::CCriteriaPassString** (**CCriteriaPassString** ∗c)
- **brathl::CCriteriaPassString::CCriteriaPassString** (const std::string &passes, const std::string &delimiter=C↩ CriteriaPassString::m_delimiter)
- **brathl::CCriteriaPassString::CCriteriaPassString** (const CStringArray &array)
- static bool **brathl::CProduct::CheckAliases** (const std::string &fileName, CStringArray &errors)
- bool **brathl::CProduct::CheckAliases** (CStringArray &errors)

- virtual void **brathl::CProduct::CheckConsistencyHighResolutionField** (**CFieldSetArrayDbl** *fieldSet↩ ArrayDbl)
- 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::CodaInit** ()
- 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 function.
- 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, **CString↩ List** &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 with↩ Except=true)
- **CField** ∗ **brathl::CProduct::FindFieldByName** (const std::string &fieldName, const std::string &dataSet↩ Name, 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 with↩ Except=true)
- static **CCriteriaPassInt** ∗ **brathl::CCriteriaPassInt::GetCriteria** (CBratObject ∗ob, bool withExcept=true)
- **CCriteria** ∗ **brathl::CProduct::GetCriteria** (**CCriteriaInfo** ∗criteriaInfo)
- virtual std::string **brathl::CProduct::GetCurrentFileName** ()
- virtual int32_t **brathl::CProduct::GetCurrentRecordNumber** ()
- **CCriteriaCycle** ∗ **brathl::CProduct::GetCycleCriteria** ()
- **CCriteriaCycleInfo** ∗ **brathl::CProduct::GetCycleCriteriaInfo** ()
- CStringArray ∗ **brathl::CProduct::GetDataDictionaryFieldNames** (bool forceReload=false)
- CStringArray ∗ **brathl::CProduct::GetDataDictionaryFieldNamesWithDatasetName** (bool force↩ Reload=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::GetDatetimeCriteriaInfo** ()
- 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 **brathl::CProduct::GetForceReadDataOneByOne** ()
- int32_t **brathl::CCriteriaPassInt::GetFrom** ()
- int_t **brathl::CProduct::GetIndexProcessedFile** ()
- bool **brathl::CProduct::GetInfoArray** ()
- bool **brathl::CProduct::GetInfoRecord** (int32_t nbDims=1, const long dim[]=DEFAULT_DIM)
- bool **brathl::CProduct::GetInfoSpecial** (int32_t nbDims=1, const long dim[]=DEFAULT_DIM)
- static **CMapProduct** & **brathl::CMapProduct::GetInstance** ()
- virtual const std::string & **brathl::CProduct::GetLabel** () const
- virtual std::string **brathl::CProduct::GetLabelForCyclePass** () const
- virtual std::string **brathl::CProduct::GetLatitudeFieldName** ()
- **CCriteriaLatLon** ∗ **brathl::CProduct::GetLatLonCriteria** ()
- **CCriteriaLatLonInfo** ∗ **brathl::CProduct::GetLatLonCriteriaInfo** ()
- virtual bool **brathl::CProduct::GetLatLonMinMax** (double &latMin, double &lonMin, double &latMax, double &lonMax, CProgressInterface ∗pi=nullptr)
- virtual bool **brathl::CProduct::GetLatLonMinMax** (CLatLonRect &latlonRectMinMax, CProgressInterface ∗pi=nullptr)
- **CStringList** ∗ **brathl::CProduct::GetListFieldOrigin** ()
- virtual std::string **brathl::CProduct::GetLongitudeFieldName** ()
- virtual void **brathl::CProduct::GetMinMaxNumberOfRecords** (int32_t &min, int32_t &max, **CIntMap** ∗datasetRecordsNumber=NULL, int32_t minThreshold=-1)
- void **brathl::CProduct::GetNamesCaseSensitive** (const CStringArray &fieldsIn, CStringArray &fieldsOut↩ NoCaseSensitive, CStringArray &fieldsOutCaseSensitive, bool forceReload=false)
- virtual int32_t **brathl::CProduct::GetNumberOfRecords** ()
- virtual int32_t **brathl::CProduct::GetNumberOfRecords** (const std::string &dataSetName)
- virtual void **brathl::CProduct::GetNumberOfRecords** (const **CStringList** &datasetNames, **CIntMap** &datasetRecordsNumber)
- virtual double **brathl::CProduct::GetOffset** ()
- **CCriteriaPass** ∗ **brathl::CProduct::GetPassCriteria** ()
- **CCriteriaPassInfo** ∗ **brathl::CProduct::GetPassCriteriaInfo** ()
- CStringArray ∗ **brathl::CCriteriaPassString::GetPasses** ()
- **CCriteriaPassInt** ∗ **brathl::CProduct::GetPassIntCriteria** ()
- **CCriteriaPassIntInfo** ∗ **brathl::CProduct::GetPassIntCriteriaInfo** ()
- **CCriteriaPassString** ∗ **brathl::CProduct::GetPassStringCriteria** ()
- **CCriteriaPassStringInfo** ∗ **brathl::CProduct::GetPassStringCriteriaInfo** ()
- int32_t **brathl::CProduct::GetPerformBoundaryChecks** ()
- int32_t **brathl::CProduct::GetPerformConversions** ()
- const std::string & **brathl::CProduct::GetProductClass** () const
- std::string **brathl::CProduct::GetProductClassAndType** ()
- void **brathl::CMapProduct::GetProductKeysWithCriteria** (CStringArray &keys)
- **CProductList** & **brathl::CProduct::GetProductList** ()
- const std::string & **brathl::CProduct::GetProductType** () const
- std::string **brathl::CProduct::GetRecordFieldName** ()
- virtual void **brathl::CProduct::GetRecords** (CStringArray &array)
- static int_t **brathl::CProduct::GetRefCount** ()
- **brathl_refDate brathl::CProduct::GetRefDate** () const
- **CDate brathl::CProduct::GetRefDateAsDate** ()
- void **brathl::CProduct::GetRootType** ()
- uint32_t **brathl::CProduct::GetSkippedRecordCount** ()
- int32_t **brathl::CCriteriaPassInt::GetTo** ()
- **CTreeField** ∗ **brathl::CProduct::GetTreeField** ()
- std::string **brathl::CProduct::GetTypeDesc** ()
- std::string **brathl::CProduct::GetTypeDesc** (coda_Type ∗type)
- std::string **brathl::CProduct::GetTypeName** ()
- std::string **brathl::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_ER↩ ROR)
- virtual bool **brathl::CProduct::HasAliases** ()
- virtual bool **brathl::CProduct::HasCompatibleDims** (const std::string &value, std::string &msg, bool use↩ VirtualDims, CUIntArray ∗commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const std::string &value, const std::string &dataSet↩ Name, std::string &msg, bool useVirtualDims, CUIntArray ∗commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CExpression &expr, std::string &msg, bool use↩ VirtualDims, CUIntArray ∗commonDimensions=NULL)
- virtual bool **brathl::CProduct::HasCompatibleDims** (const CExpression &expr, const std::string &dataSet↩ Name, 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::HasCriteriaInfo** ()
- bool **brathl::CProduct::HasCycleCriteria** ()
- bool **brathl::CProduct::HasCycleCriteriaInfo** ()
- bool **brathl::CProduct::HasDatetimeCriteria** ()
- bool **brathl::CProduct::HasDatetimeCriteriaInfo** ()
- 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::HasLatLonCriteriaInfo** ()
- bool **brathl::CProduct::HasPassCriteria** ()
- bool **brathl::CProduct::HasPassCriteriaInfo** ()
- bool **brathl::CProduct::HasPassIntCriteria** ()
- bool **brathl::CProduct::HasPassIntCriteriaInfo** ()
- bool **brathl::CProduct::HasPassStringCriteria** ()
- bool **brathl::CProduct::HasPassStringCriteriaInfo** ()
- 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::InitCriteriaInfo** ()
- virtual void **brathl::CProduct::InitDateRef** ()=0
- virtual void **brathl::CProductGeneric::InitDateRef** ()
- virtual void **brathl::CProduct::InitInternalFieldName** (const std::string &dataSetName, **CStringList** &list↩ Field, 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::IsZFXY** () 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, **C**↩ **StringList** &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, **CFieldSetDbl** ∗fieldSetDbl, uint32_t repeat, uint32_t insertRecordAt=0)
- virtual void **brathl::CProduct::Put** (**CDataSet** ∗dataSet, **CFieldSetArrayDbl** ∗fieldSetArrayDbl, uint32_↩ t repeat, uint32_t insertRecordAt=0)
- virtual void **brathl::CProduct::Put** (**CDataSet** ∗dataSet, **CFieldSetDbl** ∗fieldSetDbl)
- virtual void **brathl::CProduct::PutFlat** (**CDataSet** ∗dataSet, **CFieldSetArrayDbl** ∗fieldSetArrayDbl, uint32↩ _t insertRecordAt=0)
- virtual void **brathl::CProduct::PutFlatHighResolution** (**CDataSet** ∗dataSet, **CFieldSetArrayDbl** ∗fieldSet↩ ArrayDbl)
- 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** ∗field↩ SpecificUnit=NULL)
- static void **brathl::CProduct::ReadDataForOneMeasure** (**CDataSet** ∗dataSet, const std::string &record↩ Name, CExpression &Select, std::vector< CExpression > &Expressions, const std::vector< CUnit > &WantedUnits, double ∗∗results, int32_t ∗sizes, size_t ∗actualSize, int ignoreOutOfRange, int statistics, C↩ Product ∗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 force↩ Reload=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=CCriteria↩ PassString::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=C↩ CriteriaPassInt::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**
- **CObIntMap brathl::CProduct::m_criteriaInfoMap**
- **CObIntMap 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 **brathl::CProduct::m_previousLongitude**
- double **brathl::CProduct::m_previousTimeStamp**
- std::string **brathl::CProductList::m_productClass**
- coda_format **brathl::CProductList::m_productFormat**
- std::string **brathl::CProductList::m_productType**
- size_t **brathl::CProduct::m_recordCount**
- **brathl_refDate brathl::CProduct::m_refDate**
- int32_t **brathl::CProduct::m_refPoint**
- int32_t **brathl::CCriteriaPassInt::m_to**
- uint32_t **brathl::CProduct::m_traceProcessRecordRatio**
- static const char * **brathl::CProduct::m_transposeFieldValuesFileName** = "brathl_transposefieldvalues.↩ txt"
- **CTreeField brathl::CProduct::m_tree**
- static const std::string **brathl::CProduct::m_treeRootName** = "Root"
- coda_Type * **brathl::CProduct::CInfo::m_type**
- coda_type_class **brathl::CProduct::CInfo::m_type_class**
- static const uint32_t **brathl::CProduct::MAX_INDEX** = 4
- std::string **brathl::CProductList::mCodaProductClass**
- std::string **brathl::CProductList::mCodaProductType**
- static const uint32_t **brathl::CProduct::MEAN_INDEX** = 1
- static const uint32_t **brathl::CProduct::MIN_INDEX** = 3
- std::string **brathl::CProduct::mLabel**
- static const int32_t **brathl::CProduct::NUMBER_OF_STATISTICS** = 5
- static const uint32_t **brathl::CProduct::STDDEV_INDEX** = 2

6.3.1 Detailed Description

6.3.2 Enumeration Type Documentation

6.3.2.1 enum brathl::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 brathl::CCriteriaPassInt::CCriteriaPassInt (int32_t *from,* int32_t *to*)

Constructor.

Parameters

from	start pass
to	end pass

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

Constructor.

Parameters

from	start pass
to	end pass

6.3.3.3 brathl::CCriteriaPassInt::CCriteriaPassInt (const CStringArray & *array*)

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

Parameters

array	start and end dates

6.3.3.4 brathl::CCriteriaPassString::CCriteriaPassString (const std::string & *passes,* const std::string & *delimiter* = `CCriteriaPassString::m_delimiter`)

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

Parameters

passes	passes to set

References brathl::CCriteriaPassString::Set().

6.3.3.5 brathl::CCriteriaPassString::CCriteriaPassString (const CStringArray & *array*)

Constructor from a array that contains passes

Parameters

array	start and end dates

References brathl::CCriteriaPassString::Set().

6.3.3.6 brathl::CProduct::CProduct (const std::string & *fileName*) `[protected]`

Creates new CProduct object

Parameters

fileName	[in] : file name to be connected

6.3.3.7 brathl::CProduct::CProduct (const CStringList & *fileNameList,* bool *check_only_first_file*) `[protected]`

Creates new CProduct object

Parameters

fileNameList	[in] : list of file to be connected

6.3.3.8 brathl::CProductGeneric::CProductGeneric (const std::string & *fileName*) `[inline]`

Creates new CProdCProductGenericuct object

Parameters

fileName	[in] : file name to be connected

6.3.3.9 brathl::CProductGeneric::CProductGeneric (const CStringList & *fileNameList,* bool *check_only_first_file*) `[inline]`

Creates new CProductGeneric object

Parameters

fileNameList	[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

o	[in] : productList object to be copied

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

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

Parameters

fileName	[in] : file name to be connected

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

Creates new CProduct object

Parameters

fileNameList	[in] : list of file to be connected

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

Creates new CProduct object

Parameters

fileNameArray	[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

passes	intersect with this
intersect	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

passes	intersect with this
intersect	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

array	that contains start pass as std::string, end pass as std::string
intersect	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

array	that contains start pass as std::string, end pass as std::string
intersect	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

array	that contains start pass as std::string, end pass as std::string
intersect	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

array	that contains start pass as std::string, end pass as std::string
intersect	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

o	[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

passes	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

array	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

from	start pass
to	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

from	start pass
to	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

array	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

to	start pass

References brathl::CCriteriaPassInt::m_from.

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

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

Sets start pass

Parameters

to	start pass

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

6.3.3.35 void brathl::CCriteriaPassInt::SetTo (int32_t *to*)

Sets end pass

Parameters

to	end pass

References brathl::CCriteriaPassInt::m_to.

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

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

Sets end pass

Parameters

to	end pass

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

6.3.4 Variable Documentation

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

Product management class.

Version

1.0

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

start pass

Referenced by brathl::CCriteriaPassInt::Dump(), brathl::CCriteriaPassInt::IsDefaultValue(), brathl::CCriteriaPass↩Int::SetDefaultValue(), and brathl::CCriteriaPassInt::SetFrom().

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

Number of records to read

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

Date period

Referenced by brathl::CCriteriaPassString::Dump(), brathl::CCriteriaPassString::Intersect(), brathl::CCriteriaPass↩String::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::CCriteriaPass↩Int::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 **brathl::CFile**

Enumerations

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

6.5.1 Detailed Description

6.5.2 Enumeration Type Documentation

6.5.2.1 enum brathl::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

key	: parameter name (std::map key)
value	: parameter value

Returns

CParameter (p. 242) oject or NULL if error

References brathl::CParameter::AddValue().

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

operator[] redefinition. Searches a **CParameter** (p. 242) 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 **CParameter** (p. 242) *p = m_mapParam[key] –> use Exists method instead ;

Parameters

key	: parameter keyword

Returns

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

6.6.2.6 void brathl::CMapParameter::RemoveAll ( )

Remove all elements and clear the std::map

Referenced by brathl::CFileParams::Load(), and brathl::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	*cycle*	: number of cycle to convert
in	*pass*	: number of pass in the cycle to convert
out	*dateYMDHMSM*	: 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	*dateYMDHMSM*	: date
out	*dayOfYear*	: 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	*dateDSM1*	: date1
in	*dateDSM2*	: date2
out	*diff*	: 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	*dateJulian1*	: date1
in	*dateJulian2*	: date2
out	*diff*	: 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	*dateYMDHMSM1*	: date1
in	*dateYMDHMSM2*	: date2
out	*diff*	: 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	*dateDSM*	: date to convert
in	*refDate*	: date reference conversion
out	*dateJulian*	: 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	*dateDSM*	: date to convert
in	*refDate*	: date reference conversion
out	*dateSeconds*	: 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	*dateDSM*	: date to convert
out	*dateYMDHMSM*	: 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	*dateJulian*	: date to convert
in	*refDate*	: date reference conversion
out	*dateDSM*	: 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	*dateJulian*	: date to convert
in	*refDate*	: date reference conversion
out	*dateSeconds*	: 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	*dateJulian*	: date to convert
out	*dateYMDHMSM*	: 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	*dateYMDHMSM*	: 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	*dateSeconds*	: date to convert
in	*refDate*	: date reference conversion
out	*dateDSM*	: 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	*dateSeconds*	: date to convert
in	*refDate*	: date reference conversion
out	*dateJulian*	: 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().

6.7.2.15 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	*dateSeconds*	: date to convert
out	*dateYMDHMSM*	: 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	*mission*	: mission type (see **brathl_mission** (p. 317))
in	*dateYMDHMSM*	: date to convert
out	*cycle*	: number of cycle
out	*pass*	: 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	*dateYMDHMSM*	: date to convert
in	*refDate*	: date reference conversion
out	*dateDSM*	: 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	*dateYMDHMSM*	: date to convert
in	*refDate*	: date reference conversion
out	*dateJulian*	: 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().

6.7.2.19 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	*dateYMDHMSM*	: date to convert
in	*refDate*	: date reference conversion
out	*dateSeconds*	: 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	*nbFiles*	: Number of files in file name list This is the usable size of #fileNames
in	*fileNames*	: 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	*selection*	: 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	*nbData*	: Number of expression used to retreive data
in	*dataExpressions*	: Expression applyed 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	*units*	: 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 applyed to the data of the corresponding expression. When a unit conversion has to be applyed, 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 (internaly all data are converted to base unit of the actual fields unit which is coherent with the above assumption).
	results	[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.
	sizes	[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	*actualSize*	: 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	*ignoreOutOfRange*	: 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	*statistics*	: 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: • **Count** of *valid* 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 In this case actualSize always returns 5
in	*defaultValue*	: 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	*dateRefUser*	: date string (format: YYYY-MM-DD HH:MN: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

dateRefUser	: date string (format: YYYY-MM-DD HH:MN: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	*err*	: error code
out	*str*	: 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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iSeconds*	: date to convert
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oDays*	: numbers of days
out	*oSeconds*	: number of seconds

out	*oMuSeconds*	: 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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iDays*	: numbers of days
in	*iSeconds*	: number of seconds
in	*iMuSeconds*	: numbers of microseconds
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oSeconds*	: 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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iJulian*	: decimal julian date
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oDays*	: numbers of days
out	*oSeconds*	: number of seconds

out	*oMuSeconds*	: 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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iDays*	: numbers of days
in	*iSeconds*	: number of seconds
in	*iMuSeconds*	: numbers of microseconds
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oJulian*	: 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,iRef↩ DateDest,oDays,oSeconds,oMuSeconds)

INTEGER∗4 iYear, INTEGER∗4 iMonth, INTEGER∗4 iDay, INTEGER∗4 iHour, INTEGER∗4 iMinute, INTEGE↩ R∗4 iSecond, INTEGER∗4 iMuSecond, INTEGER∗4 iRefDateDest, INTEGER∗4 oDays, INTEGER∗4 oSeconds, INTEGER∗4 oMuSeconds

Parameters

in	*iYear*	: year
in	*iMonth*	: month
in	*iDay*	: day
in	*iHour*	: hour

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

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_YMDHMSM2DSM(), _structDateDSM::days, _structDateDSM::muSeconds, and _structDateD↩SM::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,o↩Day,oHour,oMinute,oSecond,oMuSecond)

INTEGER∗4 iRefDateSrc, INTEGER∗4 iDays, INTEGER∗4 iSeconds, INTEGER∗4 iMuSeconds INTEGER∗4 o↩Year, INTEGER∗4 oMonth, INTEGER∗4 oDay, INTEGER∗4 oHour, INTEGER∗4 oMinute, INTEGER∗4 oSecond, INTEGER∗4 oMuSecond,

Parameters

in	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iDays*	: numbers of days
in	*iSeconds*	: number of seconds
in	*iMuSeconds*	: numbers of microseconds
out	*oYear*	: year
out	*oMonth*	: month
out	*oDay*	: day
out	*oHour*	: hour convert
out	*oMinute*	: minute to convert
out	*oSecond*	: second to convert
out	*oMuSecond*	: 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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iSeconds*	: number of seconds
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oJulian*	: julian date

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

References brathl_Seconds2Julian(), _structDateJulian::julian, _structDateSecond::nbSeconds, and _structDate↩Second::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	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iJulian*	: julian date
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oSeconds*	: number of seconds

Returns

#BRATHL_SUCCESS or error code (see Date_error_codes)

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

6.9.2.13 INTEGER4 FTN_NAME (brathlf_seconds2ymdhmsm , BRATHLF_SECONDS2YMDHMSM)

Converts seconds into 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,o↩Minute,oSecond,oMuSecond)

Fortran specification

INTEGER∗4 iRefDateSrc, INTEGER∗4 iSeconds, INTEGER∗4 iRefDateDest INTEGER∗4 oYear, INTEGER∗4 o↩Month, INTEGER∗4 oDay, INTEGER∗4 oHour, INTEGER∗4 oMinute, INTEGER∗4 oSecond, INTEGER∗4 oMu↩Second,

Parameters

in	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iSeconds*	: number of seconds

out	*oYear*	: year
out	*oMonth*	: month
out	*oDay*	: day
out	*oHour*	: hour convert
out	*oMinute*	: minute to convert
out	*oSecond*	: second to convert
out	*oMuSecond*	: 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,iRef↩ DateDest,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	*iYear*	: year
in	*iMonth*	: month
in	*iDay*	: day
in	*iHour*	: hour
in	*iMinute*	: minute
in	*iSecond*	: second
in	*iMuSecond*	: micro-second
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oSeconds*	: 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,o↩ Minute,oSecond,oMuSecond)

Fortran specification

INTEGER∗4 iRefDateSrc, REAL∗8 iJulian, INTEGER∗4 oYear, INTEGER∗4 oMonth, INTEGER∗4 oDay, INTEG↩ ER∗4 oHour, INTEGER∗4 oMinute, INTEGER∗4 oSecond, INTEGER∗4 oMuSecond,

Parameters

in	*iRefDateSrc*	: source date reference (see **brathl_refDate** (p. 317))
in	*iJulian*	: julian date
out	*oYear*	: year
out	*oMonth*	: month
out	*oDay*	: day
out	*oHour*	: hour convert
out	*oMinute*	: minute to convert
out	*oSecond*	: second to convert
out	*oMuSecond*	: 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	*iYear*	: year
in	*iMonth*	: month
in	*iDay*	: day
in	*iHour*	: hour
in	*iMinute*	: minute
in	*iSecond*	: second
in	*iMuSecond*	: micro-second
in	*iRefDateDest*	: date reference conversion (see **brathl_refDate** (p. 317))
out	*oJulian*	: 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	*oYear*	: year
out	*oMonth*	: month
out	*oDay*	: day
out	*oHour*	: hour convert
out	*oMinute*	: minute to convert
out	*oSecond*	: second to convert
out	*oMuSecond*	: 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	*iYear*	: year
in	*iMonth*	: month
in	*iDay*	: day
in	*iHour*	: hour
in	*iMinute*	: minute
in	*iSecond*	: second
in	*iMuSecond*	: micro-second
out	*oQuant*	: 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	*iYear1*	: year
in	*iMonth1*	: month
in	*iDay1*	: day
in	*iHour1*	: hour
in	*iMinute1*	: minute
in	*iSecond1*	: second
in	*iMuSecond1*	: micro-second
in	*iYear2*	: year
in	*iMonth2*	: month
in	*iDay2*	: day
in	*iHour2*	: hour
in	*iMinute2*	: minute
in	*iSecond2*	: second
in	*iMuSecond2*	: micro-second
out	*diff*	: 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	*iRefDate1*	: source date reference (see **brathl_refDate** (p. 317))
in	*iDays1*	: numbers of days
in	*iSeconds1*	: number of seconds
in	*iMuSeconds1*	: numbers of microseconds
in	*iRefDate2*	: source date reference (see **brathl_refDate** (p. 317))
in	*iDays2*	: numbers of days
in	*iSeconds2*	: number of seconds
in	*iMuSeconds2*	: numbers of microseconds
out	*diff*	: 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	*iRefDate1*	: source date reference (see **brathl_refDate** (p. 317))
in	*iJulian1*	: julian date
in	*iRefDate2*	: source date reference (see **brathl_refDate** (p. 317))
in	*iJulian2*	: julian date
out	*diff*	: 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 cyle/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	*iMission*	: mission type (see **brathl_mission** (p. 317))
in	*iCycle*	: number of cycle to convert
in	*iPass*	: number of pass in the cycle to convert
out	*oYear*	: year
out	*oMonth*	: month
out	*oDay*	: day
out	*oHour*	: hour
out	*oMinute*	: minute
out	*oSecond*	: second
out	*oMuSecond*	: 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	*iMission*	: mission type (see **brathl_mission** (p. 317))
in	*iYear*	: year
in	*iMonth*	: month
in	*iDay*	: day
in	*iHour*	: hour
in	*iMinute*	: minute
in	*iSecond*	: second
in	*iMuSecond*	: micro-second
out	*oCycle*	: number of cycle to convert
out	*oPass*	: 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 INTEGER∗4 nbData CHARACTER∗(∗) dataExpressions(nbData) CHARACTER∗(∗) units(nbData) REAL∗8 results(size,nbData) INTEGER∗4 size INTEGER∗4 actualSize INTEGER∗4 ignoreOutOfRange INTEGER∗4 statistics REAL∗8 defaultValue

Parameters

in	*nbFiles*	: Number of files in file name list This is the usable size of #fileNames
in	*fileNames*	: File name list Must contain at least #nbFiles entries. If an entry is an empty string, the entry is ignored.
in	*selection*	: 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	*nbData*	: Number of expressions used to retreive data
in	*dataExpressions*	: Expression applyed 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	*units*	: Wanted unit for each expression Must contain at least #nbData entries. If an entry is an empty string, no unit conversion is applyed to the data of the corresponding expression. When a unit conversion has to be applyed, 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 (internaly all data are converted to base unit of the actual fields unit which is coherent with the above assumption).

out	*results*	: Data read Must be a matrix of at least #nbData entries of size values. Must be declared as real∗8 results[size, N] where N >= #nbData
in	*size*	: Number of data in each #results entry. Must be >=0. If 0, nothing is returned in results and results can be anything.
out	*actualSize*	: 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 if 0 this the number of values that would have been returned.
in	*ignoreOutOf↩ Range*	: 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.
in	*statistics*	: 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: • **Count** of *valid* 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 In this case actualSize always returns 5
in	*defaultValue*	: 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 **CCriteriaDatetimeInfo**
- class **CCriteriaInfo**
- 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**

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, CVectorBrat↩ AlgorithmParam &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_STANDARD** = "Generic NetCdf Standard"
- const std::string **GENERIC_NETCDF_TYPE_VARIANT_1** = "Generic NetCdf Variant 1"
- 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 **ZFXY_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> brathl::mapdouble

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

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

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

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

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

7.1.2.7 typedef std::map<std::string, int32_t> brathl::mapint

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

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

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

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

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

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

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

7.1.2.11 typedef std::map<std::string, std::string> brathl::mapstring

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

7.1.2.12 typedef std::map<std::string, uint32_t> brathl::mapuint

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

7.1.2.13 typedef std::vector<CBratObject*> brathl::obarray

Creates a type name for object array

7.1.2.14 typedef std::list<CBratObject*> brathl::oblist

Creates a type name for object std::list

7.1.2.15 typedef std::stack<CBratObject*> brathl::obstack

Creates a type name for object std::stack

7.1.2.16 typedef std::vector<std::string> brathl::stringarray

Creates a type name for std::string array

7.1.2.17 typedef std::list<std::string> brathl::stringlist

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

7.1.3 Variable Documentation

7.1.3.1 const DefCharFunction1 brathl::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:MN: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
       Microseconds'"
                                         "\n Seconds must be stricty less 86400 and Microseconds must be
       stricty less than 1000000"
                                         "\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** &copy)
- 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** &copy)
- 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** &copy)
- 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 contructor

8.7.2.2 brathl::CBratAlgoFilterGaussian1D::CBratAlgoFilterGaussian1D (const CBratAlgoFilterGaussian1D & *copy*)

Copy contructor

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 brathl::CBratAlgoFilterGaussian1D::GetName ( ) const [inline], [override], [virtual]

Gets the name of the algorithm

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

Referenced by Run().

8.7.3.4 CBratAlgoFilterGaussian1D & brathl::CBratAlgoFilterGaussian1D::operator= (const CBratAlgoFilterGaussian1D & *copy*)

Overloads operator '='

8.7.3.5 double brathl::CBratAlgoFilterGaussian1D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

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

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

- BratAlgoFilterGaussian1D.h
- BratAlgoFilterGaussian1D.cpp

8.8 brathl::CBratAlgoFilterGaussian2D Class Reference

`#include <BratAlgoFilterGaussian2D.h>`

Inherits brathl::CBratAlgoFilterGaussian.

Public Member Functions

- **CBratAlgoFilterGaussian2D** ()
- **CBratAlgoFilterGaussian2D** (const **CBratAlgoFilterGaussian2D** &copy)
- 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** &copy)
- 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** &copy)
- 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 brathl::CBratAlgoFilterGaussian2D::CBratAlgoFilterGaussian2D ( )

Default contructor

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

Copy contructor

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

Destructor

8.8.3 Member Function Documentation

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

Dump function

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

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

Gets the description of the algorithm

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

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

Gets the name of the algorithm

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

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

Overloads operator '='

8.8.3.5 double brathl::CBratAlgoFilterGaussian2D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[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:

- BratAlgoFilterGaussian2D.h
- BratAlgoFilterGaussian2D.cpp

8.9 brathl::CBratAlgoFilterLanczos1D Class Reference

`#include <BratAlgoFilterLanczos1D.h>`

Inherits brathl::CBratAlgoFilterLanczos.

Public Member Functions

- **CBratAlgoFilterLanczos1D** ()
- **CBratAlgoFilterLanczos1D** (const **CBratAlgoFilterLanczos1D** &copy)
- 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** &copy)
- 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** &copy)
- 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 brathl::CBratAlgoFilterLanczos1D::CBratAlgoFilterLanczos1D ( )

Default contructor

8.9.2.2 brathl::CBratAlgoFilterLanczos1D::CBratAlgoFilterLanczos1D (const CBratAlgoFilterLanczos1D & *copy*)

Copy contructor

8.9.2.3 virtual brathl::CBratAlgoFilterLanczos1D::~CBratAlgoFilterLanczos1D ( ) [inline], [virtual]

Destructor

8.9.3 Member Function Documentation

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

Dump function

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

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

Gets the description of the algorithm

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

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

Gets the name of the algorithm

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

Referenced by Run().

8.9.3.4 CBratAlgoFilterLanczos1D & brathl::CBratAlgoFilterLanczos1D::operator= (const CBratAlgoFilterLanczos1D & *copy*)

Overloads operator '='

8.9.3.5 double brathl::CBratAlgoFilterLanczos1D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

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

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

- BratAlgoFilterLanczos1D.h
- BratAlgoFilterLanczos1D.cpp

8.10 brathl::CBratAlgoFilterLanczos2D Class Reference

`#include <BratAlgoFilterLanczos2D.h>`

Inherits brathl::CBratAlgoFilterLanczos.

Public Member Functions

- **CBratAlgoFilterLanczos2D** ()
- **CBratAlgoFilterLanczos2D** (const **CBratAlgoFilterLanczos2D** &copy)
- 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** &copy)
- 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** &copy)
- 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 brathl::CBratAlgoFilterLanczos2D::CBratAlgoFilterLanczos2D ( )

Default contructor

8.10.2.2 brathl::CBratAlgoFilterLanczos2D::CBratAlgoFilterLanczos2D (const CBratAlgoFilterLanczos2D & *copy*)

Copy contructor

8.10.2.3 brathl::CBratAlgoFilterLanczos2D::~CBratAlgoFilterLanczos2D ( ) [virtual]

Destructor

8.10.3 Member Function Documentation

8.10.3.1 void brathl::CBratAlgoFilterLanczos2D::Dump (std::ostream & *fOut* = `std::cerr`) [override], [virtual]

Dump function

Reimplemented from **brathl::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

fmt	[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
args	[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** &copy)
- 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** &copy)
- 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** &copy)
- 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 contructor

8.11.2.2 brathl::CBratAlgoFilterLoess1D::CBratAlgoFilterLoess1D (const CBratAlgoFilterLoess1D & *copy*)

Copy contructor

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

indexParam	[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 CBrat↩ AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

indexParam	[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

indexParam	[in] : parameter index.

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

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

8.11.3.6 virtual std::string brathl::CBratAlgoFilterLoess1D::GetName ( ) const [inline], [override], [virtual]

Gets the name of the algorithm

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

Referenced by Run().

8.11.3.7 virtual uint32_t brathl::CBratAlgoFilterLoess1D::GetNumInputParam ( ) const [inline], [override], [virtual]

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

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

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.11.3.8 virtual std::string brathl::CBratAlgoFilterLoess1D::GetOutputUnit ( ) const [inline], [override], [virtual]

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

Parameters

indexParam	[in] : parameter index.

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

8.11.3.9 CBratAlgoFilterLoess1D & brathl::CBratAlgoFilterLoess1D::operator= (const CBratAlgoFilterLoess1D & *copy*)

Overloads operator '='

8.11.3.10 double brathl::CBratAlgoFilterLoess1D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

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

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

- BratAlgoFilterLoess1D.h
- BratAlgoFilterLoess1D.cpp

8.12 brathl::CBratAlgoFilterLoess2D Class Reference

`#include <BratAlgoFilterLoess2D.h>`

Inherits brathl::CBratAlgoFilterLoess.

Public Member Functions

- **CBratAlgoFilterLoess2D** ()
- **CBratAlgoFilterLoess2D** (const **CBratAlgoFilterLoess2D** &copy)
- 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** &copy)
- 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** &copy)
- 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 brathl::CBratAlgoFilterLoess2D::CBratAlgoFilterLoess2D ( )

Default contructor

8.12.2.2 brathl::CBratAlgoFilterLoess2D::CBratAlgoFilterLoess2D (const CBratAlgoFilterLoess2D & *copy*)

Copy contructor

8.12.2.3 brathl::CBratAlgoFilterLoess2D::~CBratAlgoFilterLoess2D ( ) [virtual]

Destructor

8.12.3 Member Function Documentation

8.12.3.1 void brathl::CBratAlgoFilterLoess2D::Dump (std::ostream & *fOut* = std::cerr) [override], [virtual]

Dump function

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

8.12.3.2 virtual std::string brathl::CBratAlgoFilterLoess2D::GetDescription ( ) const [inline], [override], [virtual]

Gets the description of the algorithm

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

8.12.3.3 virtual std::string brathl::CBratAlgoFilterLoess2D::GetInputParamDesc (uint32_t *indexParam*) const [inline], [override], [virtual]

Gets the description of an input parameter.

Parameters

indexParam	[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().

8.12.3.4 virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::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 CBratAlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

indexParam	[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().

8.12.3.5 virtual std::string brathl::CBratAlgoFilterLoess2D::GetInputParamUnit (uint32_t *indexParam*) const [inline], [override], [virtual]

Gets the unit of an input parameter :

Parameters

indexParam	[in] : parameter index.

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

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

8.12.3.6 virtual std::string brathl::CBratAlgoFilterLoess2D::GetName ( ) const [inline], [override], [virtual]

Gets the name of the algorithm

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

8.12.3.7 virtual uint32_t brathl::CBratAlgoFilterLoess2D::GetNumInputParam ( ) const [inline], [override], [virtual]

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

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

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.12.3.8 virtual std::string brathl::CBratAlgoFilterLoess2D::GetOutputUnit ( ) const [inline], [override], [virtual]

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

Parameters

indexParam	[in] : parameter index.

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

8.12.3.9 CBratAlgoFilterLoess2D & brathl::CBratAlgoFilterLoess2D::operator= (const CBratAlgoFilterLoess2D & *copy*)

Overloads operator '='

8.12.3.10 double brathl::CBratAlgoFilterLoess2D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[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:

- BratAlgoFilterLoess2D.h
- BratAlgoFilterLoess2D.cpp

8.13 brathl::CBratAlgoFilterMedian1D Class Reference

`#include <BratAlgoFilterMedian1D.h>`

Inherits brathl::CBratAlgoFilterMedian.

Public Member Functions

- **CBratAlgoFilterMedian1D** ()
- **CBratAlgoFilterMedian1D** (const **CBratAlgoFilterMedian1D** &copy)
- 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** &copy)
- 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** &copy)
- 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 contructor

8.13.2.2 brathl::CBratAlgoFilterMedian1D::CBratAlgoFilterMedian1D (const **CBratAlgoFilterMedian1D** & *copy*)

Copy contructor

8.13.2.3 virtual brathl::CBratAlgoFilterMedian1D::~CBratAlgoFilterMedian1D ( ) [inline], [virtual]

Destructor

8.13.3 Member Function Documentation

8.13.3.1 void brathl::CBratAlgoFilterMedian1D::Dump (std::ostream & *fOut* = std::cerr) [override], [virtual]

Dump function

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

8.13.3.2 virtual std::string brathl::CBratAlgoFilterMedian1D::GetDescription ( ) const [inline], [override], [virtual]

Gets the description of the algorithm

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

8.13.3.3 virtual std::string brathl::CBratAlgoFilterMedian1D::GetInputParamDesc (uint32_t *indexParam*) const [inline], [override], [virtual]

Gets the description of an input parameter.

Parameters

indexParam	[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().

8.13.3.4 virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::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 CBrat↩ AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

indexParam	[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().

8.13.3.5 virtual std::string brathl::CBratAlgoFilterMedian1D::GetInputParamUnit (uint32_t *indexParam*) const [inline], [override], [virtual]

Gets the unit of an input parameter :

Parameters

indexParam	[in] : parameter index.

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

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

8.13.3.6 virtual std::string brathl::CBratAlgoFilterMedian1D::GetName ( ) const [inline], [override], [virtual]

Gets the name of the algorithm

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

Referenced by Run().

8.13.3.7 virtual uint32_t brathl::CBratAlgoFilterMedian1D::GetNumInputParam ( ) const [inline], [override], [virtual]

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

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

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.13.3.8 virtual std::string brathl::CBratAlgoFilterMedian1D::GetOutputUnit ( ) const [inline], [override], [virtual]

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

Parameters

indexParam	[in] : parameter index.

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

8.13.3.9 CBratAlgoFilterMedian1D & brathl::CBratAlgoFilterMedian1D::operator= (const CBratAlgoFilterMedian1D & *copy*)

Overloads operator '='

8.13.3.10 double brathl::CBratAlgoFilterMedian1D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

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

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

- BratAlgoFilterMedian1D.h
- BratAlgoFilterMedian1D.cpp

8.14 brathl::CBratAlgoFilterMedian2D Class Reference

`#include <BratAlgoFilterMedian2D.h>`

Inherits brathl::CBratAlgoFilterMedian.

Public Member Functions

- **CBratAlgoFilterMedian2D** ()
- **CBratAlgoFilterMedian2D** (const **CBratAlgoFilterMedian2D** &copy)

- 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** &copy)
- 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** &copy)
- 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 contructor

8.14.2.2 brathl::CBratAlgoFilterMedian2D::CBratAlgoFilterMedian2D (const **CBratAlgoFilterMedian2D** & *copy*)

Copy contructor

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

indexParam	[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 CBrat↩ AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

indexParam	[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

indexParam	[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 **brathl::CBratAlgorithmBase** (p. 131).

8.14.3.7 virtual uint32_t brathl::CBratAlgoFilterMedian2D::GetNumInputParam ( ) const [inline], [override], [virtual]

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

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

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.14.3.8 virtual std::string brathl::CBratAlgoFilterMedian2D::GetOutputUnit ( ) const [inline], [override], [virtual]

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

Parameters

indexParam	[in] : parameter index.

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

8.14.3.9 CBratAlgoFilterMedian2D & brathl::CBratAlgoFilterMedian2D::operator= (const CBratAlgoFilterMedian2D & *copy*)

Overloads operator '='

8.14.3.10 double brathl::CBratAlgoFilterMedian2D::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

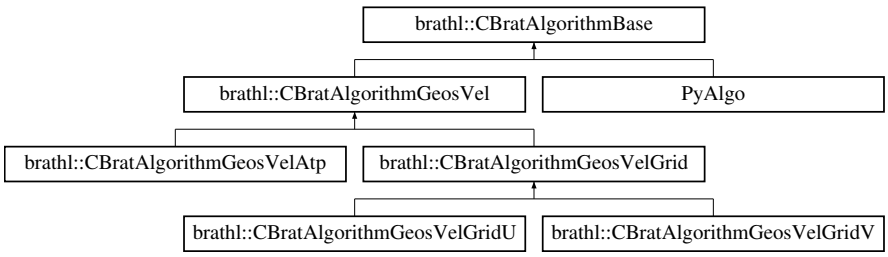
The documentation for this class was generated from the following files:

- BratAlgoFilterMedian2D.h
- BratAlgoFilterMedian2D.cpp

8.15 brathl::CBratAlgorithmBase Class Reference

`#include <BratAlgorithmBase.h>`

Inheritance diagram for brathl::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, CVectorBrat↩ AlgorithmParam &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 *algorithName)
- 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 brathl::CBratAlgorithmBase::CBratAlgorithmBase ( )

Default contructor

8.15.2.2 brathl::CBratAlgorithmBase::CBratAlgorithmBase (const CBratAlgorithmBase & *o*)

Copy contructor

8.15.2.3 brathl::CBratAlgorithmBase::~CBratAlgorithmBase ( ) [virtual]

Destructor

8.15.3 Member Function Documentation

8.15.3.1 void brathl::CBratAlgorithmBase::Dump (std::ostream & *fOut* = std::cerr) [virtual]

Dump function

Reimplemented in **brathl::CBratAlgorithmGeosVelGridV** (p. 15), **brathl::CBratAlgorithmGeosVelGridU** (p. 15), **brathl::CBratAlgoFilterLoess1D** (p. 115), **brathl::CBratAlgoFilterLoess2D** (p. 118), **brathl::CBratAlgoFilterMedian2D** (p. 125), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgorithmGeosVelGrid**

(p. 15), **brathl::CBratAlgorithmGeosVelAtp** (p. 135), **brathl::CBratAlgoFilterGaussian2D** (p. 108), **brathl::CBratAlgoFilterLanczos2D** (p. 112), **brathl::CBratAlgoFilterGaussian1D** (p. 106), **brathl::CBratAlgoFilterLanczos1D** (p. 111), and **brathl::CBratAlgorithmGeosVel** (p. 133).

References brathl::CObArray::Dump(), brathl::CObMap::Dump(), GetDescription(), GetInputParamDesc(), GetInputParamFormat(), GetInputParamUnit(), GetName(), GetNumInputParam(), and GetOutputUnit().

Referenced by brathl::CBratAlgorithmGeosVel::Dump().

8.15.3.2 virtual std::string brathl::CBratAlgorithmBase::GetDescription ( ) const [pure virtual]

Gets the description of the algorithm

Implemented in **PyAlgo** (p. 311), **brathl::CBratAlgorithmGeosVelGridV** (p. 15), **brathl::CBratAlgorithmGeosVelGridU** (p. 15), **brathl::CBratAlgoFilterGaussian1D** (p. 107), **brathl::CBratAlgoFilterGaussian2D** (p. 108), **brathl::CBratAlgoFilterLanczos1D** (p. 111), **brathl::CBratAlgoFilterLanczos2D** (p. 113), **brathl::CBratAlgoFilterLoess1D** (p. 115), **brathl::CBratAlgoFilterLoess2D** (p. 118), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgoFilterMedian2D** (p. 125), and **brathl::CBratAlgorithmGeosVelAtp** (p. 135).

Referenced by Dump().

8.15.3.3 virtual std::string brathl::CBratAlgorithmBase::GetInputParamDesc (uint32_t *indexParam*) const [pure virtual]

Gets the description of an input parameter.

Parameters

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

Implemented in **PyAlgo** (p. 312), **brathl::CBratAlgoFilterLoess1D** (p. 115), **brathl::CBratAlgoFilterLoess2D** (p. 118), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgoFilterMedian2D** (p. 125), **brathl::CBratAlgorithmGeosVelAtp** (p. 135), and **brathl::CBratAlgorithmGeosVelGrid** (p. 15).

Referenced by Dump().

8.15.3.4 virtual CBratAlgorithmParam::bratAlgoParamTypeVal brathl::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

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

Implemented in **PyAlgo** (p. 313), **brathl::CBratAlgoFilterLoess1D** (p. 115), **brathl::CBratAlgoFilterLoess2D** (p. 118), **brathl::CBratAlgoFilterMedian2D** (p. 125), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgorithmGeosVelAtp** (p. 136), and **brathl::CBratAlgorithmGeosVelGrid** (p. 15).

Referenced by Dump().

8.15.3.5 virtual std::string brathl::CBratAlgorithmBase::GetInputParamUnit (uint32_t *indexParam*) const [pure virtual]

Gets the unit of an input parameter :

Parameters

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

Implemented in **PyAlgo** (p. 313), **brathl::CBratAlgoFilterLoess1D** (p. 115), **brathl::CBratAlgoFilterLoess2D** (p. 119), **brathl::CBratAlgoFilterMedian2D** (p. 125), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgorithmGeosVelAtp** (p. 136), and **brathl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by Dump().

8.15.3.6 virtual std::string brathl::CBratAlgorithmBase::GetName ( ) const [pure virtual]

Gets the name of the algorithm

Implemented in **PyAlgo** (p. 313), **brathl::CBratAlgorithmGeosVelGridV** (p. 16), **brathl::CBratAlgorithmGeos↩VelGridU** (p. 16), **brathl::CBratAlgoFilterGaussian1D** (p. 107), **brathl::CBratAlgoFilterGaussian2D** (p. 108), **brathl::CBratAlgoFilterLanczos1D** (p. 111), **brathl::CBratAlgoFilterLanczos2D** (p. 113), **brathl::CBratAlgo↩FilterLoess1D** (p. 116), **brathl::CBratAlgoFilterLoess2D** (p. 120), **brathl::CBratAlgoFilterMedian1D** (p. 122), **brathl::CBratAlgoFilterMedian2D** (p. 125), and **brathl::CBratAlgorithmGeosVelAtp** (p. 136).

Referenced by Dump().

8.15.3.7 virtual uint32_t brathl::CBratAlgorithmBase::GetNumInputParam ( ) const [pure virtual]

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

Implemented in **PyAlgo** (p. 313), **brathl::CBratAlgoFilterLoess1D** (p. 116), **brathl::CBratAlgoFilterLoess2↩D** (p. 120), **brathl::CBratAlgoFilterMedian1D** (p. 123), **brathl::CBratAlgoFilterMedian2D** (p. 126), **brathl::C↩BratAlgorithmGeosVelAtp** (p. 136), and **brathl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by Dump().

8.15.3.8 virtual std::string brathl::CBratAlgorithmBase::GetOutputUnit ( ) const [pure virtual]

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

Implemented in **PyAlgo** (p. 313), **brathl::CBratAlgoFilterLoess1D** (p. 116), **brathl::CBratAlgoFilterLoess2↩D** (p. 120), **brathl::CBratAlgoFilterMedian2D** (p. 126), **brathl::CBratAlgoFilterMedian1D** (p. 123), **brathl::C↩BratAlgorithmGeosVelAtp** (p. 136), and **brathl::CBratAlgorithmGeosVelGrid** (p. 16).

Referenced by Dump().

8.15.3.9 CBratAlgorithmBase & brathl::CBratAlgorithmBase::operator= (const CBratAlgorithmBase & *o*)

Overloads operator '='

8.15.3.10 virtual double brathl::CBratAlgorithmBase::Run (CVectorBratAlgorithmParam & *args*) [pure virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

Implemented in **PyAlgo** (p. 314), **brathl::CBratAlgoFilterLoess1D** (p. 116), **brathl::CBratAlgoFilterLoess2↩D** (p. 120), **brathl::CBratAlgoFilterMedian2D** (p. 126), **brathl::CBratAlgoFilterMedian1D** (p. 123), **brathl↩::CBratAlgorithmGeosVelAtp** (p. 136), **brathl::CBratAlgorithmGeosVelGrid** (p. 16), **brathl::CBratAlgo↩FilterGaussian1D** (p. 107), **brathl::CBratAlgoFilterGaussian2D** (p. 109), **brathl::CBratAlgoFilterLanczos1D** (p. 111), and **brathl::CBratAlgoFilterLanczos2D** (p. 113).

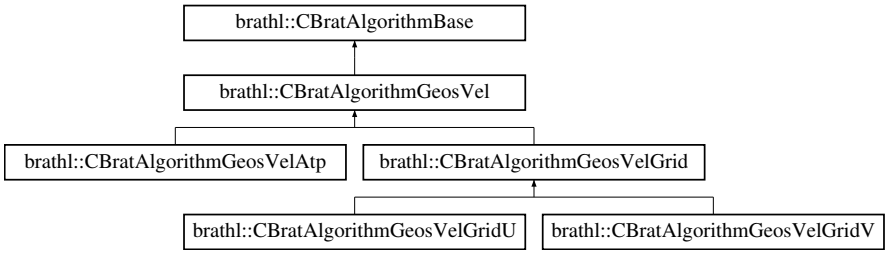
The documentation for this class was generated from the following files:

- BratAlgorithmBase.h
- BratAlgorithmBase.cpp

8.16 brathl::CBratAlgorithmGeosVel Class Reference

`#include <BratAlgorithmGeosVel.h>`

Inheritance diagram for brathl::CBratAlgorithmGeosVel:



Public Member Functions

- void **BtoE** (double lonPlane, double latPlane, double betaX, double betaY, double &lon, double &lat)
- **CBratAlgorithmGeosVel** ()
- **CBratAlgorithmGeosVel** (const **CBratAlgorithmGeosVel** &copy)
- 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** &copy)
- 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_degreeToRadianMutiplier**
- 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 brathl::CBratAlgorithmGeosVel::CBratAlgorithmGeosVel ( )

Default contructor

8.16.2.2 brathl::CBratAlgorithmGeosVel::CBratAlgorithmGeosVel (const **CBratAlgorithmGeosVel** & *copy*)

Copy contructor

8.16.2.3 brathl::CBratAlgorithmGeosVel::~CBratAlgorithmGeosVel ( ) [virtual]

Destructor

8.16.3 Member Function Documentation

8.16.3.1 void brathl::CBratAlgorithmGeosVel::Dump (std::ostream & *fOut* = `std::cerr`) [virtual]

Dump function

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

Reimplemented in **brathl::CBratAlgorithmGeosVelGridV** (p. 15), **brathl::CBratAlgorithmGeosVelGridU** (p. 15), **brathl::CBratAlgorithmGeosVelGrid** (p. 15), and **brathl::CBratAlgorithmGeosVelAtp** (p. 135).

References brathl::CBratAlgorithmBase::Dump().

Referenced by brathl::CBratAlgorithmGeosVelAtp::Dump(), and brathl::CBratAlgorithmGeosVelGrid::Dump().

8.16.3.2 **CBratAlgorithmGeosVel** & brathl::CBratAlgorithmGeosVel::operator= (const **CBratAlgorithmGeosVel** & *copy*)

Overloads operator '='

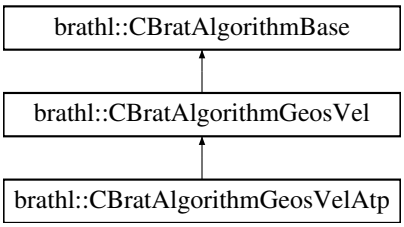
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVel.h
- BratAlgorithmGeosVel.cpp

8.17 brathl::CBratAlgorithmGeosVelAtp Class Reference

`#include <BratAlgorithmGeosVelAtp.h>`

Inheritance diagram for brathl::CBratAlgorithmGeosVelAtp:



Public Member Functions

- **CBratAlgorithmGeosVelAtp** ()
- **CBratAlgorithmGeosVelAtp** (const **CBratAlgorithmGeosVelAtp** &copy)
- 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** &copy)
- 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** &copy)
- 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 contructor

8.17.2.2 brathl::CBratAlgorithmGeosVelAtp::CBratAlgorithmGeosVelAtp (const **CBratAlgorithmGeosVelAtp** & *copy*)

Copy contructor

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

indexParam	[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 brathl::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 CBrat↩ AlgorithmParam::T_STRING for std::string CBratAlgorithmParam::T_CHAR for a character

Parameters

indexParam	[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().

8.17.3.5 virtual std::string brathl::CBratAlgorithmGeosVelAtp::GetInputParamUnit (uint32_t *indexParam*) const [inline], [override], [virtual]

Gets the unit of an input parameter :

Parameters

indexParam	[in] : parameter index.

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

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

8.17.3.6 virtual std::string brathl::CBratAlgorithmGeosVelAtp::GetName ( ) const [inline], [override], [virtual]

Gets the name of the algorithm

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

8.17.3.7 virtual uint32_t brathl::CBratAlgorithmGeosVelAtp::GetNumInputParam ( ) const [inline], [override], [virtual]

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

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

Referenced by GetInputParamDesc(), GetInputParamFormat(), and GetInputParamUnit().

8.17.3.8 virtual std::string brathl::CBratAlgorithmGeosVelAtp::GetOutputUnit ( ) const [inline], [override], [virtual]

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

Parameters

indexParam	[in] : parameter index.

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

8.17.3.9 CBratAlgorithmGeosVelAtp & brathl::CBratAlgorithmGeosVelAtp::operator= (const CBratAlgorithmGeosVelAtp & *copy*)

Overloads operator '='

8.17.3.10 double brathl::CBratAlgorithmGeosVelAtp::Run (CVectorBratAlgorithmParam & *args*) [override], [virtual]

Runs the algorithm

Parameters

fmt	[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
args	[in] : the values of input parameters i(a C/C++ va_list).

Returns

the result of the execution

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

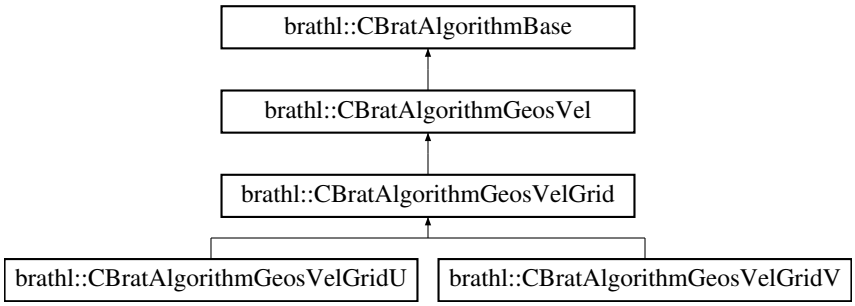
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelAtp.h
- BratAlgorithmGeosVelAtp.cpp

8.18 brathl::CBratAlgorithmGeosVelGrid Class Reference

`#include <BratAlgorithmGeosVelGrid.h>`

Inheritance diagram for brathl::CBratAlgorithmGeosVelGrid:



Public Member Functions

- **CBratAlgorithmGeosVelGrid** ()
- **CBratAlgorithmGeosVelGrid** (const **CBratAlgorithmGeosVelGrid** &copy)
- 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** &copy)
- 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** &copy)
- 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.

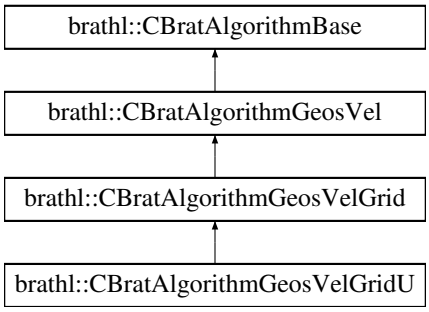
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.19 brathl::CBratAlgorithmGeosVelGridU Class Reference

`#include <BratAlgorithmGeosVelGrid.h>`

Inheritance diagram for brathl::CBratAlgorithmGeosVelGridU:



Public Member Functions

- **CBratAlgorithmGeosVelGridU** ()
- **CBratAlgorithmGeosVelGridU** (const **CBratAlgorithmGeosVelGridU** &copy)
- 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.

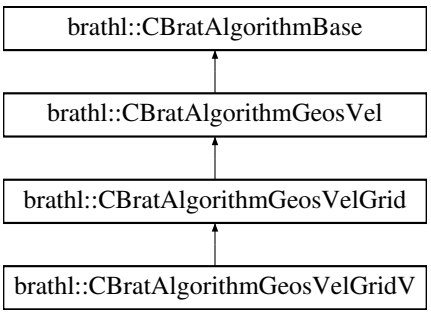
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.20 brathl::CBratAlgorithmGeosVelGridV Class Reference

`#include <BratAlgorithmGeosVelGrid.h>`

Inheritance diagram for brathl::CBratAlgorithmGeosVelGridV:



Public Member Functions

- **CBratAlgorithmGeosVelGridV** ()
- **CBratAlgorithmGeosVelGridV** (const **CBratAlgorithmGeosVelGridV** &copy)
- 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.

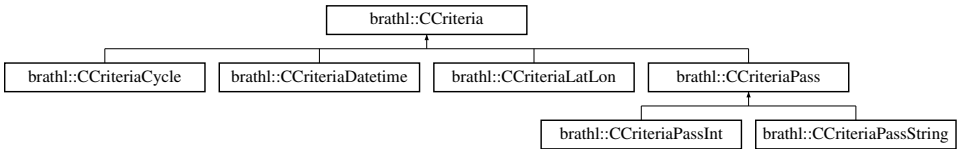
The documentation for this class was generated from the following files:

- BratAlgorithmGeosVelGrid.h
- BratAlgorithmGeosVelGrid.cpp

8.21 brathl::CCriteria Class Reference

`#include <Criteria.h>`

Inheritance diagram for brathl::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 fonction.
- 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 brathl::CCriteria::IsDefaultValue ( ) [pure virtual]

Tests whether value have been initialized or not

Returns

true if not initialized

Implemented in **brathl::CCriteriaPassInt** (p. 61), **brathl::CCriteriaLatLon** (p. 158), **brathl::CCriteriaDatetime** (p. 150), **brathl::CCriteriaCycle** (p. 145), **brathl::CCriteriaPassString** (p. 61), and **brathl::CCriteriaPass** (p. 60).

8.21.2.2 virtual void brathl::CCriteria::SetDefaultValue ( ) [pure virtual]

Sets internal value to the default value (uninitialized)

Implemented in **brathl::CCriteriaPassInt** (p. 62), **brathl::CCriteriaLatLon** (p. 160), **brathl::CCriteriaDatetime** (p. 151), **brathl::CCriteriaCycle** (p. 146), **brathl::CCriteriaPassString** (p. 62), and **brathl::CCriteriaPass** (p. 62).

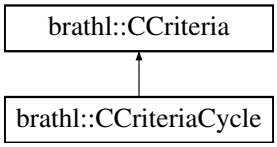
The documentation for this class was generated from the following files:

- Criteria.h
- Criteria.cpp

8.22 brathl::CCriteriaCycle Class Reference

`#include <CriteriaCycle.h>`

Inheritance diagram for brathl::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

from	start pass

to	end pass

8.22.2.2 brathl::CCriteriaCycle::CCriteriaCycle (const std::string & *from,* const std::string & *to*)

Constructor.

Parameters

from	start pass
to	end pass

8.22.2.3 brathl::CCriteriaCycle::CCriteriaCycle (const CStringArray & *array*)

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

Parameters

array	start and end dates

8.22.3 Member Function Documentation

8.22.3.1 bool brathl::CCriteriaCycle::Intersect (CStringArray & *array,* CStringArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

array	that contains start pass as std::string, end pass as std::string
intersect	intersection period

Returns

true, or false if there is no intersection

Referenced by Intersect().

8.22.3.2 bool brathl::CCriteriaCycle::Intersect (CStringArray & *array,* CIntArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

array	that contains start pass as std::string, end pass as std::string
intersect	intersection period

Returns

true, or false if there is no intersection

References Intersect().

8.22.3.3 bool brathl::CCriteriaCycle::Intersect (CIntArray & *array,* CStringArray & *intersect*)

Create the intersection of this date period with the given one

Parameters

array	that contains start pass as std::string, end pass as std::string
intersect	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

array	that contains start pass as std::string, end pass as std::string
intersect	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

from	start pass
to	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

from	start pass
to	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

array	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

to	start pass

References m_from.

Referenced by Set().

8.22.3.11 void brathl::CCriteriaCycle::SetFrom (const std::string & *from*)

Sets start pass

Parameters

to	start pass

References m_from, and brathl::CTools::StrToInt32().

8.22.3.12 void brathl::CCriteriaCycle::SetTo (int32_t *to*)

Sets end pass

Parameters

to	end pass

References m_to.

Referenced by Set().

8.22.3.13 void brathl::CCriteriaCycle::SetTo (const std::string & *to*)

Sets end pass

Parameters

to	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 brathl::CCriteriaCycle::m_to [protected]

end pass

Referenced by Dump(), IsDefaultValue(), SetDefaultValue(), and SetTo().

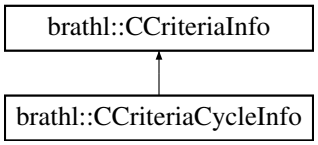
The documentation for this class was generated from the following files:

- CriteriaCycle.h
- CriteriaCycle.cpp

8.23 brathl::CCriteriaCycleInfo Class Reference

`#include <CriteriaInfo.h>`

Inheritance diagram for brathl::CCriteriaCycleInfo:



Public Member Functions

- **CCriteriaCycleInfo** ()

  *Empty **CCriteriaCycleInfo** (p. 147) ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr)

  Dump fonction.
- 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

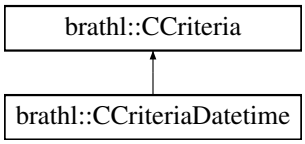
The documentation for this class was generated from the following files:

- 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=CDatePeriod::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

datePeriod	period to set

References Set().

8.24.2.2 brathl::CCriteriaDatetime::CCriteriaDatetime (CDate & *from*, CDate & *to*)

Constructor.

Parameters

from	start date
to	end date

References Set().

8.24.2.3 brathl::CCriteriaDatetime::CCriteriaDatetime (const std::string & *from*, const std::string & *to*)

Constructor.

Parameters

from	start date
to	end date

References Set().

8.24.2.4 brathl::CCriteriaDatetime::CCriteriaDatetime (double *from,* double *to*)

Constructor.

Parameters

from	start date (number of seconds since 1950-01-01)
to	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

array	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

datePeriod	intersect with this
intersect	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

otherFrom	start date intersect with this
otherTo	end date intersect with this
intersect	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 **brathl::CCriteria** (p. 141).

References brathl::CDatePeriod::IsDefaultValue(), and m_datePeriod.

8.24.3.4 void brathl::CCriteriaDatetime::Set (CDatePeriod & *datePeriod*)

Sets date period from another one

Parameters

datePeriod	period to set

References m_datePeriod, and brathl::CDatePeriod::Set().

Referenced by CCriteriaDatetime().

8.24.3.5 void brathl::CCriteriaDatetime::Set (CDate & *from,* CDate & *to*)

Sets date period from start and end date

Parameters

from	start date
to	end date

References m_datePeriod, and brathl::CDatePeriod::Set().

8.24.3.6 void brathl::CCriteriaDatetime::Set (const std::string & *from,* const std::string & *to*)

Sets date period from start and end date

Parameters

from	start date
to	end date

References m_datePeriod, and brathl::CDatePeriod::Set().

8.24.3.7 void brathl::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

array	start and end dates

References m_datePeriod, and brathl::CDatePeriod::Set().

8.24.3.8 void brathl::CCriteriaDatetime::SetDefaultValue ( ) [virtual]

Sets internal value to the default value (uninitialized)

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

References m_datePeriod, and brathl::CDatePeriod::SetDefaultValue().

8.24.3.9 void brathl::CCriteriaDatetime::SetFrom (CDate & *from*)

Sets start date

Parameters

to	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

to	start date

References m_datePeriod, and brathl::CDatePeriod::SetFrom().

8.24.3.11 void brathl::CCriteriaDatetime::SetTo (CDate & *to*)

Sets end date

Parameters

to	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

to	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().

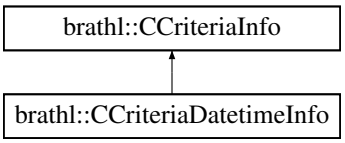
The documentation for this class was generated from the following files:

- CriteriaDatetime.h
- CriteriaDatetime.cpp

8.25 brathl::CCriteriaDatetimeInfo Class Reference

`#include <CriteriaInfo.h>`

Inheritance diagram for brathl::CCriteriaDatetimeInfo:



Public Member Functions

- **CCriteriaDatetimeInfo** ()

  *Empty **CCriteriaDatetimeInfo*** (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 ~**CCriteriaDatetimeInfo** ()

  Destructor.

Static Public Member Functions

- static **CCriteriaDatetimeInfo** * **GetCriteriaInfo** (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

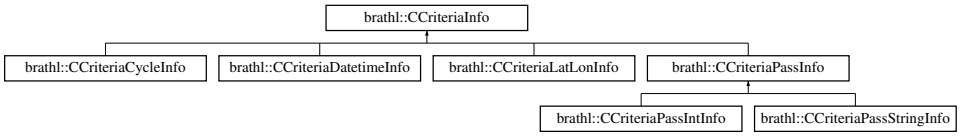
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.26 brathl::CCriteriaInfo Class Reference

`#include <CriteriaInfo.h>`

Inheritance diagram for brathl::CCriteriaInfo:



Public Member Functions

- **CCriteriaInfo** ()

  *Empty **CCriteriaInfo** (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 ~**CCriteriaInfo** ()

  Destructor.

Static Public Member Functions

- static **CCriteriaInfo** * **GetCriteriaInfo** (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

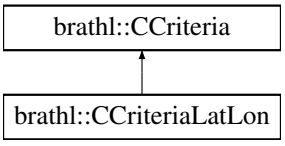
The documentation for this class was generated from the following files:

- CriteriaInfo.h
- CriteriaInfo.cpp

8.27 brathl::CCriteriaLatLon Class Reference

`#include <CriteriaLatLon.h>`

Inheritance diagram for brathl::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 brathl::CCriteriaLatLon::CCriteriaLatLon (CLatLonRect & *latLonRect*)

Constructor.

Parameters

latLonRect	lat/lon bounding box

References Set().

8.27.2.2 brathl::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

p1	one corner of the box
deltaLat	delta lat from p1. (may be positive or negetive)
deltaLon	delta lon from p1. (may be positive or negetive)

References Set().

8.27.2.3 brathl::CCriteriaLatLon::CCriteriaLatLon (CLatLonPoint & *latLonLow,* CLatLonPoint & *latLonHigh*)

Constructor.

Parameters

latLonLow	lat/lon low point
latLonHigh	lat/lon high point

References Set().

8.27.2.4 brathl::CCriteriaLatLon::CCriteriaLatLon (double *latLow,* double *lonLow,* double *latHigh,* double *lonHigh*)

Constructor.

Parameters

latLow	latitude low
lonLow	longitude low
latHigh	latitude high
lonHigh	longitude high

References Set().

8.27.2.5 brathl::CCriteriaLatLon::CCriteriaLatLon (const std::string & *latLow,* const std::string & *lonLow,* const std::string & *latHigh,* const std::string & *lonHigh*)

Constructor.

Parameters

latLow	latitude low
lonLow	longitude low
latHigh	latitude high
lonHigh	longitude high

References Set().

8.27.2.6 brathl::CCriteriaLatLon::CCriteriaLatLon (const CStringArray & *array*)

Constructor from a list that contains low latitude value, low longitude value, high latitude value, high longitude value.

Parameters

array	to be converted

References m_latLonRect.

8.27.2.7 brathl::CCriteriaLatLon::~CCriteriaLatLon ( ) [virtual]

Destructor.

Getter of the property t<tl>atLonRect/<tt.>

Returns

Returns the latLonRect.

8.27.3 Member Function Documentation

8.27.3.1 double brathl::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 brathl::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 brathl::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 brathl::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 brathl::CCriteriaLatLon::GetMinOrMaxLon (double *lon1,* double *lon2,* bool *wantMin*) [static]

Gets the min. or max. of two longitudes.

Parameters

lon1	first longitude
lon2	second longitude
wantMin	true: returns min., false: returns max.

Returns

min. lon or max. lon, depends on wantMin.

References brathl::CTools::Max(), and brathl::CTools::Min().

8.27.3.6 double brathl::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 brathl::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 brathl::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 brathl::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 brathl::CCriteriaLatLon::Intersect (CLatLonRect & *clip*, CLatLonRect & *intersect*)

Create the intersection of this LatLon Criteria with the given one

Parameters

clip	intersect with this
intersection	

Returns

true, or false if there is no intersection

References m_latLonRect.

8.27.3.11 bool brathl::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 t<tl>atLonRect/<tt.>

Parameters

latLonRect	The latLonRect 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

p1	one corner of the box
deltaLat	delta lat from p1. (may be positive or negetive)
deltaLon	delta lon from p1. (may be positive or negetive)

References m_latLonRect.

8.27.3.14 void brathl::CCriteriaLatLon::Set (CLatLonPoint & *latLonLow,* CLatLonPoint & *latLonHigh*)

Setter of the property t<tl>atLonRect/<tt.>

Parameters

latLonLow	lat/lon low point
latLonHigh	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 t<tl>atLonRect/<tt.>

Parameters

latLow	latitude low
lonLow	longitude low
latHigh	latitude high
lonHigh	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 t<tl>atLonRect/<tt.>

Parameters

latLow	latitude low
lonLow	longitude low
latHigh	latitude high
lonHigh	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<tl>atLonRect/<tt.>

Parameters

latLonRect	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(), GetLower↩ RightLon(), GetUpperLeftLat(), GetUpperLeftLon(), GetUpperRightLat(), GetUpperRightLon(), Intersect(), Is↩ DefaultValue(), Set(), and SetDefaultValue().

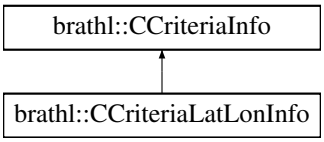
The documentation for this class was generated from the following files:

- 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

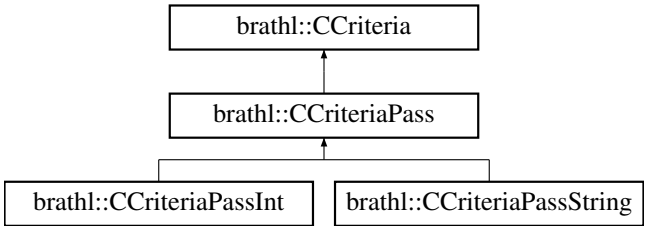
The documentation for this class was generated from the following files:

- 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 fonction.
- 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

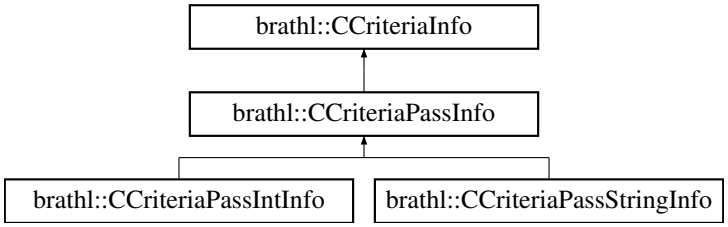
The documentation for this class was generated from the following files:

- 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

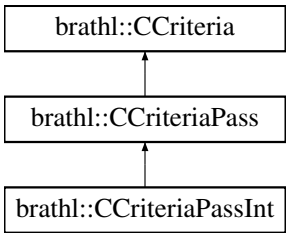
The documentation for this class was generated from the following files:

- 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 fonction.
- 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

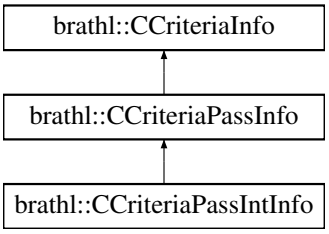
The documentation for this class was generated from the following files:

- 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

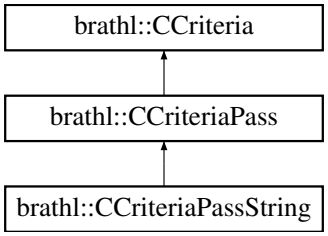
The documentation for this class was generated from the following files:

- 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=C↩ CriteriaPassString::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

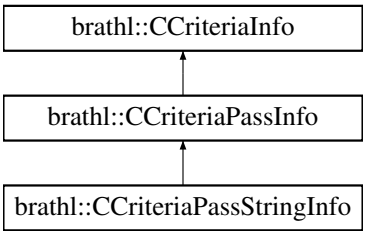
The documentation for this class was generated from the following files:

- CriteriaPass.h
- CriteriaPass.cpp

8.34 brathl::CCriteriaPassStringInfo Class Reference

`#include <CriteriaInfo.h>`

Inheritance diagram for brathl::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

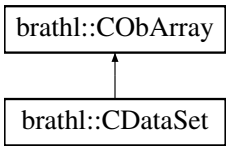
The documentation for this class was generated from the following files:

- 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)
- **CFieldSetArrayDbl** * **GetFieldSetAsArrayDbl** (const std::string &fieldSetKey)
- **CFieldSetDbl** * **GetFieldSetAsDbl** (const std::string &fieldSetKey)
- double **GetFieldSetAsDblValue** (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

fieldSetKey	[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

fieldSetKey	[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

fieldSetKey	[in] : fieldset key
fieldSet	[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** ref↩ Date=**REF19500101**) const
- int32_t **Convert2DSM** (double &days, double &seconds, double &muSeconds, const **brathl_refDate** ref↩ Date=**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_ref**↩ **Date** 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

strDate	: Allowed format are : • YYYY-MM-DD HH:MN: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

dateSeconds	[in]: decimal number of seconds
refDate	[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

d	[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

days	[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

Format	[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)

withMuSecond	[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

refDate	[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

julian	[out]: decimal julian day (can be < 0)
refDate	[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

days	[out]: number of days (can be < 0)
milliSeconds	[out]: number of milliseconds
muSeconds	[out]: number of microseconds
refDate	[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

days	[out]: number of days (can be < 0)
milliSeconds	[out]: number of milliseconds
muSeconds	[out]: number of microseconds
refDate	[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

days	[out]: number of days (can be < 0)
seconds	[out]: number of seconds
muSeconds	[out]: number of microseconds
refDate	[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

days	[out]: number of days (can be < 0)
seconds	[out]: number of seconds
muSeconds	[out]: number of microseconds
refDate	[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

seconds	[out]: decimal number of seconds day (can be < 0)
refDate	[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

seconds	[out]: number of milliseconds (can be < 0)
muSeconds	[out]: number of microseconds
refDate	[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

seconds	[out]: number of milliseconds (can be < 0)
muSeconds	[out]: number of microseconds
refDate	[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

year	[out]: year
month	[out]: month
day	[out]: day
hour	[out]: hour
minute	[out]: minute
second	[out]: second
muSecond	[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:MN: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

strDate	: 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

year	[in]: year
month	[in]: month of year
day	[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

date	[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

date	[int] : **CDate** (p. 170) object whose value corresponds to the refDate parameter
refDate	[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

month	[in] : month
year	[in] : year
nbDaysIn↩ Month[out]	: 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

year	[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

year	[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

year	[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:MN: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

strDate	: 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

date	[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

date	[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

days	[in]: number of days
seconds	[in]: number of seconds
muSeconds	[in]: number of microseconds
refDate	[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

date	[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

date	[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

dateSeconds	[in]: decimal number of seconds
refDate	[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

dateJulian	[in]: decimal julian day
refDate	[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

days	[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 brathl::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 brathl::CDate::m_internalRefYear = 1950 [static]

Internal reference year (1950)

Referenced by Convert2YMDHMSM(), DayOfYear(), and HowManyLeapYear().

8.36.4.4 const double brathl::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 brathl::CDate::m_minutesInHour = 60.0 [static]

Number of minutes in an hour

Referenced by Convert2YMDHMSM().

8.36.4.6 const double brathl::CDate::m_secInDay = 86400.0 [static]

Number of seconds in a day

Referenced by Convert2DecimalJulian(), and Convert2DSM().

8.36.4.7 const double brathl::CDate::m_secInHour = 3600.0 [static]

Number of seconds in an hour

8.36.4.8 const double brathl::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 brathl::CDatePeriod Class Reference

`#include <DatePeriod.h>`

Inherits brathl::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 fonction.
- 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

datePeriod	period to set

References Set().

8.37.2.2 brathl::CDatePeriod::CDatePeriod (CDate & *from,* CDate & *to*)

Constructor.

Parameters

from	start date
to	end date

References SetFrom(), and SetTo().

8.37.2.3 brathl::CDatePeriod::CDatePeriod (const std::string & *from,* const std::string & *to*)

Constructor.

Parameters

from	start date
to	end date

References Set().

8.37.2.4 brathl::CDatePeriod::CDatePeriod (double *from,* double *to*)

Constructor.

Parameters

from	start date (number of seconds since 1950-01-01)
to	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

array	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

datePeriod	intersect with this
intersect	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

otherFrom	start date intersect with this
otherTo	end date intersect with this
intersect	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

from	start date
to	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

from	start date
to	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

array	start and end dates

References Set().

8.37.3.10 void brathl::CDatePeriod::Set (CDatePeriod & *datePeriod*)

Sets date period from another one

Parameters

datePeriod	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

to	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

to	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

to	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

to	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 duble 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 brathl::CExpressionValue Class Reference

`#include <Expression.h>`

Inherits brathl::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 brathl::CExternalFilesAvisoGrid Class Reference

`#include <ExternalFilesAvisoGrid.h>`

Inherits brathl::CExternalFilesNetCDFCF.

Inherited by brathl::CExternalFilesDotGrid, and brathl::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** *map↩ Attributes=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 brathl::CExternalFilesAvisoGrid::LoadStructure ( ) [protected], [virtual]

Array of the global dimension's index

Implements **brathl::CExternalFilesNetCDF** (p. 200).

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

- ExternalFilesAvisoGrid.h
- ExternalFilesAvisoGrid.cpp

8.43 brathl::CExternalFilesJason2 Class Reference

`#include <ExternalFilesJason2.h>`

Inherits brathl::CExternalFilesNetCDFCF.

Inherited by brathl::CExternalFilesJason2GDR, brathl::CExternalFilesJason2SGDR, and brathl::CExternalFiles↩ Jason2SSHA.

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 &wanted↩ Unit, 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 &wanted↩ Unit)
- 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 must↩ Exist=true, double defaultValue=**CTools::m_defaultValueDOUBLE**)
- int **GetAttribute** (const std::string &varName, const std::string &attName, std::string &attValue, bool must↩ Exist=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 default↩ Value=**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 &common↩ DimensionNames)
- 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** (**brathl_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** ∗map↩Attributes=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).

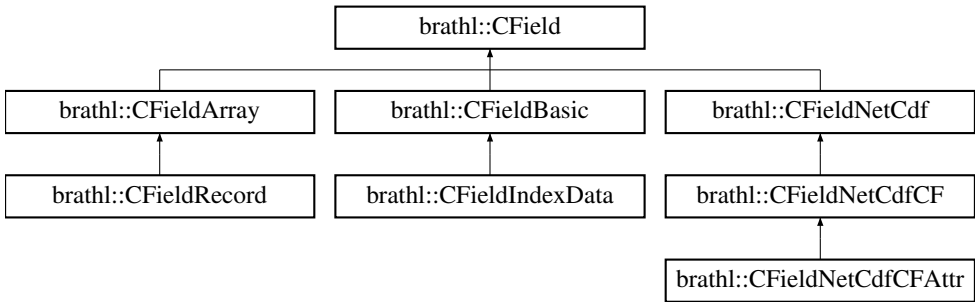
The documentation for this class was generated from the following files:

- 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 **IsFieldNetCdfCFAttr** ()
- 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 brathl::CField::m_dim[MAX_NUM_DIMS] [protected]

total number of dimensions

8.45.2.2 bool brathl::CField::m_isFixedSize [protected]

(maximum) dimensions

Referenced by Dump().

8.45.2.3 double brathl::CField::m_validMax [protected]

Valid max value

8.45.2.4 double brathl::CField::m_validMin [protected]

Valid min value

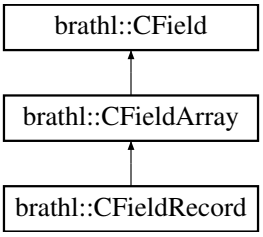
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.46 brathl::CFieldArray Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::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

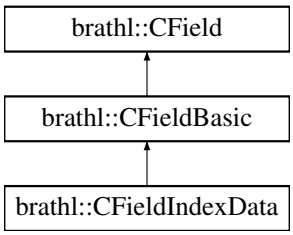
The documentation for this class was generated from the following files:

- 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

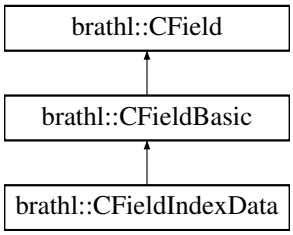
The documentation for this class was generated from the following files:

- 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

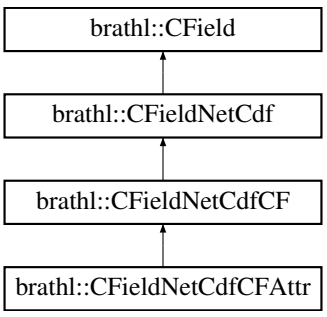
The documentation for this class was generated from the following files:

- 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 **CIntMap** & **GetDimIds** ()
- void **GetDimIdsAsArray** (CIntArray &values, bool bRemoveAll=true)
- const CStringArray & **GetDimNames** ()
- uint32_t **GetDimRange** (const std::string &dimName)
- const **CUIntMap** & **GetDimRanges** ()
- uint32_t * **GetDimsCountArray** ()
- uint32_t * **GetDimsIndexArray** ()
- const **CUIntMap** & **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 CStringArray 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 CStringMap 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().

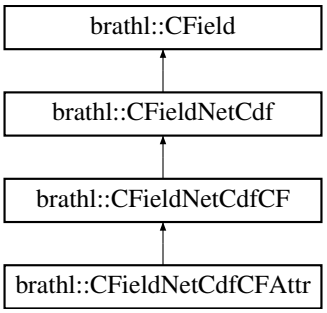
The documentation for this class was generated from the following files:

- 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 fonction.
- 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

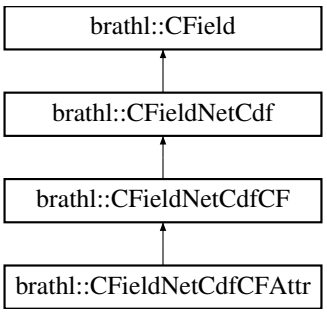
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.51 brathl::CFieldNetCdfCFAttr Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::CFieldNetCdfCFAttr:



Public Member Functions

- **CFieldNetCdfCFAttr** ()
  Ctor.
- **CFieldNetCdfCFAttr** (CNetCDFVarDef ∗netCDFVarDef, CNetCDFAttr ∗netCDFAttr)
- **CFieldNetCdfCFAttr** (CNetCDFAttr ∗netCDFAttr)
- **CFieldNetCdfCFAttr** (**CFieldNetCdfCFAttr** &f)
- virtual CBratObject ∗ **Clone** ()
- **CFieldNetCdfCFAttr** ∗ **CloneThis** ()
- void **DeleteNetCDFAttr** ()
- virtual void **Dump** (std::ostream &fOut=std::cerr)
  Dump fonction.
- virtual void **DumpFieldDictionary** (std::ostream &fOut=std::cout)
- virtual std::string **GetMostExplicitName** ()
- CNetCDFAttr ∗ **GetNetCDFAttr** ()
- const std::string & **GetRelatedVarName** ()
- bool **IsFieldNetCdfCFAttrGlobal** ()
- bool **IsFieldNetCdfCFAttrVariable** ()
- const **CFieldNetCdfCFAttr** & **operator=** (**CFieldNetCdfCFAttr** &f)
- void **Set** (**CFieldNetCdfCFAttr** &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 ∼**CFieldNetCdfCFAttr** ()
  Dtor.

Static Public Member Functions

- static bool **IsFieldNetCdfCFAttrGlobal** (CBratObject ∗ob)
- static bool **IsFieldNetCdfCFAttrVariable** (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

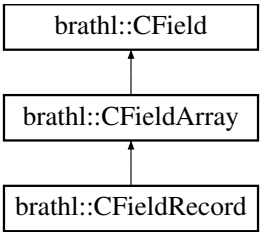
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.52 brathl::CFieldRecord Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::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 fonction.
- 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

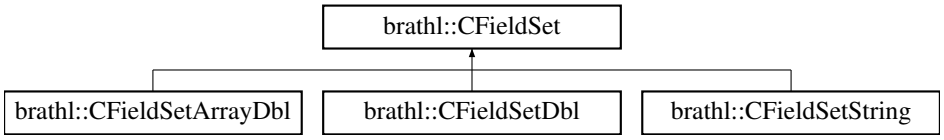
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.53 brathl::CFieldSet Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::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

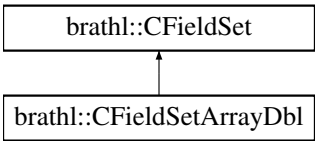
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.54 brathl::CFieldSetArrayDbl Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::CFieldSetArrayDbl:



Public Member Functions

- **CFieldSetArrayDbl** (const std::string &name="")

  Ctor.
- **CFieldSetArrayDbl** (**CFieldSetArrayDbl** &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)
- **CFieldSetArrayDbl** & **operator=** (**CFieldSetArrayDbl** &o)
- virtual ~**CFieldSetArrayDbl** ()

  Dtor.

Public Attributes

- CUIntArray **m_dim**
- int32_t **m_nbDims**
- CDoubleArray **m_vector**

Protected Member Functions

- void **Copy** (**CFieldSetArrayDbl** &f)

Additional Inherited Members

8.54.1 Detailed Description

a set of double array field value.

Version

1.0

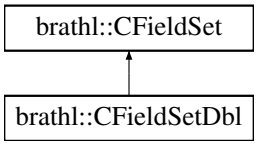
The documentation for this class was generated from the following files:

- Field.h
- Field.cpp

8.55 brathl::CFieldSetDbl Class Reference

`#include <Field.h>`

Inheritance diagram for brathl::CFieldSetDbl:



Public Member Functions

- int32_t **AsInt32** ()
- int32_t **AsUInt32** ()
- **CFieldSetDbl** (const std::string &name="")

  Ctor.
- **CFieldSetDbl** (**CFieldSetDbl** &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)
- **CFieldSetDbl** & **operator=** (**CFieldSetDbl** &o)
- void **SetData** (double value)
- virtual ~**CFieldSetDbl** ()

  Dtor.

Public Attributes

- double **m_value**

Protected Member Functions

- void **Copy** (**CFieldSetDbl** &f)

Additional Inherited Members

8.55.1 Detailed Description

a set of double field value.

Version

1.0

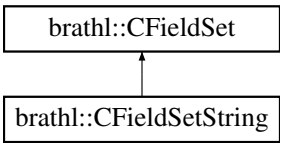
The documentation for this class was generated from the following files:

- 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 fonction.
- 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

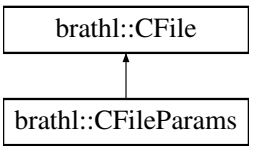
The documentation for this class was generated from the following files:

- 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

name	[in] : full name of the file;
mode	[in] : access mode - default value : modeRead|typeBinary (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

filename	[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

newFileName	[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(), ReadTo↩Buffer(), Rename(), SeekToBegin(), SeekToEnd(), SetBufferingMode(), SetPosition(), WriteChar(), WriteFrom↩Buffer(), 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

name	[in] : full name of the file;
mode	[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, modeRead↩Write, 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

lineRead	[out] : line data read
size	[in] : max number of bytes of the line

Returns

the number of bytes in the lineRead parameter. -1 if end of file reached

References brathl::CTools::Trim().

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

8.57.3.12 int32_t brathl::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

destinationBuffer	[out] : destination buffer
numBytesTo↩ Read	[in] : number of bytes to reads

Returns

the number of bytes actually reads, zero if end of file reached

References Dump(), brathl::CTools::Format(), GetLength(), GetPosition(), and IsOpen().

Referenced by Duplicate().

8.57.3.13 bool brathl::CFile::Rename (const std::string & *newName*)

Renames file object If file with specified filename exists, it's contents are erased. The current file is closed, renamed and opened as new name

Parameters

newName	[in] : new file name

Returns

true on success, otherwise false

References Close(), IsOpen(), and Open().

8.57.3.14 bool brathl::CFile::Rename (const std::string & *oldName,* const std::string & *newName*) `[static]`

Renames a file If file with specified filename exists, it's contents are erased.

Parameters

oldName	[in] : file to rename
newName	[in] : new file name

Returns

true on success, otherwise false

8.57.3.15 bool brathl::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 brathl::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 brathl::CFile::SetBufferingMode (bool *mode* = `true`)

Change buffering mode. Function must be used before any read/write operation occurs!

Parameters

mode	[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 brathl::CFile::SetPosition (long *positionOffset*)

Function moves file pointer by PositionOffset bytes relative to current position.

Parameters

positionOffset	[in] : offset to move

Returns

true on success, otherwise false

References IsOpen().

Referenced by Duplicate().

8.57.3.19 bool brathl::CFile::WriteChar (const int *character*)

Writes a single character to a file If an error occurred, a CFileException is raised.

Parameters

character	[in] : character to write

Returns

true on success, otherwise false

References Dump(), brathl::CTools::Format(), IsOpen(), and modeRead.

8.57.3.20 uint32_t brathl::CFile::WriteFromBuffer (const char * *sourceBuffer,* uint32_t *sourceBufferLength*)

Writes data from memory to a file If an error occurred, a CFileException is raised.

Parameters

sourceBuffer	[in] : data to write
sourceBuffer↩ Length	[in] : data lentgh to write

Returns

the number of bytes actually written.

References Dump(), brathl::CTools::Format(), IsOpen(), and modeRead.

Referenced by Duplicate().

8.57.3.21 bool brathl::CFile::WriteString (const char * *str*)

Writes a std::string to a file If an error occurred, a CFileException is raised.

Parameters

str	[in] : std::string to write

Returns

true on success, otherwise false

References Dump(), brathl::CTools::Format(), IsOpen(), and modeRead.

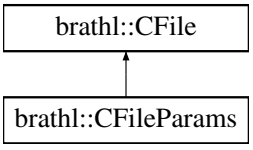
The documentation for this class was generated from the following files:

- File.h
- File.cpp

8.58 brathl::CFileParams Class Reference

`#include <FileParams.h>`

Inheritance diagram for brathl::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 fonction.
- 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

name	[in] : full name of the file;
mode	[in] : access mode - default value : modeRead|typeBinary (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

ValidMin	[in] : Minimal number of values
ValidMax	[in] : Maximal number of values. If <=0, it is considered as infinite. If < ValidMin, it is considered as equal to ValidMin.

Returns

actual number of occurences 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

name	[in] : full name of the file;
mode	[in] : access mode - default value : modeRead|typeBinary (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

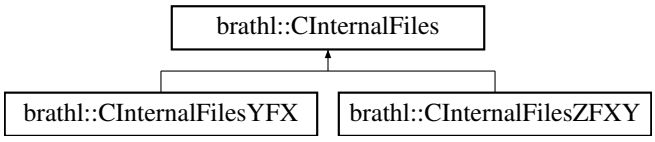
The documentation for this class was generated from the following files:

- 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 must↩ Exist=true, double defaultValue=**CTools::m_defaultValueDOUBLE**)
- int **GetAttribute** (const std::string &varName, const std::string &attName, std::string &attValue, bool must↩ Exist=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, CString↩ Array &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 **IsZFXYFile** (const std::string &fileName, CStringArray *fieldNames=NULL, **CInternalFiles** **pf=NULL)
- static bool **IsZFXYFile** (**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

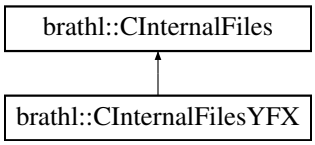
The documentation for this class was generated from the following files:

- InternalFiles.h
- InternalFiles.cpp

8.61 brathl::CInternalFilesYFX Class Reference

`#include <InternalFilesYFX.h>`

Inheritance diagram for brathl::CInternalFilesYFX:



Public Member Functions

- **CInternalFilesYFX** (std::string Name="", **brathl_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=**C↩ Tools::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=**C↩ Tools::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

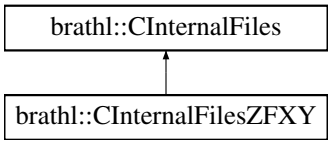
The documentation for this class was generated from the following files:

- InternalFilesYFX.h
- InternalFilesYFX.cpp

8.62 brathl::CInternalFilesZFXY Class Reference

`#include <InternalFilesZFXY.h>`

Inheritance diagram for brathl::CInternalFilesZFXY:



Public Member Functions

- **CInternalFilesZFXY** (std::string Name="", **brathl_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:

- InternalFilesZFXY.h
- InternalFilesZFXY.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 with↩ Except=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

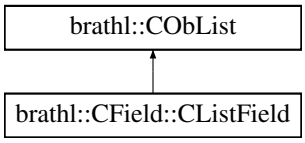
The documentation for this class was generated from the following files:

- 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().

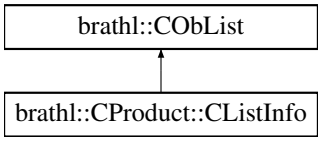
The documentation for this class was generated from the following files:

- 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

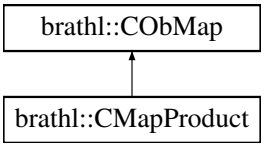
The documentation for this class was generated from the following files:

- **MapParameter.h**
- MapParameter.cpp

8.68 brathl::CMapProduct Class Reference

`#include <Product.h>`

Inheritance diagram for brathl::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

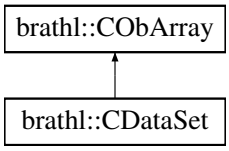
The documentation for this class was generated from the following files:

- 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 brathl::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::CObIntMap Class Reference

`#include <List.h>`

Inherits mapintobject.

Public Member Functions

- **CObIntMap** (bool bDelete=true)

  CObMap *(p. 241) ctor.*
- virtual void **Dump** (std::ostream &fOut=std::cerr) const

  Dump fonction.
- virtual bool **Erase** (CObIntMap::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 **CObIntMap** &obMap, bool withExcept=true)
- virtual const **CObIntMap** & **operator=** (const **CObIntMap** &obMap)
- virtual CBratObject * **operator[]** (int32_t key)
- virtual void **RemoveAll** ()
- bool **RenameKey** (int32_t oldKey, int32_t newKey)
- void **SetDelete** (bool value)
- virtual ~**CObIntMap** ()

  CObMap *(p. 241) dtor.*

Protected Attributes

- bool **m_bDelete**

8.71.1 Detailed Description

a set of object management classes.

Version

1.0

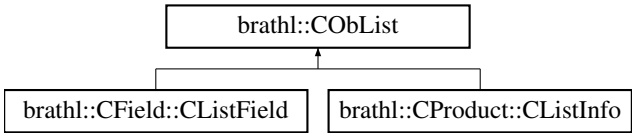
The documentation for this class was generated from the following files:

- 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

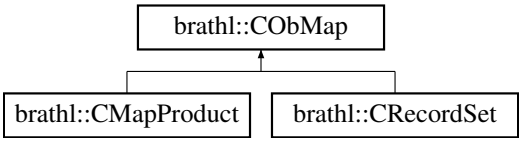
The documentation for this class was generated from the following files:

- List.h
- List.cpp

8.73 brathl::CObMap Class Reference

`#include <List.h>`

Inheritance diagram for brathl::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 fonction.

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 fonction.
- 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

keyword	[in] : parameter name

8.75.2.2 brathl::CParameter::CParameter (const char * *keyword,* const char * *value*)

Creates a new **CParameter** (p. 242) object.

Parameters

keyword	[in] : parameter name
value	[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

value	[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

value	[out] : parameter value
pos	[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

value	[out] : parameter value
bufferSize	[in] : size of value
pos	[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	[in] : index value to remove

8.75.3.7 void brathl::CParameter::SetAliases (const CStringMap & *aliases*)

Register the formulas aliases defined.

Parameters

Aliases	[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

fileName	[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

fileNameList	[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

fileName	[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

fileNameList	[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 **InitCriterialnfo** ()
- 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, **CFieldSetArrayDbl** ∗fieldSetArray↩ Dbl)
- 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

fileName	[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

fileNameList	[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

field	[in] : field to be tested.

References brathl::CTools::StringToLower().

8.78.3.4 virtual void brathl::CProductEnvisat::SetHighResolutionLatDiffFieldName (const std::string & *value*) [inline], [protected], [virtual]

Set the "High resolution latitude differences" field name

8.78.3.5 virtual void brathl::CProductEnvisat::SetHighResolutionLonDiffFieldName (const std::string & *value*) [inline], [protected], [virtual]

Set the "High resolution longitude differences" field name

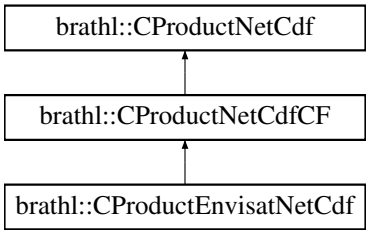
The documentation for this class was generated from the following files:

- ProductEnvisat.h
- ProductEnvisat.cpp

8.79 brathl::CProductEnvisatNetCdf Class Reference

`#include <ProductEnvisatNetCdf.h>`

Inheritance diagram for brathl::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 brathl::CProductEnvisatNetCdf::CProductEnvisatNetCdf (const std::string & *path*) [inline]

Creates new **CProductEnvisatNetCdf** (p. 249) object

Parameters

fileName	[in] : file name to be connected

8.79.2.2 brathl::CProductEnvisatNetCdf::CProductEnvisatNetCdf (const CStringList & *paths,* bool *check_only_first_files*) [inline]

Creates new **CProductEnvisatNetCdf** (p. 249) object

Parameters

fileNameList	[in] : list of file to be connected

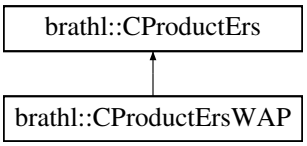
The documentation for this class was generated from the following file:

- ProductEnvisatNetCdf.h

8.80 brathl::CProductErs Class Reference

`#include <ProductErs.h>`

Inheritance diagram for brathl::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

fileName	[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

fileNameList	[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

field	[in] : field to be tested.

Reimplemented in **brathl::CProductErsWAP** (p. 253).

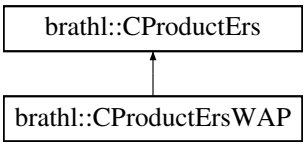
The documentation for this class was generated from the following files:

- 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

fileName	[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

fileNameList	[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

field	[in] : field to be tested.

Reimplemented from **brathl::CProductErs** (p. 251).

References brathl::CTools::Format().

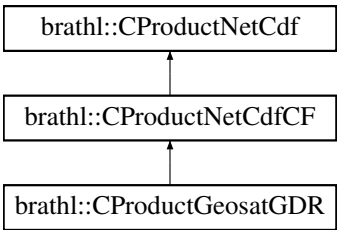
The documentation for this class was generated from the following files:

- 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

fileName	[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

fileNameList	[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

fileName	[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

fileNameList	[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

field	[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 **InitCriterialnfo** ()
- 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

fileName	[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

fileNameList	[in] : list of file to be connected

8.84.3 Member Function Documentation

8.84.3.1 bool brathl::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

field	[in] : field to be tested.

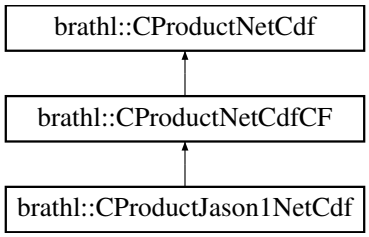
The documentation for this class was generated from the following files:

- ProductJason.h
- ProductJason.cpp

8.85 brathl::CProductJason1NetCdf Class Reference

`#include <ProductJason1NetCdf.h>`

Inheritance diagram for brathl::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

fileName	[in] : file name to be connected

8.85.2.2 brathl::CProductJason1NetCdf::CProductJason1NetCdf (const CStringList & *paths,* bool *check_only_first_file*) [inline]

Creates new **CProductJason1NetCdf** (p. 257) object

Parameters

fileNameList	[in] : list of file to be connected

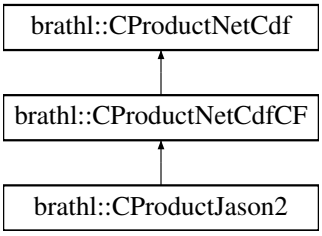
The documentation for this class was generated from the following file:

- ProductJason1NetCdf.h

8.86 brathl::CProductJason2 Class Reference

`#include <ProductJason2.h>`

Inheritance diagram for brathl::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 **HasCriterialnfo** ()
- virtual void **InitCriterialnfo** ()
- 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

fileName	[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

fileNameList	[in] : list of file to be connected

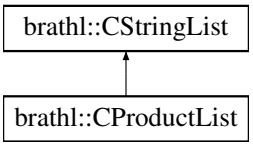
The documentation for this class was generated from the following files:

- ProductJason2.h
- ProductJason2.cpp

8.87 brathl::CProductList Class Reference

`#include <Product.h>`

Inheritance diagram for brathl::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 **IsZFXY** () 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

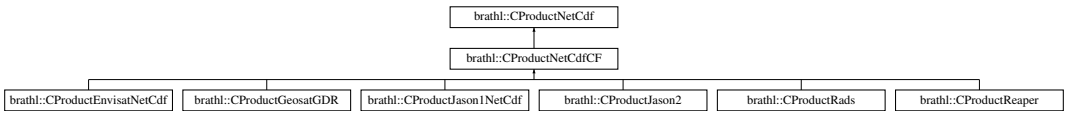
The documentation for this class was generated from the following files:

- 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** (**CCriteriaInfo** *criteriaInfo) override
- virtual bool **ApplyCriteriaDatetime** (**CCriteriaInfo** *criteriaInfo) override
- virtual bool **ApplyCriteriaLatLon** (**CCriteriaInfo** *criteriaInfo) override
- virtual bool **ApplyCriteriaPass** (**CCriteriaInfo** *criteriaInfo) override
- virtual bool **ApplyCriteriaPassInt** (**CCriteriaInfo** *criteriaInfo) override
- virtual bool **ApplyCriteriaPassString** (**CCriteriaInfo** *criteriaInfo) 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 convert↩ Date=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 CAdjustValid↩ MinMax &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

fileName	[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

fileNameList	[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 !!!)

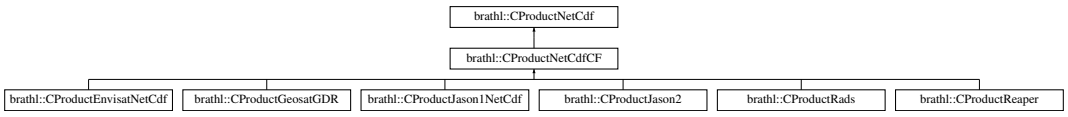
The documentation for this class was generated from the following files:

- 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

fileName	[in] : file name to be connected

8.89.2.2 brathl::CProductNetCdfCF::CProductNetCdfCF (const CStringList & *fileNameList,* bool *check_only_first_file*)

Creates new **CProductNetCdf** (p. 261) object

Parameters

fileNameList	[in] : list of file to be connected

8.89.3 Member Data Documentation

8.89.3.1 bool brathl::CProductNetCdfCF::m_atBeginning [protected]

'At beginning" flag

Referenced by Dump().

8.89.3.2 CIntMap brathl::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 brathl::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 brathl::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 brathl::CProductPodaac Class Reference

`#include <ProductPodaac.h>`

Inherits brathl::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

fileName	[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

fileNameList	[in] : list of file to be connected

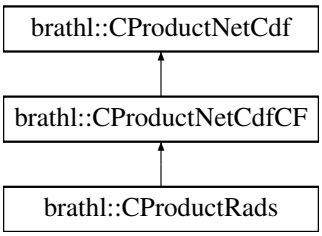
The documentation for this class was generated from the following files:

- 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 fonction.
- 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

fileName	[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

fileNameList	[in] : list of file to be connected

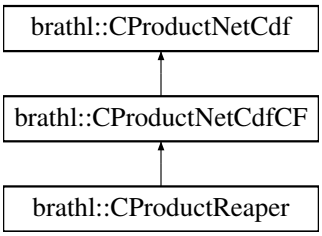
The documentation for this class was generated from the following files:

- 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

fileName	[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

fileNameList	[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

fileName	[in] : file name to be connected

8.93.2.2 brathl::CProductRiverLake::CProductRiverLake (const CStringList & *fileNameList,* bool *check_only_first_file*)

Creates new **CProductRiverLake** (p. 270) object

Parameters

fileNameList	[in] : list of file to be connected

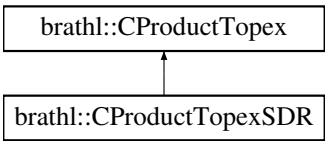
The documentation for this class was generated from the following files:

- ProductRiverLake.h
- ProductRiverLake.cpp

8.94 brathl::CProductTopex Class Reference

`#include <ProductTopex.h>`

Inheritance diagram for brathl::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

fileName	[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

fileNameList	[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

field	[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

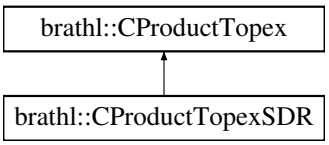
The documentation for this class was generated from the following files:

- 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** (**CFieldSetArrayDbl** ∗fieldSetArrayDbl)
- void **ComputeHighResolutionFields** (**CDataSet** ∗dataSet, double deltaLat, double deltaLon)
- virtual bool **IsHighResolutionField** (**CField** ∗field)
- virtual void **ProcessHighResolutionWithoutFieldCalculation** ()
- virtual void **PutFlatHighResolution** (**CDataSet** ∗dataSet, **CFieldSetArrayDbl** ∗fieldSetArrayDbl)
- 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

fileName	[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

fileNameList	[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

field	[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

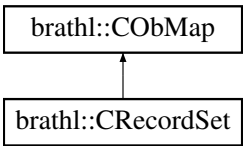
The documentation for this class was generated from the following files:

- 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

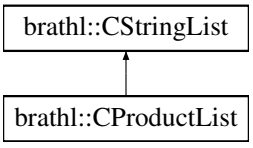
The documentation for this class was generated from the following file:

- 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 fonction.
- 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 with↩ Except=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 &regexpPattern, std::vector< std::string > &stringFound)
- static void **FindAliases** (const std::string &inText, std::vector< std::string > &aliasesFound, bool only↩ Name=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_Compare↩ Epsilon)
- 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 &regexpPattern, 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
- 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 countContinousBlock)
- 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** = 0x7FFF

  default values for int 16 bits
- static const int32_t **m_defaultValueINT32** = 0x7FFFFFFF

  default values for int 32 bits
- static const int64_t **m_defaultValueINT64** = 0x7FFFFFFFFFFFFFFFLL

  default values for unsigned int 64 bits
- static const int8_t **m_defaultValueINT8** = 0x7F

  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

in	*X*	: 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

in	*partialPath*	: 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 brathl::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 brathl::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 brathl::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 bitwize 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 brathl::CTools::BitwiseNot (double *X*) [static]

Complement an integer. The number is taken as a signed integer (int32_t). Then a bitwize 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 bitwize 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	*X*	: 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	*X*	: Number involved
in	*Y*	: 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 unpredictible.

Parameters

in	*NewValue*	: New value to take into account for statistics. Only valid values are kept; valid values are those different from default value (#IsDefaultValue#)
	in/out]	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.
	in/out]	Mean : Incremental mean
	in/out]	StdDev : Temporary value used to compute standard deviation
	in/out]	Min : Minimum value
	in/out]	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	*value*	: 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

in	*X*	: 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

in	*value*	: The std::string 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 begining by '+' are expanded in uppercase. Variables begining by '-' are expanded in lowercase.

Parameters

in	*value*	: The std::string to expand
in	*VarValues*	: The values of the variables. If NULL, the environment variables are taken.
in	*Begining*	: Char identifying the begining of a var reference
in	*Recurse*	: 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

in	*Name*	: File name

Returns

Returns true if file exists and is readable

8.102.2.20 std::string brathl::CTools::FileExtension (const std::string & *fileName*) [static]

Gets a file name extension.

Parameters

in	*filename*	: 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	*Count*	: number of valid data used for stats. If count is 0 or default value, all other output parameters are set to default value.
	in/out]	Mean : Computed mean or default value (see Count)
	in/out]	StdDev : On output, actual value of standard deviation
	in/out]	Min : Computed min or default value (see Count)
	in/out]	Max : Computed max or default value (see Count)
in	*DefaultValue*	: 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	*Name*	: 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	*filename*	: File name
in	*path*	: 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

in	*X*	: Number involved

Returns

Result of operation

8.102.2.25 std::string brathl::CTools::Format (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

in	*size*	: maximum number of characters to store
in	*format*	: format-control std::string
in	...	: optional arguments

Returns

formatted std::string

Referenced by brathl::CDate::AsString(), brathl::BuildExistingInternalFileKind(), brathl::CFileParams::Check↩Count(), brathl::CDate::CvDate(), brathl::CDoubleMap::Dump(), brathl::CObDoubleMap::Dump(), brathl::CDouble↩PtrDoubleMap::Dump(), brathl::CDataSet::EraseFieldSet(), Format(), brathl::CBratAlgorithmGeosVelGrid::Get↩InputParamDesc(), brathl::CBratAlgorithmGeosVelAtp::GetInputParamDesc(), brathl::CBratAlgoFilterMedian1D↩::GetInputParamDesc(), brathl::CBratAlgoFilterLoess1D::GetInputParamDesc(), brathl::CBratAlgoFilterLoess2D↩::GetInputParamDesc(), brathl::CBratAlgoFilterMedian2D::GetInputParamDesc(), brathl::CBratAlgorithmGeos↩VelGrid::GetInputParamFormat(), brathl::CBratAlgorithmGeosVelAtp::GetInputParamFormat(), brathl::CBrat↩AlgoFilterMedian1D::GetInputParamFormat(), brathl::CBratAlgoFilterLoess2D::GetInputParamFormat(), brathl↩::CBratAlgoFilterLoess1D::GetInputParamFormat(), brathl::CBratAlgoFilterMedian2D::GetInputParamFormat(), brathl::CBratAlgorithmGeosVelGrid::GetInputParamUnit(), brathl::CBratAlgorithmGeosVelAtp::GetInputParam↩Unit(), brathl::CBratAlgoFilterMedian1D::GetInputParamUnit(), brathl::CBratAlgoFilterMedian2D::GetInputParam↩Unit(), brathl::CBratAlgoFilterLoess2D::GetInputParamUnit(), brathl::CBratAlgoFilterLoess1D::GetInputParam↩Unit(), brathl::CParameter::GetValue(), brathl::CUIntMap::Insert(), brathl::CDataSet::InsertFieldSet(), brathl::↩CProductErsWAP::IsHighResolutionField(), brathl::CFile::Open(), brathl::CFile::ReadToBuffer(), brathl::CBrat↩AlgoFilterLanczos1D::Run(), brathl::CBratAlgoFilterGaussian1D::Run(), brathl::CBratAlgoFilterMedian1D::Run(), brathl::CBratAlgoFilterLoess1D::Run(), brathl::CDatePeriod::SetFrom(), brathl::CDatePeriod::SetTo(), Slashes↩Decode(), SlashesEncode(), brathl::CFile::WriteChar(), brathl::CFile::WriteFromBuffer(), and brathl::CFile::Write↩String().

8.102.2.26 std::string brathl::CTools::Format (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

in	*format*	: format-control std::string
in	...	: optional arguments

Returns

formatted std::string

References Format().

8.102.2.27 std::string brathl::CTools::Format (size_t *size*, const char * *format*, va_list *args*) [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

in	*size*	: maximum number of characters to store
in	*format*	: format-control std::string
in	*args*	: 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	*value*	: int to be converted

Returns

coanverted 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	*Min*	: Lower bound
in	*X*	: Number involved
in	*Max*	: Upper bound

Returns

Result of operation: 0 if not Min <= X <= 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	*Min*	: Lower bound
in	*X*	: Number involved
in	*Max*	: Upper bound

Returns

Result of operation: 0 if not Min < X < 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 otherwize

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

in	*path*	: 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

in	*X1*	: Number involved
in	*X2*	: 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

in	*X1*	: Number involved
in	*X2*	: Number involved

Returns

Result of operation

Referenced by brathl::CCriteriaLatLon::GetMinOrMaxLon().

8.102.2.40 double brathl::CTools::Minus (double *X*, double *Y*) [static]

Substracts one number from another. TAKES default values into account

Parameters

in	*X*	: Number involved
in	*Y*	: 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	*X*	: Number involved
in	*Y*	: Divider

Returns

Result of operation

8.102.2.42 double brathl::CTools::Multiply (double *X*, double *Y*) [static]

Multiply two numbers. Takes default values into account

Parameters

in	*X*	: Number involved
in	*Y*	: Number involved

Returns

Result of operation

8.102.2.43 double brathl::CTools::NormalizeLongitude (double *Floor*, double *Longitude*) [static]

Find a number satisfying the condition Floor <= Longitude < Floor+360. Takes default values into account

Parameters

in	*Floor*	: Base longitude
in	*Longitude*	: Longitude to normalize

Returns

Result of operation

8.102.2.44 double brathl::CTools::Or (double *X*, double *Y*) [static]

Do a logical or on two numbers. Takes default values into account

Parameters

in	*X*	: Number involved
in	*Y*	: Number involved

Returns

Result of operation

8.102.2.45 double brathl::CTools::Plus (double *X*, double *Y*) [static]

Add two numbers. Takes default values into account

Parameters

in	*X*	: Number involved
in	*Y*	: 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 brathl::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 * brathl::CTools::RemoveAllSpaces (char * *str*) [static]

Remove all the blank characters in a std::string. Blank characters are identified by the function isspace (3C).

Parameters

str	[in/out] : std::string to be modified

Returns

a pointer to the std::string

Referenced by StringRemoveAllSpaces().

8.102.2.49 std::string brathl::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	*str*	: The std::string to modify
in	*c1*	: the first surrounding char
in	*c2*	: the last surrounding char

Returns

the newly modified std::string.

8.102.2.50 void brathl::CTools::SetInternalDataDir (const std::string & *DataDir*) [static]

Explicitly set the Data Directory.

Parameters

in	*DataDir*	: 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

in	*X*	: 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

in	*X*	: 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

in	*X*	: 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

in	*str*	: The std::string to decode.
in	*exclude*	: A list of charters to exclude from decoding.
in	*decodeliterals*	: Set if non standard escaped literals are to be deocded.

Returns

the newly encoded std::string.

References Format().

8.102.2.55 std::string brathl::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	*str*	: The std::string to encode.
in	*exclude*	: A list of charters to exclude from encoding.
in	*literals*	:A list of printable characters to be included in the encodeing.
	hexadecimal	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	*str*	: storage location for output.
in	*size*	: maximum number of characters to store
in	*format*	: format-control std::string
in	*...*	: 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	*X*	: 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	*X*	: 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	*str1*	: std::string 1
in	*str2*	: 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	*str*	: 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	*str*	: std::string to be modified
in	*c*	: char to replace
in	*replaceBy*	: 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	*str*	: std::string to be modified
in	*c*	: std::string to replace
in	*replaceBy*	: 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

str	[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

in	*str*	: 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

str	[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

in	*value*	: std::string to be converted

Returns

coanverted 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

in	*value*	: std::string to be converted

Returns

coanverted value or CTool::m_defaultValueINT if no possible conversion.

Referenced by brathl::CCriteriaCycle::Set(), brathl::CCriteriaPassInt::Set(), brathl::CCriteriaCycle::SetFrom(), brathl::CCriteriaPassInt::SetFrom(), brathl::CCriteriaCycle::SetTo(), and brathl::CCriteriaPassInt::SetTo().

8.102.2.68 int64_t brathl::CTools::StrToInt64 (const std::string & *s*) [static]

Convert an std::string to int64

Parameters

in	*value*	: std::string to be converted

Returns

coanverted value or CTool::m_defaultValueINT if no possible conversion.

8.102.2.69 uint64_t brathl::CTools::StrToUInt64 (const std::string & *s*) [static]

Convert an std::string to uint64

Parameters

in	*value*	: std::string to be converted

Returns

coanverted value or CTool::m_defaultValueINT if no possible conversion.

8.102.2.70 double brathl::CTools::Tan (double *X*) [static]

Do the tangent of a number expressed in radians. Takes default values into account

Parameters

in	*X*	: Number involved

Returns

Result of operation

References IsInf(), and IsNan().

Referenced by TanD().

8.102.2.71 double brathl::CTools::TanD (double *X*) [static]

Do the tangent of a number expressed in degrees. Takes default values into account

Parameters

in	*X*	: 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

str	[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

in	*chr*	: character

Returns

the lowercase character

8.102.2.74 char * brathl::CTools::ToUpper (char * *str*) [static]

Set a std::string in uppercase

Parameters

str	[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

in	*chr*	: 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

in	*Text*	: String
in	*dotTrim*	: if true, remove dot at the end : 2.000 –> 2, if false, leave dot : 2.000 –> 2.

Returns

Returns modifed 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

str	[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

in	*X*	: 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

in	*X*	: 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

value	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

normalize	set to true to normalize longitude value
value	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::CObjectTree.

Public Member Functions

- virtual CObjectTreeIterator **AddChild** (CObjectTreeNode *parent, const std::string &nm, **CField** *x, bool goCurrent=false)
- virtual CObjectTreeIterator **AddChild** (CObjectTreeIterator &parent, const std::string &nm, **CField** *x, bool goCurrent=false)
- virtual CObjectTreeIterator **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** (CObjectTreeNode *node, bool withExcept=true)
- static **CFieldRecord** * **GetDataAsFieldRecordObject** (CObjectTreeNode *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 fonction.
- 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 with↩ Except=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:

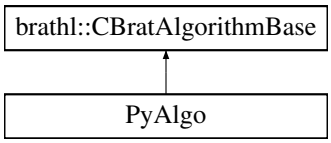
- List.h
- List.cpp

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

indexParam	[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

indexParam	[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

indexParam	[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

fmt	[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
args	[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. 103)

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 varaiable 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 **_cfleft** 1
- #define **_cfleft_cfright** 0
- #define **_cfright**
- #define **ANSI_C_preprocessor** _cfleft_cfright
- #define **FTN_NAME**(Low, Up) ADD_UNDERSCORE(Low)
- #define **INTEGER4** int32_t
- #define **MIN**(A, B) ((A) < (B) ? (A) : (B))
- #define **REAL8** double
- #define **SECOND_UNDERSCORE**(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

in	*err*	: 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