

fwidbmgr

0.1

Generated by Doxygen 1.8.1.1

Sat Dec 15 2012 13:00:17

Contents

1	FWI database manager	1
2	Distribution folder structure	3
3	Application configuration file	5
3.1	Database connection	5
3.2	Application paths	5
3.3	Application files	5
3.4	Application images	6
4	Namespace Index	7
4.1	Namespace List	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Namespace Documentation	13
7.1	fwi Namespace Reference	13
7.1.1	Detailed Description	13
7.2	grid Namespace Reference	15
7.2.1	Detailed Description	15
8	Class Documentation	17
8.1	fwi::CommandLineArguments Class Reference	17
8.1.1	Detailed Description	19
8.1.2	Member Function Documentation	19
8.1.2.1	canTryDbConnection	19
8.1.2.2	closePGConnection	20
8.1.2.3	getAction	20
8.1.2.4	getConfigFilePath	20
8.1.2.5	getConnectionString	21

8.1.2.6	getDate	21
8.1.2.7	getDbName	21
8.1.2.8	getHelp	22
8.1.2.9	getHost	22
8.1.2.10	getPassword	22
8.1.2.11	getPGConnection	23
8.1.2.12	getPort	23
8.1.2.13	getUser	24
8.1.2.14	isSetAction	24
8.1.2.15	isSetDate	24
8.1.2.16	isSetDbName	25
8.1.2.17	isSetHelp	25
8.1.2.18	isSetHost	25
8.1.2.19	isSetPassword	26
8.1.2.20	isSetPort	26
8.1.2.21	isSetUser	27
8.1.2.22	setAction	27
8.1.2.23	setConfigFilePath	28
8.1.2.24	setDate	28
8.1.2.25	setDbName	28
8.1.2.26	setHelp	29
8.1.2.27	setHost	29
8.1.2.28	setPassword	30
8.1.2.29	setPort	30
8.1.2.30	setUser	30
8.2	fwi::grid::Grid< T > Class Template Reference	31
8.2.1	Detailed Description	35
8.2.2	Constructor & Destructor Documentation	35
8.2.2.1	Grid	35
8.2.3	Member Function Documentation	36
8.2.3.1	configure	36
8.2.3.2	getCols	36
8.2.3.3	getCtlPath	37
8.2.3.4	getData	37
8.2.3.5	getDate	37
8.2.3.6	getDatPath	38
8.2.3.7	getElementsCount	38
8.2.3.8	getExportCtlPath	38
8.2.3.9	getExportDatPath	39
8.2.3.10	getFields	39

8.2.3.11	getFileNameDateOffset	40
8.2.3.12	getGradsDate	40
8.2.3.13	getIOFormat	40
8.2.3.14	getRows	40
8.2.3.15	getSlotSize	41
8.2.3.16	getSRID	41
8.2.3.17	getStartTime	42
8.2.3.18	getTable	42
8.2.3.19	getTimeBand	42
8.2.3.20	getTimeBandsNumber	43
8.2.3.21	getTimeIncrement	43
8.2.3.22	getTitle	43
8.2.3.23	getTotalElementsCount	43
8.2.3.24	getType	43
8.2.3.25	getUndefValue	44
8.2.3.26	getVarNum	44
8.2.3.27	getXDir	45
8.2.3.28	getXStart	45
8.2.3.29	getXStep	45
8.2.3.30	getYDir	46
8.2.3.31	getYStart	46
8.2.3.32	getYStep	46
8.2.3.33	insert	47
8.2.3.34	merge	47
8.2.3.35	operator()	49
8.2.3.36	operator=	49
8.2.3.37	read	49
8.2.3.38	readBand	50
8.2.3.39	readBin	50
8.2.3.40	readCtrl	51
8.2.3.41	readTxt	51
8.2.3.42	retrieve	51
8.2.3.43	setCols	52
8.2.3.44	setCtlPath	52
8.2.3.45	setDate	52
8.2.3.46	setDatPath	53
8.2.3.47	setExportCtlPath	53
8.2.3.48	setExportDatPath	53
8.2.3.49	setFields	54
8.2.3.50	setFileNameDateOffset	54

8.2.3.51	setIOFormat	54
8.2.3.52	setRows	54
8.2.3.53	setSlotSize	55
8.2.3.54	setSRID	55
8.2.3.55	setStartTime	55
8.2.3.56	setTable	55
8.2.3.57	setTimeBand	56
8.2.3.58	setTimeBandsNumber	56
8.2.3.59	setTimeIncrement	56
8.2.3.60	setTitle	56
8.2.3.61	setType	57
8.2.3.62	setUndefValue	57
8.2.3.63	setVarNum	57
8.2.3.64	setXDir	58
8.2.3.65	setXStart	58
8.2.3.66	setXStep	59
8.2.3.67	setYDir	59
8.2.3.68	setYStart	59
8.2.3.69	setYStep	59
8.2.3.70	skipBand	59
8.2.3.71	store	59
8.2.3.72	stored	60
8.2.3.73	subgrid	61
8.2.3.74	update	62
8.2.3.75	write	63
8.2.3.76	writeCtrl	63
8.2.3.77	writeTxt	64
8.3	fwl::grid::GridField Class Reference	64
8.3.1	Detailed Description	66
8.3.2	Constructor & Destructor Documentation	66
8.3.2.1	GridField	66
8.3.3	Member Function Documentation	67
8.3.3.1	getDescription	67
8.3.3.2	getFieldName	67
8.3.3.3	getLevels	67
8.3.3.4	getName	67
8.3.3.5	getPosition	67
8.3.3.6	getType	68
8.3.3.7	getUnits	68
8.3.3.8	operator=	68

8.3.3.9	operator=	68
8.3.3.10	operator==	69
8.3.3.11	operator==	69
8.3.3.12	setName	69
8.3.3.13	setPosition	69
8.3.3.14	setType	70
8.4	fwl::grid::GridFields Class Reference	70
8.4.1	Detailed Description	72
8.4.2	Constructor & Destructor Documentation	72
8.4.2.1	GridFields	72
8.4.2.2	GridFields	72
8.4.3	Member Function Documentation	73
8.4.3.1	addField	73
8.4.3.2	addField	73
8.4.3.3	getFieldByFieldName	73
8.4.3.4	getFieldByName	74
8.4.3.5	getFieldsNum	74
8.4.3.6	hasField	74
8.4.3.7	hasField	75
8.4.3.8	removeField	75
8.4.3.9	removeField	75
8.5	struct Struct Reference	75
8.5.1	Detailed Description	76
9	File Documentation	77
9.1	include/CommandLineArguments.h File Reference	77
9.1.1	Detailed Description	78
9.2	include/ctlgen.hpp File Reference	78
9.2.1	Detailed Description	79
9.3	include/fwl_define.h File Reference	80
9.3.1	Detailed Description	82
9.3.2	Macro Definition Documentation	82
9.3.2.1	ACTION_COMPUTE_INDEXES	82
9.3.2.2	ACTION_COMPUTE_INDEXES_24	82
9.3.2.3	FAILURE	82
9.3.2.4	GRD_X_START	83
9.3.2.5	GRD_Y_START	83
9.4	include/Grid.h File Reference	83
9.4.1	Detailed Description	84
9.4.2	Variable Documentation	84

9.4.2.1	logger	84
9.5	include/GridField.h File Reference	85
9.5.1	Detailed Description	86
9.6	src/CommandLineArguments.cpp File Reference	86
9.6.1	Detailed Description	87
9.7	src/fwidbmgr.cpp File Reference	87
9.7.1	Detailed Description	89
9.7.2	Function Documentation	89
9.7.2.1	compute_index	89
9.7.2.2	compute_indexes	90
9.7.2.3	create_database	90
9.7.2.4	create_standard_grid	91
9.7.2.5	delete_fwi_indexes	93
9.7.2.6	delete_images	94
9.7.2.7	delete_meteo_input	94
9.7.2.8	execute	95
9.7.2.9	export_indexes	96
9.7.2.10	fill_database	97
9.7.2.11	fill_nometeo_points	98
9.7.2.12	getFileBytea	99
9.7.2.13	getProgramHome	99
9.7.2.14	getSqlFiles	100
9.7.2.15	import_provinces	100
9.7.2.16	import_regions	101
9.7.2.17	load_computation_indexes	102
9.7.2.18	loadQueryFromFile	102
9.7.2.19	main	103
9.7.2.20	parseDate	104
9.7.2.21	prepare_fwi_indexes_grid	105
9.7.2.22	prepare_meteo_input	107
9.7.2.23	process_cmd_line	107
9.7.2.24	retrieve_fwi_indexes	109
9.7.2.25	retrieve_images	110
9.7.2.26	retrieve_meteo_input	111
9.7.2.27	store_fwi_indexes	112
9.7.2.28	store_images	114
9.7.2.29	store_meteo_input	115
9.7.2.30	usage	117
9.8	src/Grid.cpp File Reference	118
9.8.1	Detailed Description	118

9.9	src/GridField.cpp File Reference	118
9.9.1	Detailed Description	118

Chapter 1

FWI database manager

fwidbmanager is a custom application developed to store/retrieve meteo data and fire weather indexes in/from a postgresql database.

Presently fwi indexes are computed on a daily basis taking as input the following data:

- a 174x177 point grid of Lombardy starting from point (1436301.375 4916704.500) with 1500 meters resolution expressed in Gauss-Boaga (EPSG:3003). To each grid point the following information are associated
 1. *nometeo* a flag indicating the fact that no meteo data are associated to the grid point
 - 1 means that no meteo data are associated to the grid point
 - 0 means that meteo data are associated to the grid point
 2. *name* a symbolic name representing the grid point row and column in the format [row]-[column] with values starting from 1
 3. *mask* a flag indicating that the grid point is inside the region border
 - 1 means that the grid point is inside the region border
 - 0 means that the grid point is outside the region border
 4. *dz/dx* how z coordinate changes in the x direction
 5. *dz/dy* how z coordinate changes in the y direction
 6. *lake_mask* a flag indicating if the the point falls inside a lake area
 - 1 means that the grid point is inside a lake area
 - 0 means that the grid point is outside a lake area
 7. *urban_weight* a real parameter between 0.0 and 1.0 indicating the urban weight
 8. *p* the point geometry in the form (x, y, z)
- a set of GRIB files on the previous grid containing the following meteo data
 - temperature
 1. *xb* T background field
 2. *xa* T analysis field
 3. *tidi* T integral data influence
 - humidity
 1. *tdb* TD background field
 2. *ta* T analysis field
 3. *tda* ??????
 4. *rha* RH analysis field
 5. *rhidi* RH integral data influence
 6. *hdx* HUMIDEX analysis
 - wind speed

1. *bu* u background
 2. *bv* v background
 3. *bhu* horizontal u background
 4. *bhv* horizontal v background
 5. *bvw* background vertical wind
 6. *avw* analysis bertical wind
 7. *au* u analysis
 8. *av* v analysis
 9. *ahu* horizontal u analysis
 10. *ahv* horizontal v analysis
 11. *adiv* analysis divergence
 12. *avor* ?????
 13. *wsidi* WS integral data influence
- cumulated rain
1. *xpa* precipitation analysis
 2. *ridiw* wet integral data influence
 3. *ridid* dry integral data influence
 4. *snow_covering* snow covering factor
 5. *snow_dissolution* snow dissolution factor

Chapter 2

Distribution folder structure

The application resides on disk in a specific folder to be defined by the environment variable `FWIDBMGR_HOME`. For example `FWIDBMGR_HOME=~ /fwidbmgr`

Folder structure following:

config Contains the needed configuration files:

- *fwidbmgr.conf* is the application main configuration file
- *fwiscale.ini* (description is needed) not used at the moment
- *LogConfig.xml* is the log4cxx configuration file

log Contains the log file:

- *fwidbmgrDailyLog.log*

sql Contains the sql files containing the needed queries to create the fwi database from scratch

- *create_fwidb.sql* (description is needed)
- *create_fwi_indexes_table.sql* (description is needed)
- *create_grid_table.sql* (description is needed)
- *create_images_table.sql* (description is needed)
- *create_meteo_input_table.sql* (description is needed)
- *create_provinces_table.sql* (description is needed)
- *create_regions_table.sql* (description is needed)
- *partition_fwi_indexes_table.sql* (description is needed)
- *partition_meteo_input_table.sql* (description is needed)
- *postgis-64.sql* (description is needed)
- *spatial_ref_sys.sql* (description is needed)

Chapter 3

Application configuration file

The application configuration is done via a config file which format is driven by libconfig++ specifications. See the related codumentation at <http://www.hyperrealm.com/libconfig/>

The following sections describe briefly the main configuration parameters to be set in the configuration file.

- [Database connection](#)
- [Application paths](#)
- [Applications files](#)
- [Application images](#)

3.1 Database connection

The first configuration parameters to be set are those related to the database connection. You need postgres superuser credentials and the ones of a standard postgres user that must be setup in the database server before running the application. Database connection needed parameters are:

- *host* the network name or IP address of the machine running postgresql server e.g "localhost"
- *port* e.g 5432 is the standard postgresql port change it accordingly to your system setup
- *dbname* the database name e.g. "fwidb"
- *user* the standard postgres user e.g. "meteo"
- *password* standard user password e.g. "secret"
- *superuser* e.g. postgres
- *superpwd* postgres user password

3.2 Application paths

To be done.

3.3 Application files

This is a big configuration section with many subsections.

To be done

3.4 Application images

To be done

Chapter 4

Namespace Index

4.1 Namespace List

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

fwi	Main fwidbmgr namespace	13
grid	Contains all grid related classes	15

Chapter 5

Class Index

5.1 Class List

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

fwi::CommandLineArguments		
Command line arguments class	17
fwi::grid::Grid< T >		
3-dimensional grid	31
fwi::grid::GridField		
Grid field descriptor class	64
fwi::grid::GridFields		
Fields list class	70
struct		
Defines floating-point values fixed-point obj.getType()generator policy	75

Chapter 6

File Index

6.1 File List

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

include/ CommandLineArguments.h	
Command line arguments class	77
include/ ctlgen.hpp	
Grads control file generator using spirit.karma from boost libraries	78
include/ fwi_define.h	
Defines for program	80
include/ Grid.h	
2-dimensions grid class	83
include/ GridField.h	
Grid field description class	85
include/ test.hpp	??
src/ CommandLineArguments.cpp	
Command line argument class implementation	86
src/ fwidbmgr.cpp	
The main program file	87
src/ Grid.cpp	
Grid	118
src/ GridField.cpp	
Grid field class implementation	118

Chapter 7

Namespace Documentation

7.1 fwi Namespace Reference

main fwidbmgr namespace

Classes

- class [CommandLineArguments](#)
Command line arguments class.

7.1.1 Detailed Description

main fwidbmgr namespace

Computed indexes

- *FFMC* - Fine Fuel Moisture Code
- *DMC* - Duff Moisture Code
- *DC* - Drought Code
- *ISI* - Initial Spread Index
- *BUI* - Build Up Index
- *FWI* - Fire Weather Index

Index	Low	Moderate	High	Very high	Estreme
FFMC	0 - 81	81 - 88	88 - 90.5	90.5 - 92.5	92.5+
DMC	0 - 13	13 - 28	28 - 42	42 - 63	63+
DC	0 - 80	80 - 210	210 - 274	274 - 360	360+
ISI	0 - 4	4 - 8	8 - 11	11 - 19	19+
BUI	0 - 19	19 - 34	34 - 54	54 - 77	77+
FWI	0 - 5	5 - 14	14 - 21	21 - 33	33+

Not computed indexes

- *FFDC* - Forest Fire Danger Code
- *SFDC* - Scrub Fire Danger Code

- *GFDC* - Grass Fire Danger Code

FFMC (*Fine Fuel Moisture Code*) is a numerical rating of the moisture content of surface litter and other cured fine fuels. It shows the relative ease of ignition and flammability of fine fuels. The moisture content of fine fuels is very sensitive to the weather.

Even a day of rain, or of fine and windy weather, will significantly affect the FFMC rating.

The system uses a time lag of two-thirds of a day to accurately measure the moisture content in fine fuels.

The FFMC rating is on a scale of 0 to 99. Any figure above 70 is high, and above 90 is extreme.

DMC (*Duff Moisture Code*) is a numerical rating of the average moisture content of loosely compacted organic layers of moderate depth.

The code indicates the depth that fire will burn in moderate duff layers and medium size woody material.

Duff layers take longer than surface fuels to dry out but weather conditions over the past couple of weeks will significantly affect the DMC.

The system applies a time lag of 12 days to calculate the DMC.

A DMC rating of more than 30 is dry, and above 40 indicates that intensive burning will occur in the duff and medium fuels.

Burning off operations should not be carried out when the DMC rating is above 40.

DC (*Drought Code*) is a numerical rating of the moisture content of deep, compact, organic layers.

It is a useful indicator of seasonal drought and shows the likelihood of fire involving the deep duff layers and large logs.

A long period of dry weather (the system uses 52 days) is needed to dry out these fuels and affect the Drought Code.

A DC rating of 200 is high, and 300 or more is extreme indicating that fire will involve deep sub-surface and heavy fuels.

Burning off should not be permitted when the DC rating is above 300.

ISI (*Initial Spread Index*) indicates the rate fire will spread in its early stages. It is calculated from the FFMC rating and the wind factor.

The open-ended ISI scale starts at zero and a rating of 10 indicates high rate of spread shortly after ignition.

A rating of 16 or more indicates extremely rapid rate of spread.

BUI (*Build Up Index*) index shows the amount of fuel available for combustion, indicating how the fire will develop after initial spread.

It is calculated from the Duff Moisture Code and the Drought Code.

The BUI scale starts at zero and is open-ended. A rating above 40 is high, above 60 is extreme.

FWI (*Fire Weather Index*) Information from the ISI and BUI is combined to provide a numerical rating of fire intensity - the Fire Weather Index.

The FWI indicates the likely intensity of a fire. The FWI is divided into four fire danger classes: Low 0 - 7 Moderate 8 - 16 High 17 - 31 Extreme 32+

FFDC (*Forest Fire Danger Code*) Based on predicted generated "fire intensity (kw/m²)" in forest type vegetation (pine, beech).

This code denotes how difficult it would be to control a fire in this vegetation type should one start. (Low, Moderate, High, Very High, Extreme)

SFDC (*Scrub Fire Danger Code*) Based on predicted generated "fire intensity (kw/m²)" in scrub type vegetation (manuka, gorse, broom).

This code denotes how difficult it would be to control a fire in this vegetation type should one start. (Low, Moderate, High, Very High, Extreme)

GFDC (*Grass Fire Danger Code*) Based on predicted generated "fire intensity (kw/m²)" in grass type vegetation (dry grass, tussock).

This code denotes how difficult it would be to control a fire in this vegetation type should one start. (Low, Moderate, High, Very High, Extreme)

7.2 grid Namespace Reference

Contains all grid related classes.

7.2.1 Detailed Description

Contains all grid related classes.

Chapter 8

Class Documentation

8.1 fwi::CommandLineArguments Class Reference

Command line arguments class.

```
#include <CommandLineArguments.h>
```

Collaboration diagram for fwi::CommandLineArguments:

fwi::CommandLineArguments
<div>+ CommandLineArguments() + ~CommandLineArguments() + getConfigFilePath() + getAction() + getDate() + getDbName() + getHost() + getPort() + getUser() + getPassword() and 22 more...</div>

Public Member Functions

- [CommandLineArguments](#) ()
Standard constructor.
- virtual [~CommandLineArguments](#) ()
Destructor.
- string [getConfigFilePath](#) () const
Config file path getter.
- string [getAction](#) () const
Action getter.

- string [getDate](#) () const
Date getter.
- string [getDbName](#) () const
Database name getter.
- string [getHost](#) () const
Host getter.
- int [getPort](#) () const
Port getter.
- string [getUser](#) () const
User getter.
- string [getPassword](#) () const
Password getter.
- bool [getHelp](#) () const
Help argument presence.
- PGconn * [getPGConnection](#) (Config &cfg, bool create=false)
Gets postgresql connection.
- void [closePGConnection](#) ()
Gently close postgresql connection.
- void [setConfigFilePath](#) (string cfgpath)
ConfigFilePath setter.
- void [setAction](#) (string action)
Action setter.
- void [setDate](#) (string date)
Date setter.
- void [setHost](#) (string host)
Host setter.
- void [setDbName](#) (string dbname)
Database name setter.
- void [setPort](#) (int port)
Port setter.
- void [setUser](#) (string user)
User setter.
- void [setPassword](#) (string password)
Password setter.
- void [setHelp](#) (bool help)
Help flag setter.
- bool [isSetAction](#) ()
Checks for action setting.
- bool [isSetDate](#) ()
Checks for date setting.
- bool [isSetHost](#) ()
Checks for host setting.
- bool [isSetDbName](#) ()
Checks for database name setting.
- bool [isSetPort](#) ()
Checks for port setting.
- bool [isSetUser](#) ()
Checks for user setting.
- bool [isSetPassword](#) ()
Checks for password setting.
- bool [isSetHelp](#) ()

- *Checks for help setting.*
- bool `canTryDbConnection` ()
Checks if a database connection could be done based on current settings.
- string `getConnectionString` ()
Gets the connection string based on current settings.

8.1.1 Detailed Description

Command line arguments class.

```
This class stores and manages command line arguments passed to fwidbmgr.<br />
<h2>fwidbmgr synopsis</h2>
<i>fwidbmgr -a action [-d date] [-D database] [-H host] [-P port] [-U user] [-p password] [-h]</i><br />
The possible actions for <b>fwidbmgr</b> are:<br />
<table border="0">
  <tr><td><b>create</b></td><td>create an empty database structure</td></tr>
  <tr><td><b>createstdgrid</b></td><td>creates the standard 177x174 point grid</td></tr>
  <tr><td><b>fillnometeo</b></td><td>fills nometeo field in standard grid</td></tr>
  <tr><td><b>fillregionmask</b></td><td>fills mask field in standard grid</td></tr>
  <tr><td><b>in</b></td><td>save in db input data for date given by option date</td></tr>
  <tr><td><b>out</b></td><td>save in db output data of fwi indexes computation</td></tr>
  <tr><td><b>outimg</b></td><td>save in db output images</td></tr>
</table>
```

- *date* must be a valid date in ISO 8601 format ex. (2012-03-22)
- *database* is the database name to be used
- *host* is the database host name or IP address
- *port* is the postgresql port
- *user* is the database user that has the proper rights
- *where* password is the user password
- **h ->** prints this text

Definition at line 64 of file `CommandLineArguments.h`.

8.1.2 Member Function Documentation

8.1.2.1 bool fwi::CommandLineArguments::canTryDbConnection ()

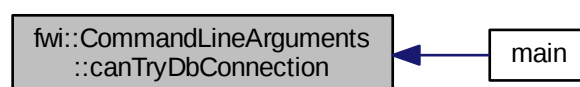
Checks if a database connection could be done based on current settings.

Returns

true if can try connection else false

Definition at line 204 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.2 void fwi::CommandLineArguments::closePGConnection ()

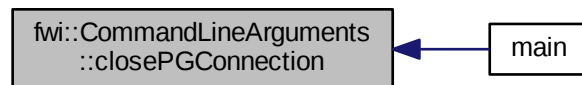
Gently close postgresql connection.

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 106 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.3 string fwi::CommandLineArguments::getAction () const

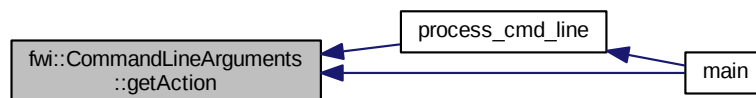
Action getter.

Returns

action argument

Definition at line 43 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.4 string fwi::CommandLineArguments::getConfigFilePath () const

Config file path getter.

Returns

configFilePath argument

Definition at line 38 of file CommandLineArguments.cpp.

8.1.2.5 string fwi::CommandLineArguments::getConnectionString ()

Gets the connection string based on current settings.

Returns

the connection string

Definition at line 214 of file CommandLineArguments.cpp.

8.1.2.6 string fwi::CommandLineArguments::getDate () const

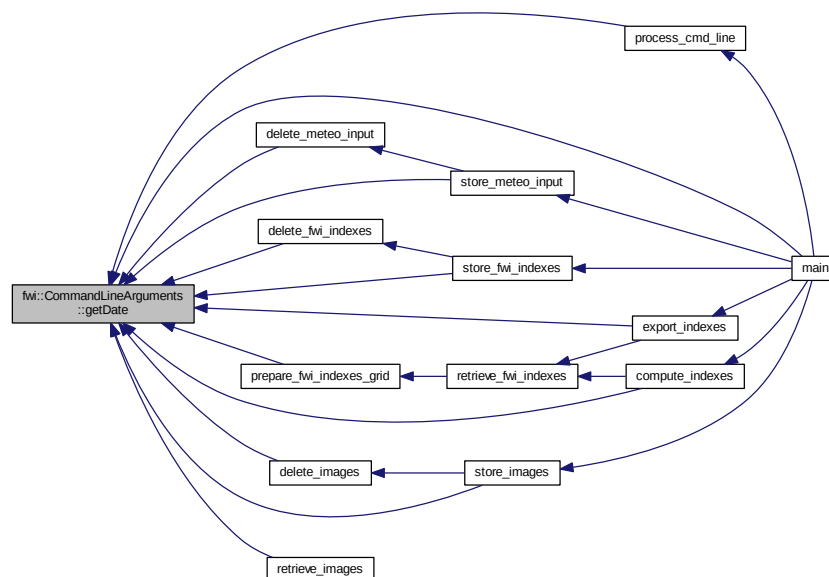
Date getter.

Returns

date argument

Definition at line 48 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.7 string fwi::CommandLineArguments::getDbName () const

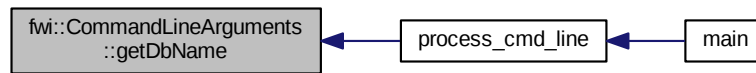
Database name getter.

Returns

database name argument

Definition at line 53 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.8 `bool fwi::CommandLineArguments::getHelp () const`

Help argument presence.

Returns

true help argument passed to program else false

Definition at line 78 of file `CommandLineArguments.cpp`.

8.1.2.9 `string fwi::CommandLineArguments::getHost () const`

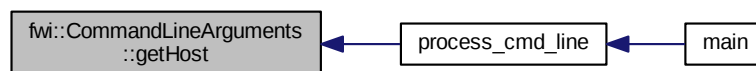
Host getter.

Returns

host argument

Definition at line 58 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.10 `string fwi::CommandLineArguments::getPassword () const`

Password getter.

Returns

password argument

Definition at line 63 of file `CommandLineArguments.cpp`.

8.1.2.11 PGconn * fwi::CommandLineArguments::getPGConnection (Config & *cfg*, bool *create* = false)

Gets postgresql connection.

Parameters

<i>cfg</i>	configuration class from libconfig++
<i>create</i>	if true create a new connection

Returns

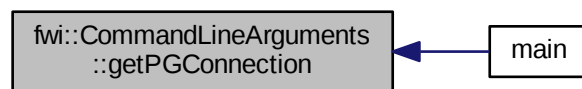
postgresql connection

See also

libconfig++ documentation at <http://www.hyperrealm.com/libconfig/>
postgresql documentation at <http://www.postgresql.org/>

Definition at line 83 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.12 int fwi::CommandLineArguments::getPort () const

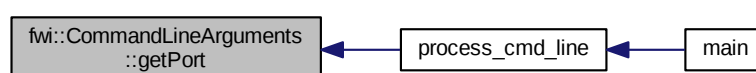
Port getter.

Returns

port argument

Definition at line 68 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.13 `string fwi::CommandLineArguments::getUser () const`

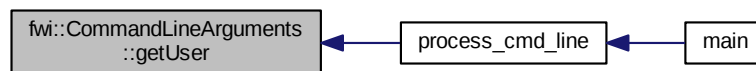
User getter.

Returns

user argument

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

Here is the caller graph for this function:



8.1.2.14 `bool fwi::CommandLineArguments::isSetAction ()`

Checks for action setting.

Returns

true if action is set else false

Definition at line 160 of file `CommandLineArguments.cpp`.

8.1.2.15 `bool fwi::CommandLineArguments::isSetDate ()`

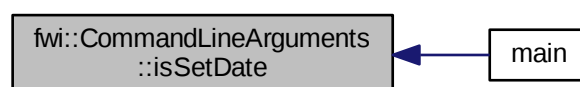
Checks for date setting.

Returns

true if date is set else false

Definition at line 165 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.16 bool fwi::CommandLineArguments::isSetDbName ()

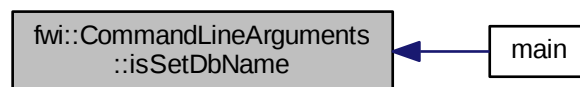
Checks for database name setting.

Returns

true if database name is set else false

Definition at line 175 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.17 bool fwi::CommandLineArguments::isSetHelp ()

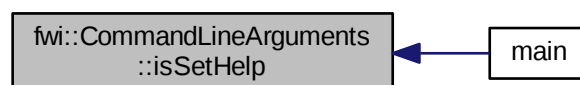
Checks for help setting.

Returns

true if help is set else false

Definition at line 195 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.18 bool fwi::CommandLineArguments::isSetHost ()

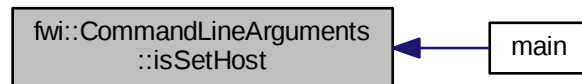
Checks for host setting.

Returns

true if host is set else false

Definition at line 170 of file CommandLineArguments.cpp.

Here is the caller graph for this function:

**8.1.2.19 bool fwi::CommandLineArguments::isSetPassword ()**

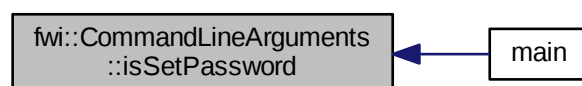
Checks for password setting.

Returns

true if password is set else false

Definition at line 180 of file CommandLineArguments.cpp.

Here is the caller graph for this function:

**8.1.2.20 bool fwi::CommandLineArguments::isSetPort ()**

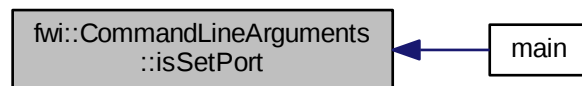
Checks for port setting.

Returns

true if port is set else false

Definition at line 185 of file CommandLineArguments.cpp.

Here is the caller graph for this function:

**8.1.2.21 bool fwi::CommandLineArguments::isSetUser ()**

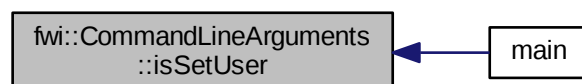
Checks for user setting.

Returns

true if user is set else false

Definition at line 190 of file CommandLineArguments.cpp.

Here is the caller graph for this function:

**8.1.2.22 void fwi::CommandLineArguments::setAction (string action)**

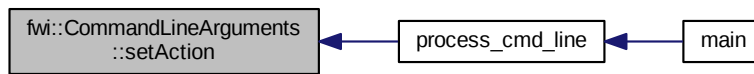
Action setter.

Parameters

<i>action</i>	action to set
---------------	---------------

Definition at line 120 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.23 void fwi::CommandLineArguments::setConfigFilePath (string *cfgpath*)

ConfigFilePath setter.

Parameters

<i>cfgpath</i>	configFilePath to set
----------------	-----------------------

Definition at line 115 of file `CommandLineArguments.cpp`.

8.1.2.24 void fwi::CommandLineArguments::setDate (string *date*)

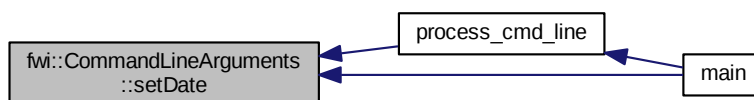
Date setter.

Parameters

<i>date</i>	date to set
-------------	-------------

Definition at line 125 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.25 void fwi::CommandLineArguments::setDbName (string *dbname*)

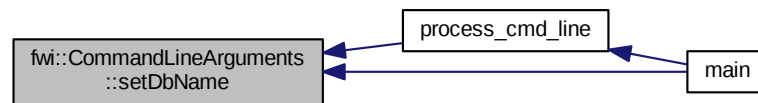
Database name setter.

Parameters

<i>dbname</i>	database name to set
---------------	----------------------

Definition at line 130 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.26 void fwi::CommandLineArguments::setHelp (bool *help*)

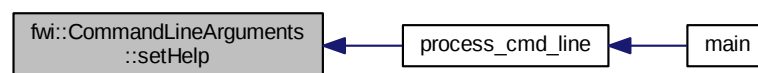
Help flag setter.

Parameters

<i>help</i>	help flag to set
-------------	------------------

Definition at line 155 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.27 void fwi::CommandLineArguments::setHost (string *host*)

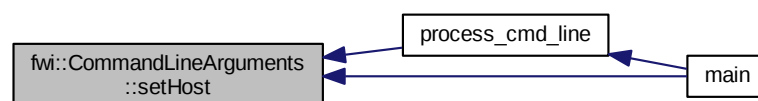
Host setter.

Parameters

<i>host</i>	host to set
-------------	-------------

Definition at line 135 of file `CommandLineArguments.cpp`.

Here is the caller graph for this function:



8.1.2.28 void fwi::CommandLineArguments::setPassword (string *password*)

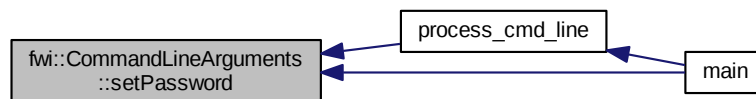
Password setter.

Parameters

<i>password</i>	password to set
-----------------	-----------------

Definition at line 140 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.29 void fwi::CommandLineArguments::setPort (int *port*)

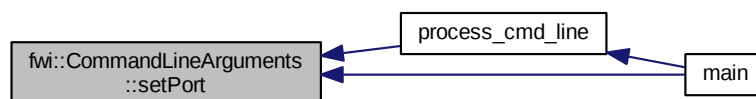
Port setter.

Parameters

<i>port</i>	port to set
-------------	-------------

Definition at line 145 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



8.1.2.30 void fwi::CommandLineArguments::setUser (string *user*)

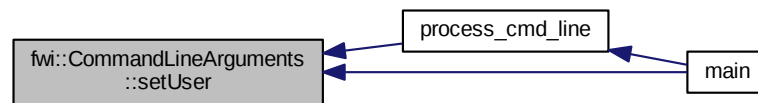
User setter.

Parameters

<i>user</i>	user to set
-------------	-------------

Definition at line 150 of file CommandLineArguments.cpp.

Here is the caller graph for this function:



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

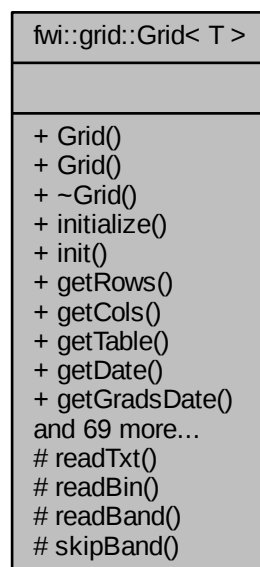
- [include/CommandLineArguments.h](#)
- [src/CommandLineArguments.cpp](#)

8.2 fwi::grid::Grid< T > Class Template Reference

3-dimensional grid

```
#include <Grid.h>
```

Collaboration diagram for `fwi::grid::Grid< T >`:



Public Types

- `typedef grid_t::index` [index](#)
typedef helper

Public Member Functions

- [Grid](#) (int varnum=[GRD_DEFAULT_VARNUM](#))
Standard constructor.
- [Grid](#) (int rows, int cols, std::string table=[GRD_DEFAULT_TABLE](#), int type=[GEOGRID](#), float xstart=[GRD_X_START](#), float ystart=[GRD_Y_START](#), float xstep=[GRD_X_STEP](#), float ystep=[GRD_Y_STEP](#), COORDINATE_DIRECTION xdir=INCREASING, COORDINATE_DIRECTION ydir=INCREASING, int varnum=[GRD_DEFAULT_VARNUM](#), int srid=[GIS_DEFAULT_SRID](#), float undefValue=[GRD_DEFAULT_UNDEF_VALUE](#), int slotSize=[GRD_DEFAULT_SLOTSIZE](#))
Parameterized constructor.
- virtual [~Grid](#) ()
Destructor.
- void [initialize](#) ()
Initialize grid memory.
- void [init](#) (T t)
set all grid elements to t
- int [getRows](#) () const
rows number getter
- int [getCols](#) () const
cols number getter
- std::string [getTable](#) () const
grid database table name getter
- std::string [getDate](#) () const
date getter
- std::string [getGradsDate](#) () const
date in GrADS format getter
- std::string [getCtlPath](#) () const
grid ctl file path getter
- std::string [getDatPath](#) () const
grid dat file path getter
- std::string [getExportDatPath](#) () const
grid export dat file path getter
- std::string [getExportCtlPath](#) () const
grid export ctl file path getter
- std::string [getTitle](#) () const
grid title getter
- int [getType](#) () const
grid type getter
- int [getTimeBand](#) () const
grid time band
- int [getTimeBandsNumber](#) () const
grid time bands number
- std::string [getStartTime](#) () const
grid start time getter
- std::string [getTimeIncrement](#) () const
grid time increment getter
- int [getFileNameDateOffset](#) () const
grid file name date offset getter
- int [getIOFormat](#) () const
grid I/O format getter
- float [getXStart](#) () const

- grid start x coordinate getter*
- float [getXStep](#) () const
- grid step in x direction getter*
- float [getYStart](#) () const
- grid start y coordinate getter*
- float [getYStep](#) () const
- grid step in y direction getter*
- COORDINATE_DIRECTION [getXDir](#) () const
- xDir getter*
- COORDINATE_DIRECTION [getYDir](#) () const
- yDir getter*
- int [getVarNum](#) () const
- variables number getter*
- int [getSRID](#) () const
- grid srid getter*
- float [getUndefValue](#) () const
- grid undefindined value getter*
- int [getSlotSize](#) () const
- grid slot size getter*
- grid_t [getData](#) ()
- grid internal data pointer getter*
- [GridFields](#) * [getFields](#) () const
- grid fields list getter*
- int [getElementsCount](#) ()
- gets the element count as rows x cols*
- int [getTotalElementsCount](#) ()
- gets the total element count as rows x cols x varNum*
- void [setRows](#) (int rows)
- grid rows number setter*
- void [setCols](#) (int cols)
- grid columns number setter*
- void [setTable](#) (std::string table)
- grid table name setter*
- void [setDate](#) (std::string date)
- date setter*
- void [setCtlPath](#) (std::string filepath)
- grid ctl file path setter*
- void [setDatPath](#) (std::string filepath)
- grid dat file path setter*
- void [setExportCtlPath](#) (std::string filepath)
- grid export ctl file path setter*
- void [setExportDatPath](#) (std::string filepath)
- grid export dat file path setter*
- void [setTitle](#) (std::string title)
- grid title setter*
- void [setType](#) (int type)
- grid type setter*
- void [setTimeBand](#) (int band)
- grid time band setter*
- void [setTimeBandsNumber](#) (int bandsNumber)
- grid time bands number setter*

- void [setStartTime](#) (std::string t)
grid start time setter
- void [setTimeIncrement](#) (std::string increment)
grid time increment setter
- void [setFileNameDateOffset](#) (int offset)
grid file name date offset setter
- void [setIOFormat](#) (int format)
grid I/O format setter
- void [setXStart](#) (float xstart)
grid start x coordinate setter
- void [setXStep](#) (float xstep)
grid step in x direction setter
- void [setYStart](#) (float ystart)
grid start y coordinate setter
- void [setYStep](#) (float ystep)
grid step in y direction setter
- void [setXDir](#) (COORDINATE_DIRECTION xDir)
xDir setter
- void [setYDir](#) (COORDINATE_DIRECTION yDir)
yDir setter
- void [setVarNum](#) (int varnum)
grid variables number setter
- void [setSRID](#) (int srid)
grid srid setter
- void [setUndefValue](#) (float undefValue)
grid undefined value setter
- void [setSlotSize](#) (int slotSize)
grid slot size setter
- void [setFields](#) ([GridFields](#) *fields)
grid fields list setter
- T & [operator\(\)](#) (int i, int j, int k)
grid element access helper
- [Grid](#)< T > & [operator=](#) (const [Grid](#)< T > &grid)
grid assignement operator
- void [raw_dump](#) ()
raw dump helper
- bool [configure](#) (std::string name, Config &cfg)
configure grid from config file
- bool [merge](#) ([Grid](#)< T > &other)
*merges other whith **this***
- bool [subgrid](#) (std::vector< std::string > fieldnames, [Grid](#)< T > &sg)
Extracts a subgrid from this having fields contained in fieldnames
- bool [readCtrl](#) (ifstream &in)
reads grid control file
- bool [read](#) ()
reads grid binary file
- bool [writeCtrl](#) (ofstream &out)
write grid ctl file
- bool [write](#) (ofstream &out)
writes grid binary file
- bool [writeTxt](#) (ofstream &out, bool esri=false)

- writes grid text file*
- bool [stored](#) (PGconn *conn)
*verify if **this** is already stored in database*
- bool [store](#) (PGconn *conn)
stores grid in database
- bool [insert](#) (PGconn *conn)
insert grid in database
- bool [update](#) (PGconn *conn)
updates grid in database
- bool [retrieve](#) (PGconn *conn)
retrieves grid from database

Protected Member Functions

- bool [readTxt](#) (ifstream &in)
reads grid data from text file
- bool [readBin](#) (ifstream &in)
reads grid data from binary file
- bool [readBand](#) (ifstream &in)
reads data from a grid time band (stream must be opened in binary mode) not applicable to text streams
- void [skipBand](#) (ifstream &in)
skip the next timeband from reading

Friends

- ostream & [operator<<](#) (ostream &stream, Point &p)
output stream operator for Point data type
- ostream & [operator<<](#) (ostream &stream, Point *p)
output stream operator for Point data type*
- istream & [operator>>](#) (istream &stream, Point &p)
input stream operator for Point data type
- istream & [operator>>](#) (istream &stream, Point *p)
input stream operator for Point data type*

8.2.1 Detailed Description

template<typename T>class fwi::grid::Grid< T >

3-dimensional grid

Definition at line 70 of file Grid.h.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 template<typename T > fwi::grid::Grid< T >::Grid (int rows, int cols, std::string table = GRD_DEFAULT_TABLE, int type = GEOGRID, float xstart = GRD_X_START, float ystart = GRD_Y_START, float xstep = GRD_X_STEP, float ystep = GRD_Y_STEP, COORDINATE.DIRECTION xdir = INCREASING, COORDINATE.DIRECTION ydir = INCREASING, int varnum = GRD_DEFAULT_VARNUM, int srid = GIS_DEFAULT_SRID, float undefValue = GRD_DEFAULT_UNDEF_VALUE, int slotSize = GRD_DEFAULT_SLOTSIZE)

Parameterized constructor.

Parameters

<i>rows</i>	rows number
<i>cols</i>	columns number
<i>table</i>	grid table name
<i>type</i>	grid type
<i>xstart</i>	grid start x coordinate
<i>ystart</i>	grid y start coordinate
<i>xstep</i>	grid step in x direction
<i>ystep</i>	grid step in y direction
<i>xdir</i>	x coordinate changing direction
<i>ydir</i>	y coordinate changing direction
<i>varnum</i>	variables number
<i>srid</i>	grid srid
<i>undefValue</i>	undefined value
<i>slotSize</i>	slot size

Definition at line 996 of file Grid.h.

8.2.3 Member Function Documentation

8.2.3.1 `template<typename T> bool fwi::grid::Grid< T >::configure (std::string name, Config & cfg)`

configure grid from config file

Parameters

<i>name</i>	grid name as present in config file
<i>cfg</i>	configuration class from libconfig++

Returns

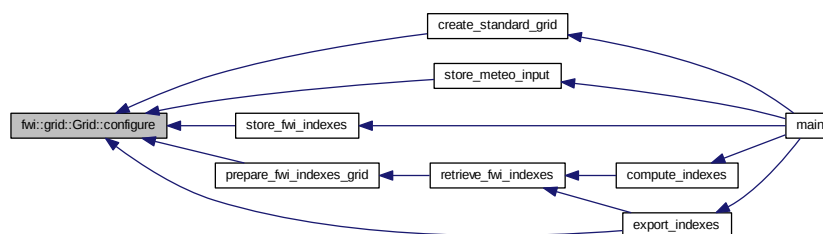
true on success else false

See also

libconfig++ documentation at <http://www.hyperrealm.com/libconfig/>

Definition at line 1532 of file Grid.h.

Here is the caller graph for this function:



8.2.3.2 `template<typename T> int fwi::grid::Grid< T >::getCols () const`

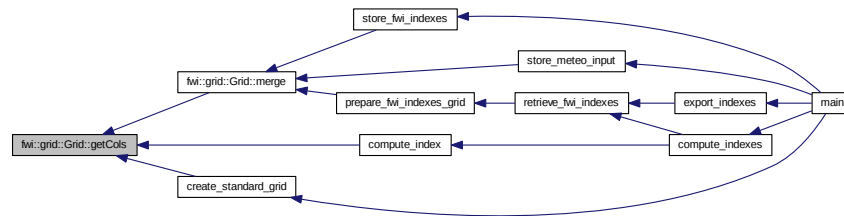
cols number getter

Returns

grid cols number

Definition at line 1051 of file Grid.h.

Here is the caller graph for this function:



8.2.3.3 `template<typename T> std::string fwi::grid::Grid< T >::getCtlPath () const`

grid ctl file path getter

Returns

grid ctl file full path

Definition at line 1112 of file Grid.h.

8.2.3.4 `template<typename T> Grid< T >::grid_t fwi::grid::Grid< T >::getData ()`

grid internal data pointer getter

Returns

internal data representation

Definition at line 1244 of file Grid.h.

8.2.3.5 `template<typename T> std::string fwi::grid::Grid< T >::getDate () const`

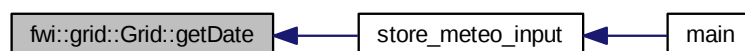
date getter

Returns

date current value

Definition at line 1063 of file Grid.h.

Here is the caller graph for this function:



8.2.3.6 `template<typename T> std::string fwi::grid::Grid< T >::getDatPath () const`

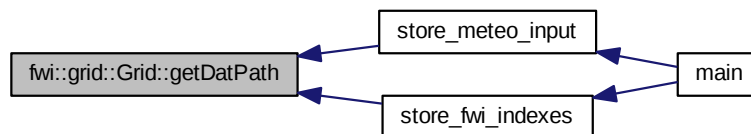
grid dat file path getter

Returns

grid datfile full path

Definition at line 1118 of file Grid.h.

Here is the caller graph for this function:



8.2.3.7 `template<typename T> int fwi::grid::Grid< T >::getElementsCount ()`

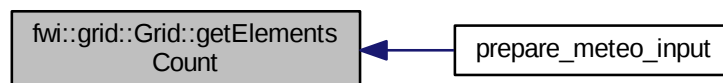
gets the element count as rows x cols

Returns

element number only x and y dimensions

Definition at line 1256 of file Grid.h.

Here is the caller graph for this function:



8.2.3.8 `template<typename T> std::string fwi::grid::Grid< T >::getExportCtlPath () const`

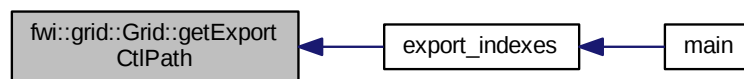
grid export ctl file path getter

Returns

grid export ctl file full path

Definition at line 1124 of file Grid.h.

Here is the caller graph for this function:

**8.2.3.9** `template<typename T> std::string fwi::grid::Grid< T >::getExportDatPath () const`

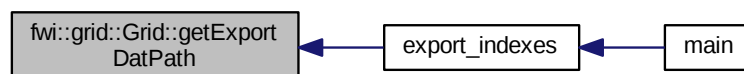
grid export dat file path getter

Returns

grid export datfile full path

Definition at line 1130 of file Grid.h.

Here is the caller graph for this function:

**8.2.3.10** `template<typename T> GridFields * fwi::grid::Grid< T >::getFields () const`

grid fields list getter

Returns

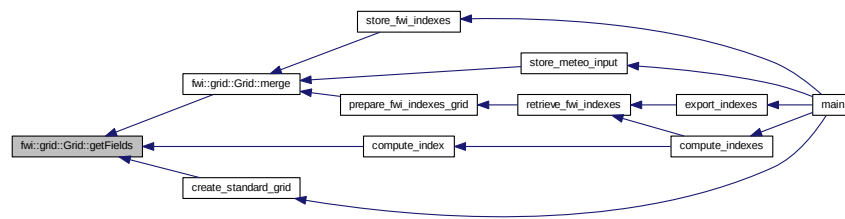
grid fields list

See also

[GridFields](#)

Definition at line 1250 of file Grid.h.

Here is the caller graph for this function:



8.2.3.11 `template<typename T> int fwi::grid::Grid< T >::getFileNameDateOffset () const`

grid file name date offset getter

Returns

the grid file name date offset in days (+/-)

Definition at line 1172 of file Grid.h.

8.2.3.12 `template<typename T> std::string fwi::grid::Grid< T >::getGradsDate () const`

date in GrADS format getter

Returns

date current value

Definition at line 1084 of file Grid.h.

8.2.3.13 `template<typename T> int fwi::grid::Grid< T >::getIOFormat () const`

grid I/O format getter

Returns

grid I/O format

See also

[fwi_define.h](#)

Definition at line 1178 of file Grid.h.

8.2.3.14 `template<typename T> int fwi::grid::Grid< T >::getRows () const`

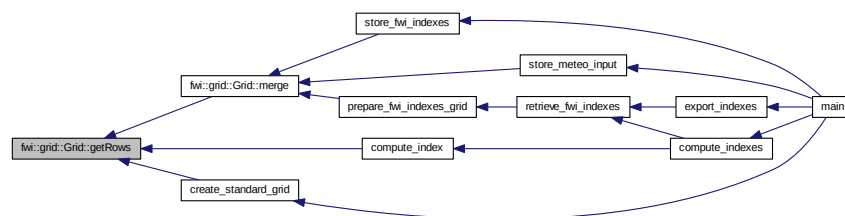
rows number getter

Returns

grid rows number

Definition at line 1045 of file Grid.h.

Here is the caller graph for this function:



8.2.3.15 `template<typename T> int fwi::grid::Grid< T >::getSlotSize () const`

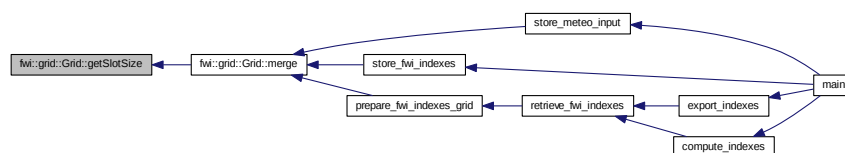
grid slot size getter

Returns

grid slot size

Definition at line 1238 of file Grid.h.

Here is the caller graph for this function:



8.2.3.16 `template<typename T> int fwi::grid::Grid< T >::getSRID () const`

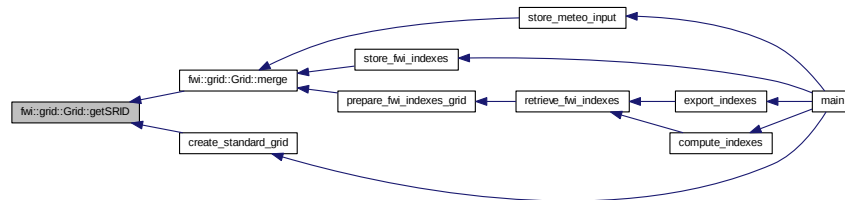
grid srid getter

Returns

grid srid

Definition at line 1226 of file Grid.h.

Here is the caller graph for this function:



8.2.3.17 `template<typename T> std::string fwi::grid::Grid< T >::getStartTime () const`

grid start time getter

Returns

the grid start time in GrADS format

Definition at line 1160 of file Grid.h.

8.2.3.18 `template<typename T> std::string fwi::grid::Grid< T >::getTable () const`

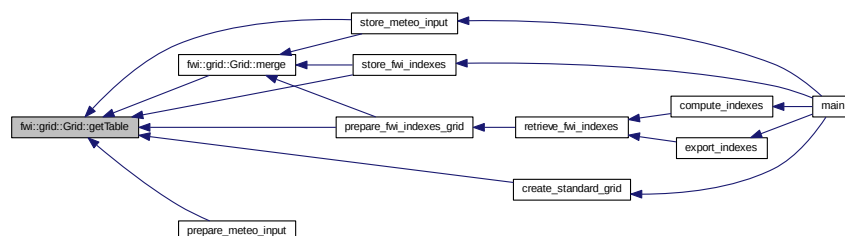
grid database table name getter

Returns

table name

Definition at line 1057 of file Grid.h.

Here is the caller graph for this function:



8.2.3.19 `template<typename T> int fwi::grid::Grid< T >::getTimeBand () const`

grid time band

Returns

the grid time band

Definition at line 1148 of file Grid.h.

8.2.3.20 `template<typename T> int fwi::grid::Grid< T >::getTimeBandsNumber () const`

grid time bands number

Returns

the grid time bands number

Definition at line 1154 of file Grid.h.

8.2.3.21 `template<typename T> std::string fwi::grid::Grid< T >::getTimeIncrement () const`

grid time increment getter

Returns

the grid time increment in GrADS format

Definition at line 1166 of file Grid.h.

8.2.3.22 `template<typename T> std::string fwi::grid::Grid< T >::getTitle () const`

grid title getter

Returns

the grid title as in grib files

Definition at line 1136 of file Grid.h.

8.2.3.23 `template<typename T> int fwi::grid::Grid< T >::getTotalElementsCount ()`

gets the total element count as rows x cols x varNum

Returns

element total number

Definition at line 1262 of file Grid.h.

8.2.3.24 `template<typename T> int fwi::grid::Grid< T >::getType () const`

grid type getter

Returns

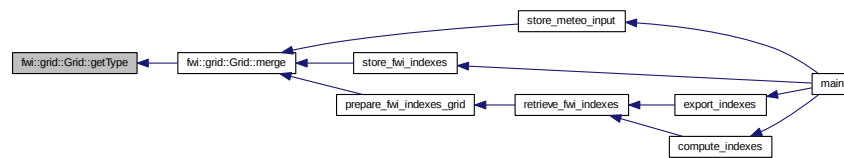
grid type

See also

[fwi_define.h](#)

Definition at line 1142 of file Grid.h.

Here is the caller graph for this function:



8.2.3.25 `template<typename T> float fwi::grid::Grid< T >::getUndefValue () const`

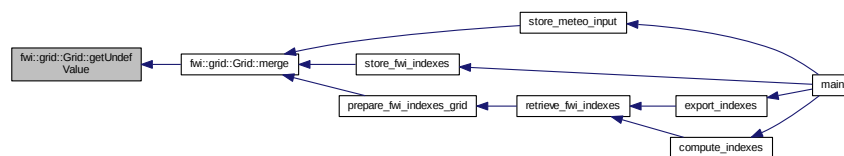
grid undefindined value getter

Returns

grid undefined value

Definition at line 1232 of file Grid.h.

Here is the caller graph for this function:



8.2.3.26 `template<typename T> int fwi::grid::Grid< T >::getVarNum () const`

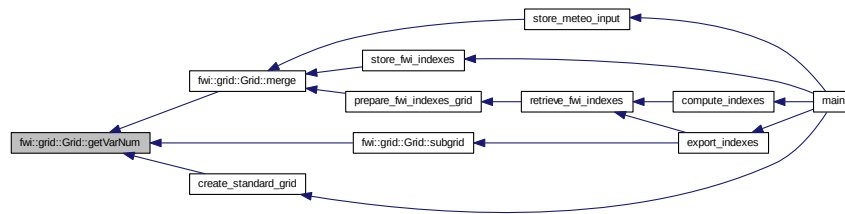
variables number getter

Returns

variables number

Definition at line 1220 of file Grid.h.

Here is the caller graph for this function:



8.2.3.27 `template<typename T> COORDINATE_DIRECTION fwi::grid::Grid< T >::getXDir () const`

xDir getter

Returns

current value for xDir

Definition at line 1208 of file Grid.h.

8.2.3.28 `template<typename T> float fwi::grid::Grid< T >::getXStart () const`

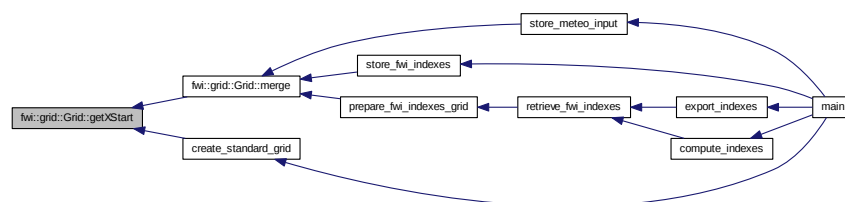
grid start x coordinate getter

Returns

grid start x coordinate

Definition at line 1184 of file Grid.h.

Here is the caller graph for this function:



8.2.3.29 `template<typename T> float fwi::grid::Grid< T >::getXStep () const`

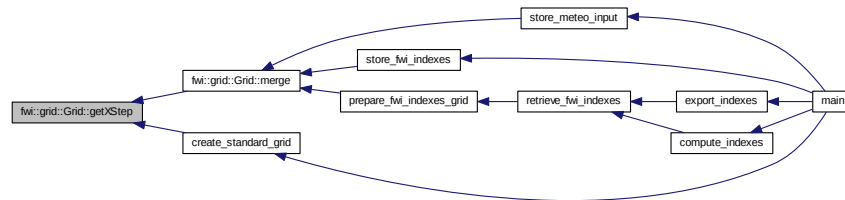
grid step in x direction getter

Returns

grid step in x direction

Definition at line 1190 of file Grid.h.

Here is the caller graph for this function:



8.2.3.30 `template<typename T> COORDINATE_DIRECTION fwi::grid::Grid< T >::getYDir () const`

yDir getter

Returns

current value for yDir

Definition at line 1214 of file Grid.h.

8.2.3.31 `template<typename T> float fwi::grid::Grid< T >::getYStart () const`

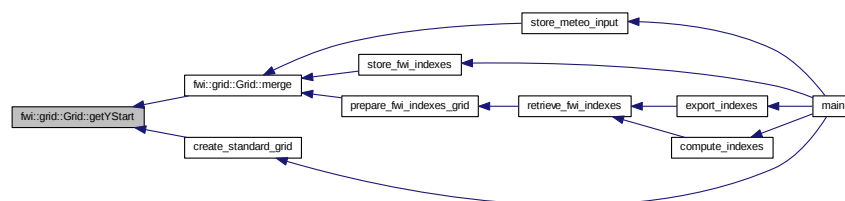
grid start y coordinate getter

Returns

grid start y coordinate

Definition at line 1196 of file Grid.h.

Here is the caller graph for this function:



8.2.3.32 `template<typename T> float fwi::grid::Grid< T >::getYStep () const`

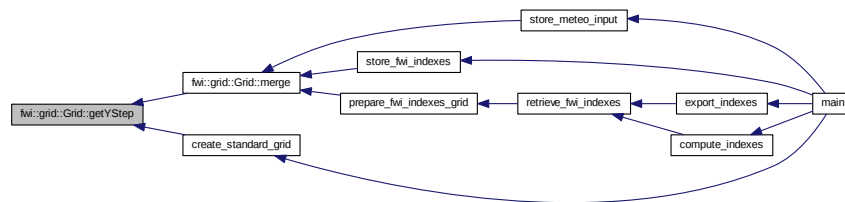
grid step in y direction getter

Returns

grid step in y direction

Definition at line 1202 of file Grid.h.

Here is the caller graph for this function:



8.2.3.33 template<typename T> bool fwi::grid::Grid< T >::insert (PGconn * conn)

insert grid in database

Parameters

<i>conn</i>	postgresql connection
-------------	-----------------------

Returns

true on success else false

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 2213 of file Grid.h.

8.2.3.34 template<typename T> bool fwi::grid::Grid< T >::merge (Grid< T > & other)

merges *other* with **this**

Parameters

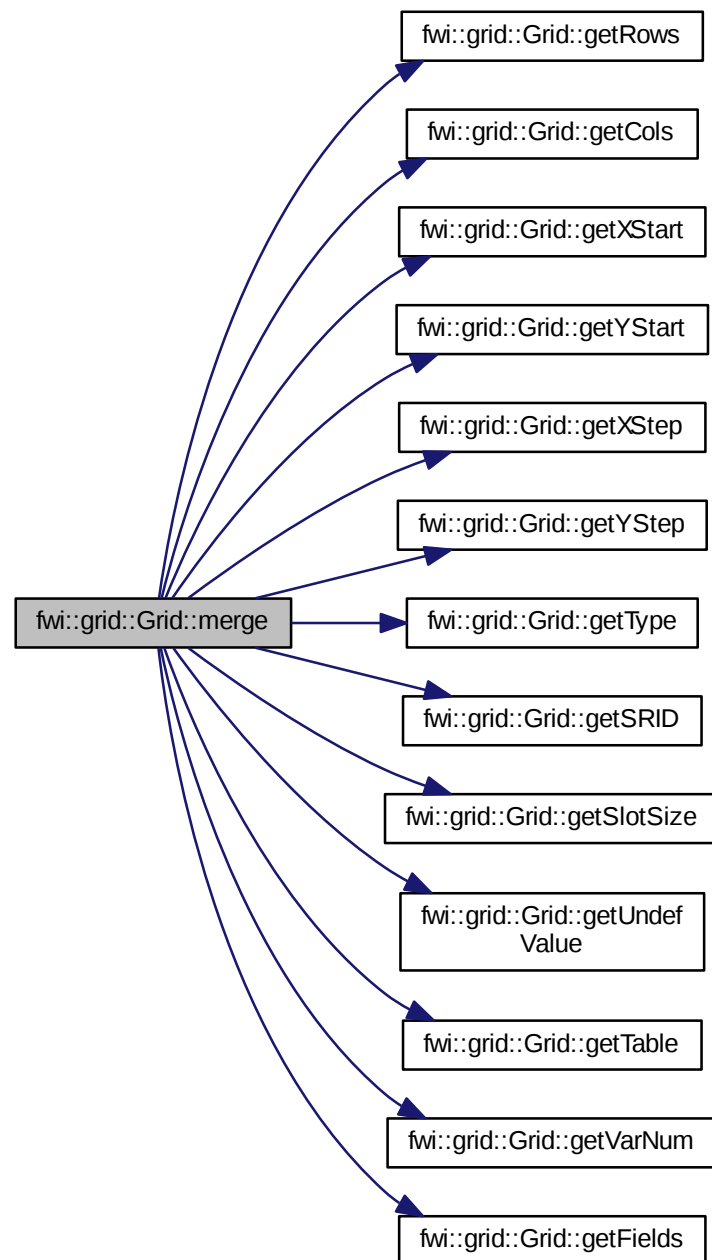
<i>other</i>	second grid
	<pre> merge can be done only if rows == other.getRows() && cols == other.getCols() and xStart == other.getXStart() && yStart == other.getYStart() and xStep == other.getXStep() && yStep == other.getYStep() and type == other.getType() && srid == other.getSRID() and slotSize == other.getSlotSize() && undefValue == other.getUndefValue() and table == other.getTable() </pre>

Returns

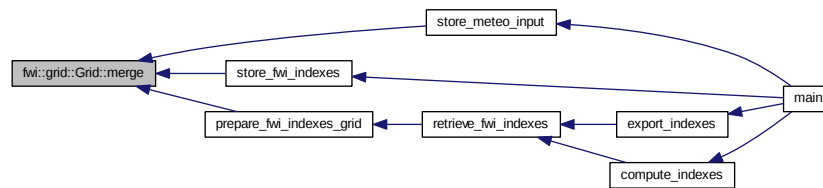
true on success else false

Definition at line 1652 of file Grid.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.35 `template<typename T> T & fwi::grid::Grid< T >::operator() (int i, int j, int k)`

grid element access helper

Parameters

<i>i</i>	row number
<i>j</i>	column number
<i>k</i>	plane number

Returns

grid element at row *i* column *j*

Definition at line 1447 of file Grid.h.

8.2.3.36 `template<typename T> Grid< T > & fwi::grid::Grid< T >::operator= (const Grid< T > & grid)`

grid assignement operator

Parameters

<i>grid</i>	grid to be assigned
-------------	---------------------

Returns

this grid after assignement

Definition at line 1455 of file Grid.h.

8.2.3.37 `template<typename T> bool fwi::grid::Grid< T >::read ()`

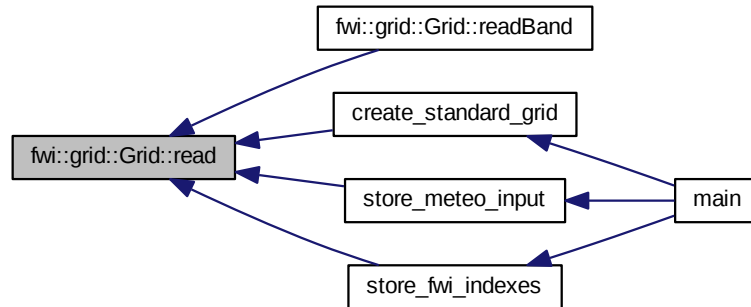
reads grid binary file

Returns

true on success else false

Definition at line 1776 of file Grid.h.

Here is the caller graph for this function:



8.2.3.38 `template<typename T> bool fwi::grid::Grid< T >::readBand (ifstream & in)` [protected]

reads data from a grid time band (stream must be opened in binary mode) not applicable to text streams

Parameters

<i>in</i>	input stream
-----------	--------------

Returns

true on success else false

Definition at line 1919 of file Grid.h.

Here is the call graph for this function:



8.2.3.39 `template<typename T> bool fwi::grid::Grid< T >::readBin (ifstream & in)` [protected]

reads grid data from binary file

Parameters

<i>in</i>	input stream
-----------	--------------

Returns

true on success else false

Definition at line 1864 of file Grid.h.

8.2.3.40 `template<typename T> bool fwi::grid::Grid< T >::readCtrl (ifstream & in)`

reads grid control file

Parameters

<i>in</i>	input stream
-----------	--------------

Returns

true on success else false

Definition at line 1770 of file Grid.h.

8.2.3.41 `template<typename T> bool fwi::grid::Grid< T >::readTxt (ifstream & in)` [protected]

reads grid data from text file

Parameters

<i>in</i>	input stream
-----------	--------------

Returns

true on success else false

Definition at line 1819 of file Grid.h.

8.2.3.42 `template<typename T> bool fwi::grid::Grid< T >::retrieve (PGconn * conn)`

retrieves grid from database

Parameters

<i>conn</i>	postgresql connection
-------------	-----------------------

Returns

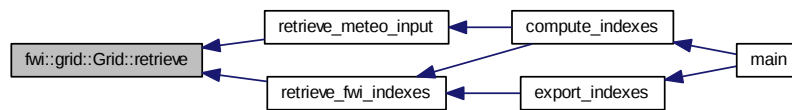
true on success else false

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 2352 of file Grid.h.

Here is the caller graph for this function:



8.2.3.43 `template<typename T> void fwi::grid::Grid< T >::setCols (int cols)`

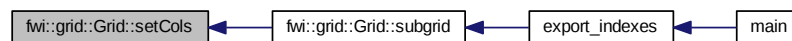
grid columns number setter

Parameters

<i>cols</i>	new columns number value
-------------	--------------------------

Definition at line 1274 of file Grid.h.

Here is the caller graph for this function:



8.2.3.44 `template<typename T> void fwi::grid::Grid< T >::setCtlPath (std::string filepath)`

grid ctl file path setter

Parameters

<i>filepath</i>	new ctl file path value
-----------------	-------------------------

Definition at line 1304 of file Grid.h.

8.2.3.45 `template<typename T> void fwi::grid::Grid< T >::setDate (std::string date)`

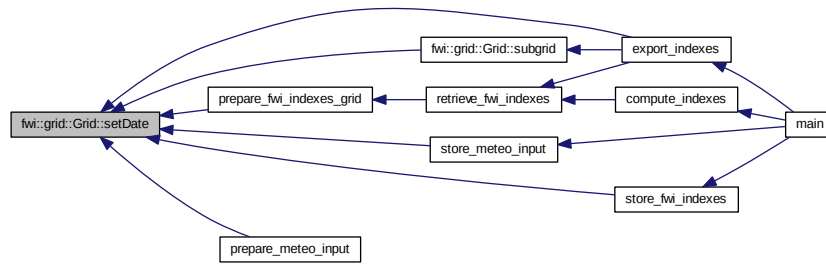
date setter

Parameters

<i>date</i>	date value expressed as YYYYMMDD
-------------	----------------------------------

Definition at line 1286 of file Grid.h.

Here is the caller graph for this function:



8.2.3.46 `template<typename T> void fwi::grid::Grid< T >::setDatPath (std::string filepath)`

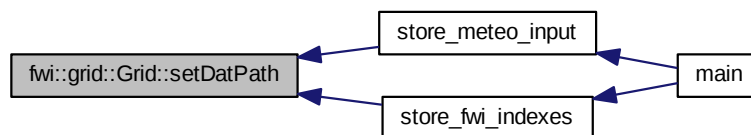
grid dat file path setter

Parameters

<i>filepath</i>	new dat file path value
-----------------	-------------------------

Definition at line 1310 of file Grid.h.

Here is the caller graph for this function:



8.2.3.47 `template<typename T> void fwi::grid::Grid< T >::setExportCtlPath (std::string filepath)`

grid export ctl file path setter

Parameters

<i>filepath</i>	new ctl export file path value
-----------------	--------------------------------

Definition at line 1316 of file Grid.h.

8.2.3.48 `template<typename T> void fwi::grid::Grid< T >::setExportDatPath (std::string filepath)`

grid export dat file path setter

Parameters

<i>filepath</i>	new dat export file path value
-----------------	--------------------------------

Definition at line 1322 of file Grid.h.

8.2.3.49 `template<typename T> void fwi::grid::Grid< T >::setFields (GridFields * fields)`

grid fields list setter

Parameters

<i>fields</i>	new grid fields list
---------------	----------------------

Definition at line 1437 of file Grid.h.

8.2.3.50 `template<typename T> void fwi::grid::Grid< T >::setFileNameDateOffset (int offset)`

grid file name date offset setter

Parameters

<i>offset</i>	file name date offset value
---------------	-----------------------------

Definition at line 1364 of file Grid.h.

8.2.3.51 `template<typename T> void fwi::grid::Grid< T >::setIOFormat (int format)`

grid I/O format setter

Parameters

<i>format</i>	new I/O format value
---------------	----------------------

Definition at line 1370 of file Grid.h.

8.2.3.52 `template<typename T> void fwi::grid::Grid< T >::setRows (int rows)`

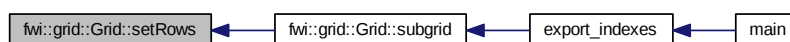
grid rows number setter

Parameters

<i>rows</i>	new rows number value
-------------	-----------------------

Definition at line 1268 of file Grid.h.

Here is the caller graph for this function:



8.2.3.53 `template<typename T> void fwi::grid::Grid< T >::setSlotSize (int slotSize)`

grid slot size setter

Parameters

<i>slotSize</i>	new grid slot size value
-----------------	--------------------------

Definition at line 1431 of file Grid.h.

8.2.3.54 `template<typename T> void fwi::grid::Grid< T >::setSRID (int srid)`

grid srid setter

Parameters

<i>srid</i>	new srid value
-------------	----------------

Definition at line 1419 of file Grid.h.

8.2.3.55 `template<typename T> void fwi::grid::Grid< T >::setStartTime (std::string t)`

grid start time setter

Parameters

<i>t</i>	grid new start time value in GrADS format
----------	---

See also

<http://www.iges.org/grads/>

Definition at line 1352 of file Grid.h.

8.2.3.56 `template<typename T> void fwi::grid::Grid< T >::setTable (std::string table)`

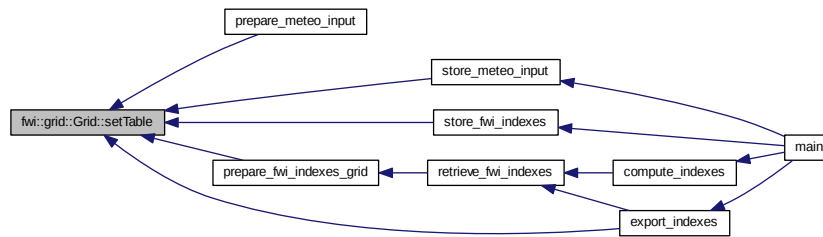
grid table name setter

Parameters

<i>table</i>	new table name value
--------------	----------------------

Definition at line 1280 of file Grid.h.

Here is the caller graph for this function:



8.2.3.57 `template<typename T> void fwi::grid::Grid< T >::setTimeBand (int band)`

grid time band setter

Parameters

<i>band</i>	new time band value
-------------	---------------------

Definition at line 1340 of file Grid.h.

8.2.3.58 `template<typename T> void fwi::grid::Grid< T >::setTimeBandsNumber (int bandsNumber)`

grid time bands number setter

Parameters

<i>bandsNumber</i>	new time bands number value
--------------------	-----------------------------

Definition at line 1346 of file Grid.h.

8.2.3.59 `template<typename T> void fwi::grid::Grid< T >::setTimeIncrement (std::string increment)`

grid time increment setter

Parameters

<i>increment</i>	new time increment value
------------------	--------------------------

Definition at line 1358 of file Grid.h.

8.2.3.60 `template<typename T> void fwi::grid::Grid< T >::setTitle (std::string title)`

grid title setter

Parameters

<i>title</i>	new title value
--------------	-----------------

Definition at line 1328 of file Grid.h.

8.2.3.61 `template<typename T> void fwi::grid::Grid< T >::setType (int type)`

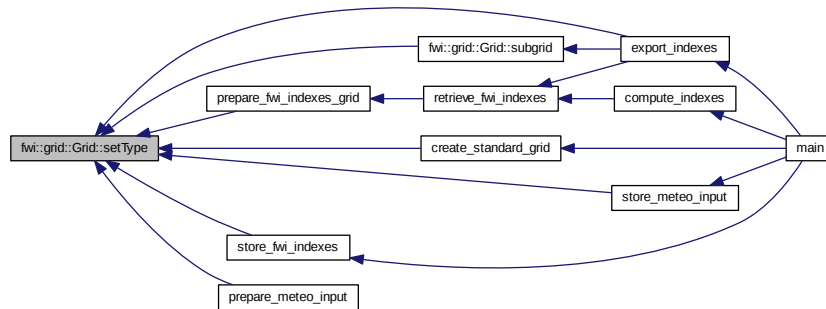
grid type setter

Parameters

<i>type</i>	new type value
-------------	----------------

Definition at line 1334 of file Grid.h.

Here is the caller graph for this function:

8.2.3.62 `template<typename T> void fwi::grid::Grid< T >::setUndefValue (float undefValue)`

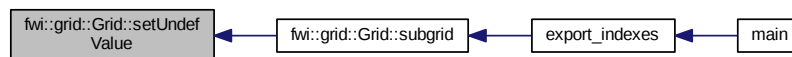
grid undefined value setter

Parameters

<i>undefValue</i>	new grid undefined value
-------------------	--------------------------

Definition at line 1425 of file Grid.h.

Here is the caller graph for this function:

8.2.3.63 `template<typename T> void fwi::grid::Grid< T >::setVarNum (int varnum)`

grid variables number setter

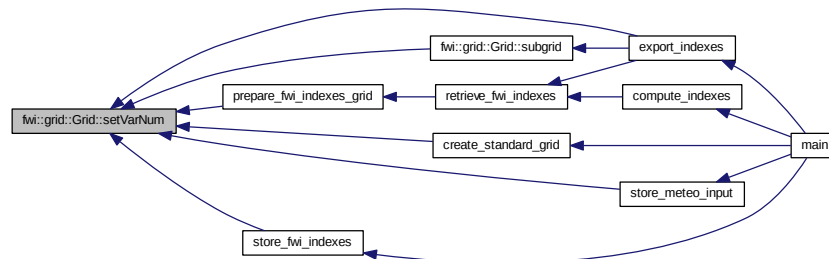
Implies grid resize. Elements are preserved only if the new value for `<i>varNum</i>` is greater than or equal to the old one.

Parameters

<i>varnum</i>	new variables number value
---------------	----------------------------

Definition at line 1412 of file Grid.h.

Here is the caller graph for this function:



8.2.3.64 `template<typename T> void fwi::grid::Grid< T >::setXDir (COORDINATE_DIRECTION xDir)`

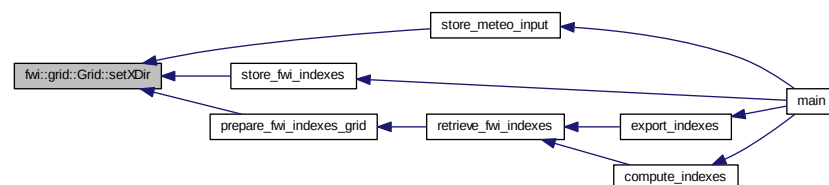
xDir setter

Parameters

<i>xDir</i>	new xDir value
-------------	----------------

Definition at line 1400 of file Grid.h.

Here is the caller graph for this function:



8.2.3.65 `template<typename T> void fwi::grid::Grid< T >::setXStart (float xstart)`

grid start x coordinate setter

Parameters

<i>xstart</i>	new xtsrat value
---------------	------------------

Definition at line 1376 of file Grid.h.

8.2.3.66 `template<typename T> void fwi::grid::Grid< T >::setXStep (float xstep)`

grid step in x direction setter

Parameters

<i>xstep</i>	new xstep value
--------------	-----------------

Definition at line 1382 of file Grid.h.

8.2.3.67 `template<typename T> void fwi::grid::Grid< T >::setYDir (COORDINATE_DIRECTION yDir)`

yDir setter

Parameters

<i>yDir</i>	new yDir value
-------------	----------------

Definition at line 1406 of file Grid.h.

8.2.3.68 `template<typename T> void fwi::grid::Grid< T >::setYStart (float ystart)`

grid start y coordinate setter

Parameters

<i>ystart</i>	new ystart value
---------------	------------------

Definition at line 1388 of file Grid.h.

8.2.3.69 `template<typename T> void fwi::grid::Grid< T >::setYStep (float ystep)`

grid step in y direction setter

Parameters

<i>ystep</i>	new ystep value
--------------	-----------------

Definition at line 1394 of file Grid.h.

8.2.3.70 `template<typename T> void fwi::grid::Grid< T >::skipBand (ifstream & in)` [protected]

skip the next timeband from reading

Parameters

<i>in</i>	input stream
-----------	--------------

Definition at line 1970 of file Grid.h.

8.2.3.71 `template<typename T> bool fwi::grid::Grid< T >::store (PGconn * conn)`

stores grid in database

Parameters

<i>conn</i>	postgresql connection
-------------	-----------------------

Returns

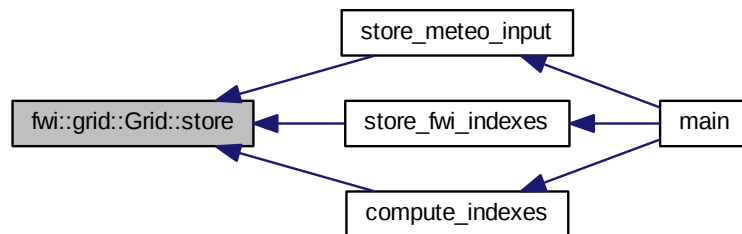
true on success else false

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 2193 of file Grid.h.

Here is the caller graph for this function:



8.2.3.72 `template<typename T> bool fwi::grid::Grid< T >::stored (PGconn * conn)`

verify if **this** is already stored in database

The existence check is based on the element number: if there aren't elements for `this` grid he grid is not present

Parameters

<i>conn</i>	postgresql connection
-------------	-----------------------

Returns

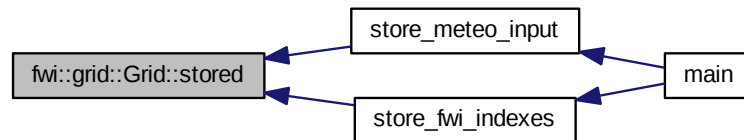
true if grid is present else false

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 2160 of file Grid.h.

Here is the caller graph for this function:



8.2.3.73 `template<typename T> bool fwi::grid::Grid< T >::subgrid (std::vector< std::string > fieldnames, Grid< T > & sg)`

Extracts a subgrid from *this* having fields contained in *fieldnames*

Parameters

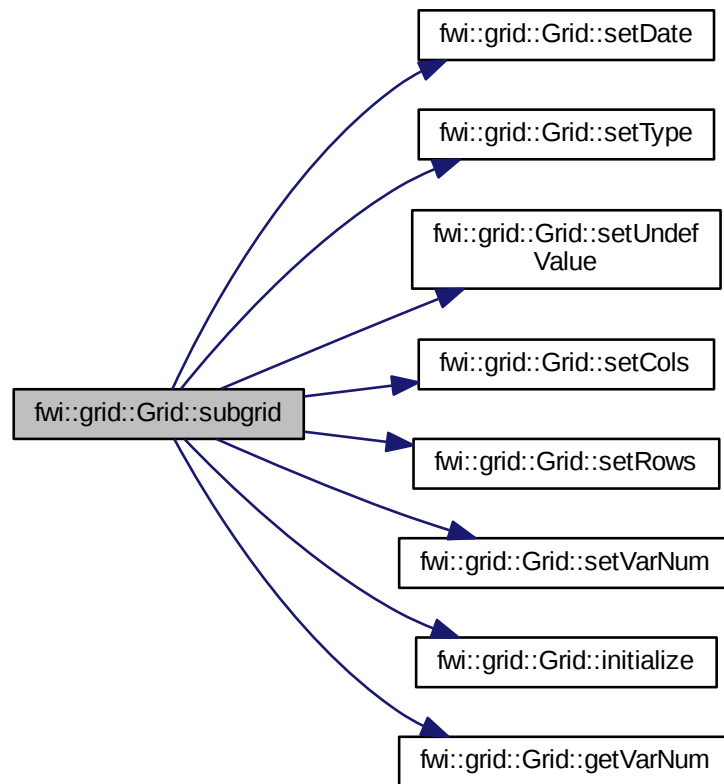
<i>fieldnames</i>	array containing the field names to extract from <i>this</i> grid.
<i>sg</i>	the computed subgrid

Returns

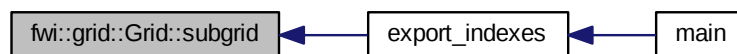
true on success else false

Definition at line 1707 of file Grid.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.3.74 `template<typename T> bool fwi::grid::Grid< T >::update (PGconn * conn)`

updates grid in database

Parameters

<i>conn</i>	postgresql connection
-------------	-----------------------

Returns

true on success else false

See also

postgresql documentation at <http://www.postgresql.org/>

Definition at line 2297 of file Grid.h.

8.2.3.75 template<typename T> bool fwi::grid::Grid< T >::write (ofstream & out)

writes grid binary file

Parameters

<i>out</i>	output stream
------------	---------------

Returns

true on success else false

Definition at line 2066 of file Grid.h.

Here is the call graph for this function:



Here is the caller graph for this function:

**8.2.3.76 template<typename T> bool fwi::grid::Grid< T >::writeCtrl (ofstream & out)**

write grid ctl file

Parameters

<i>out</i>	output stream
------------	---------------

Returns

true on success else false

Definition at line 1980 of file Grid.h.

Here is the caller graph for this function:



8.2.3.77 `template<typename T> bool fwi::grid::Grid< T >::writeTxt (ofstream & out, bool esri = false)`

writes grid text file

Parameters

<i>out</i>	output stream
<i>esri</i>	flag for ESRI grid format

Returns

true on success else false

Definition at line 2103 of file Grid.h.

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

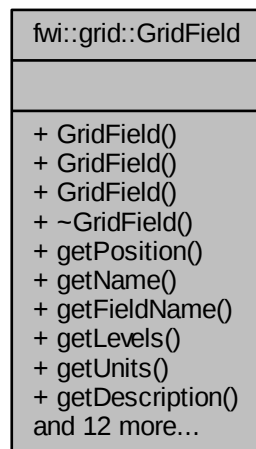
- [include/Grid.h](#)

8.3 fwi::grid::GridField Class Reference

grid field descriptor class

```
#include <GridField.h>
```

Collaboration diagram for fwi::grid::GridField:



Public Member Functions

- [GridField](#) ()
Standard constructor.
- [GridField](#) (int position, string name, string field_name, levels_t levels, int units, string description, GRID_VALUE_TYPE type=FLOAT)
Parameterized constructor.
- [GridField](#) (const [GridField](#) &field)
Copy constructor.
- virtual [~GridField](#) ()
Destructor.
- int [getPosition](#) () const
position getter
- string [getName](#) () const
name getter
- string [getFieldName](#) () const
field name getter
- levels_t [getLevels](#) () const
levels getter
- int [getUnits](#) () const
units getter
- string [getDescription](#) () const
description getter
- GRID_VALUE_TYPE [getType](#) () const
type getter
- void [setPosition](#) (int position)
position setter
- void [setName](#) (string name)
name setter

- void [setLevels](#) (levels_t levels)
levels setter
- void [setUnits](#) (int units)
units setter
- void [setDescription](#) (string description)
description setter
- void [setType](#) (GRID_VALUE_TYPE type)
type setter
- [GridField](#) & [operator=](#) ([GridField](#) &field)
assignment operator
- [GridField](#) & [operator=](#) ([GridField](#) *field)
assignment operator
- bool [operator==](#) ([GridField](#) &field)
equality operator
- bool [operator==](#) ([GridField](#) *field)
equality operator

Friends

- ostream & [operator<<](#) (ostream &stream, [GridField](#) gfd)
output stream operator
- ostream & [operator<<](#) (ostream &stream, [GridField](#) *gfd)
output stream operator
- istream & [operator>>](#) (istream &stream, GRID_VALUE_TYPE &t)
input stream operator
- istream & [operator>>](#) (istream &stream, [GridField](#) &gfd)
input stream operator

8.3.1 Detailed Description

grid field descriptor class

Definition at line 48 of file GridField.h.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 `fwi::grid::GridField::GridField (int position, string name, string field_name, levels_t levels, int units, string description, GRID_VALUE_TYPE type = GRID_FLOAT)`

Parameterized constructor.

Parameters

<i>position</i>	position in field list
<i>name</i>	field name
<i>field_name</i>	database field name
<i>levels</i>	levels vector
<i>units</i>	GrADS units tag
<i>description</i>	field description
<i>type</i>	field type

Definition at line 31 of file GridField.cpp.

8.3.3 Member Function Documentation

8.3.3.1 string fwi::grid::GridField::getDescription () const

description getter

Returns

description

Definition at line 81 of file GridField.cpp.

8.3.3.2 string fwi::grid::GridField::getFieldName () const

field name getter

Returns

field name

Definition at line 66 of file GridField.cpp.

8.3.3.3 levels_t fwi::grid::GridField::getLevels () const

levels getter

Returns

levels

Definition at line 71 of file GridField.cpp.

8.3.3.4 string fwi::grid::GridField::getName () const

name getter

Returns

field name

Definition at line 61 of file GridField.cpp.

8.3.3.5 int fwi::grid::GridField::getPosition () const

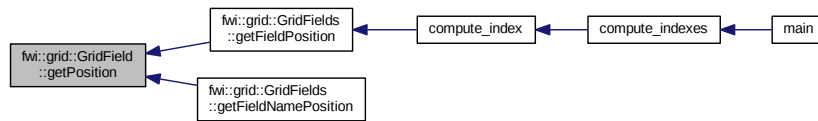
position getter

Returns

field position

Definition at line 56 of file GridField.cpp.

Here is the caller graph for this function:



8.3.3.6 GRID_VALUE_TYPE fwi::grid::GridField::getType () const

type getter

Returns

field type

Definition at line 86 of file GridField.cpp.

8.3.3.7 int fwi::grid::GridField::getUnits () const

units getter

Returns

units

Definition at line 76 of file GridField.cpp.

8.3.3.8 GridField & fwi::grid::GridField::operator= (GridField & field)

assignment operator

Parameters

<i>field</i>	assigned value
--------------	----------------

Returns

the changed object

Definition at line 130 of file GridField.cpp.

8.3.3.9 GridField & fwi::grid::GridField::operator= (GridField * field)

assignment operator

Parameters

<i>field</i>	object to be assigned
--------------	-----------------------

Returns

this after assignment

Definition at line 146 of file GridField.cpp.

8.3.3.10 bool fwi::grid::GridField::operator==(GridField & field)

equality operator

Parameters

<i>field</i>	object to test for equality
--------------	-----------------------------

Returns

true on equality else false

Definition at line 164 of file GridField.cpp.

8.3.3.11 bool fwi::grid::GridField::operator==(GridField * field)

equality operator

Parameters

<i>field</i>	object to test for equality
--------------	-----------------------------

Returns

true on equality else false

Definition at line 170 of file GridField.cpp.

8.3.3.12 void fwi::grid::GridField::setName (string name)

name setter

Parameters

<i>name</i>	new field name
-------------	----------------

Definition at line 96 of file GridField.cpp.

8.3.3.13 void fwi::grid::GridField::setPosition (int position)

position setter

Parameters

<i>position</i>	new field position
-----------------	--------------------

Definition at line 91 of file GridField.cpp.

Here is the caller graph for this function:



8.3.3.14 void fwi::grid::GridField::setType (GRID_VALUE_TYPE *type*)

type setter

Parameters

<i>type</i>	new field type
-------------	----------------

Definition at line 121 of file GridField.cpp.

Here is the caller graph for this function:



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

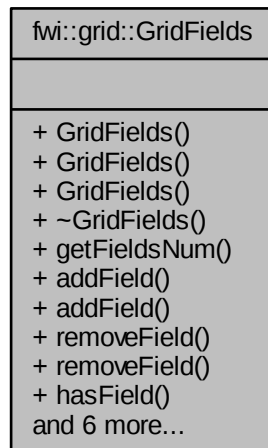
- [include/GridField.h](#)
- [src/GridField.cpp](#)

8.4 fwi::grid::GridFields Class Reference

Fields list class.

```
#include <GridField.h>
```


Collaboration diagram for fwi::grid::GridFields:



Public Member Functions

- [GridFields](#) ()
Standard constructor.
- [GridFields](#) (int fieldsNum)
Parameterized constructor.
- [GridFields](#) (int fieldsNum, GRID_VALUE_TYPE type)
Parameterized constructor.
- virtual [~GridFields](#) ()
Destructor.
- int [getFieldsNum](#) () const
fields number getter
- void [addField](#) ([GridField](#) *field)
adds a field to fields list
- void [addField](#) (int position, string name, string field_name, levels_t levels, int units, string description, GRID_VALUE_TYPE type=FLOAT)
adds a field to fields list by its parameters
- void [removeField](#) ([GridField](#) *field)
removes field from fields list
- void [removeField](#) (string name)
removes field by name
- bool [hasField](#) ([GridField](#) *field)
checks if field is in fields list
- bool [hasField](#) (string name)
checks if field with name is present in fields list
- [GridField](#) * [getFieldByName](#) (string &name)
Gets a [GridField](#) given its name.
- int [getFieldPosition](#) (string &name)

Gets field position given its name name the searched field name return The [GridField](#) position of field named with name. If field's name is not found (NOT_FOUND = -1) is returned.

- [GridField](#) * [getFieldByFieldName](#) (string &fname)

Gets a [GridField](#) given its field name.

- int [getFieldNamePosition](#) (string &fname)

Gets field position given its field name name the searched field field name return The [GridField](#) position of field named with fname. If field's fname is not found (NOT_FOUND = -1) is returned.

8.4.1 Detailed Description

Fields list class.

Definition at line 268 of file GridField.h.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 fwi::grid::GridFields::GridFields (int fieldsNum)

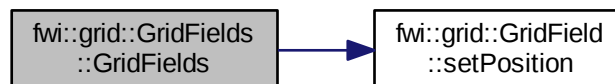
Parameterized constructor.

Parameters

<i>fieldsNum</i>	fields number
------------------	---------------

Definition at line 212 of file GridField.cpp.

Here is the call graph for this function:



8.4.2.2 fwi::grid::GridFields::GridFields (int fieldsNum, GRID_VALUE_TYPE type)

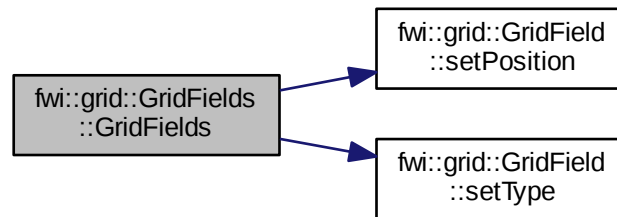
Parameterized constructor.

Parameters

<i>fieldsNum</i>	fields number
<i>type</i>	fields type

Definition at line 221 of file GridField.cpp.

Here is the call graph for this function:



8.4.3 Member Function Documentation

8.4.3.1 void fwi::grid::GridFields::addField (GridField * field)

adds a field to fields list

Parameters

<i>field</i>	new field
--------------	-----------

Definition at line 246 of file GridField.cpp.

8.4.3.2 void fwi::grid::GridFields::addField (int position, string name, string field_name, levels.t levels, int units, string description, GRID_VALUE_TYPE type = FLOAT)

adds a field to fields list by its parameters

Parameters

<i>position</i>	field's position
<i>name</i>	field's name
<i>field_name</i>	database field's name
<i>levels</i>	levels vector
<i>units</i>	field units
<i>description</i>	field description
<i>type</i>	field's type

Definition at line 251 of file GridField.cpp.

8.4.3.3 GridField * fwi::grid::GridFields::getFieldByFieldName (string & fname)

Gets a [GridField](#) given its field name.

Parameters

<i>fname</i>	the searched field name
--------------	-------------------------

Returns

The [GridField](#) which fieldname is *name* else NULL

Definition at line 348 of file GridField.cpp.

Here is the caller graph for this function:



8.4.3.4 `GridField * fwi::grid::GridFields::getFieldByName (string & name)`

Gets a [GridField](#) given its name.

Parameters

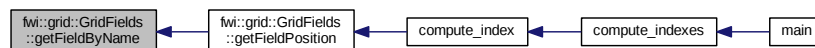
<i>name</i>	the searched field name
-------------	-------------------------

Returns

The [GridField](#) which name is *name* else NULL

Definition at line 300 of file GridField.cpp.

Here is the caller graph for this function:



8.4.3.5 `int fwi::grid::GridFields::getFieldsNum () const`

fields number getter

Returns

fields number

Definition at line 241 of file GridField.cpp.

8.4.3.6 `bool fwi::grid::GridFields::hasField (GridField * field)`

checks if field is in fields list

Returns

true if field is present else false

Definition at line 281 of file GridField.cpp.

8.4.3.7 bool fwi::grid::GridFields::hasField (string *name*)

checks if field with name is present in fields list

Parameters

<i>name</i>	name to check for
-------------	-------------------

Returns

true if field is present else false

Definition at line 287 of file GridField.cpp.

8.4.3.8 void fwi::grid::GridFields::removeField (GridField * *field*)

removes field from fields list

Parameters

<i>field</i>	field to be removed
--------------	---------------------

Definition at line 258 of file GridField.cpp.

8.4.3.9 void fwi::grid::GridFields::removeField (string *name*)

removes field by name

Parameters

<i>name</i>	name of field to be removed
-------------	-----------------------------

Definition at line 269 of file GridField.cpp.

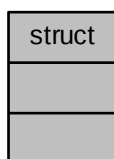
The documentation for this class was generated from the following files:

- [include/GridField.h](#)
- [src/GridField.cpp](#)

8.5 struct Struct Reference

defines floating-point values fixed-point obj.getType()generator policy

Collaboration diagram for struct:



8.5.1 Detailed Description

defines floating-point values fixed-point obj.getType()generator policy

See also

http://www.boost.org/doc/libs/1_41_0/libs/spirit/doc/html/spirit/karma/reference/number_number.html

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

- include/[ctlgen.hpp](#)

Chapter 9

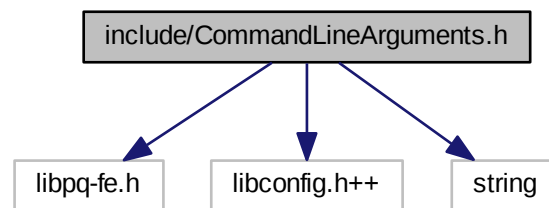
File Documentation

9.1 include/CommandLineArguments.h File Reference

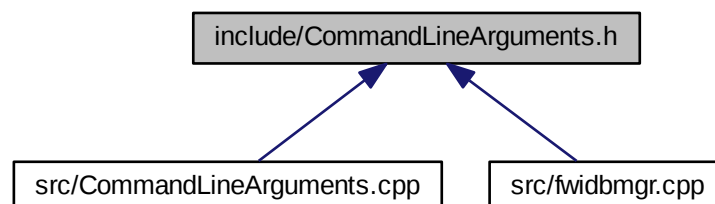
Command line arguments class.

```
#include <libpq-fe.h>
#include <libconfig.h++>
#include <string>
```

Include dependency graph for CommandLineArguments.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [fwi::CommandLineArguments](#)

Command line arguments class.

Namespaces

- namespace [fwi](#)

main fwidbmgr namespace

9.1.1 Detailed Description

Command line arguments class.

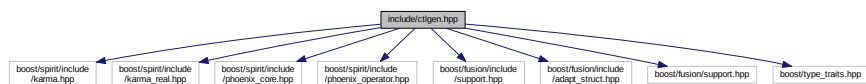
Definition in file [CommandLineArguments.h](#).

9.2 include/ctlgen.hpp File Reference

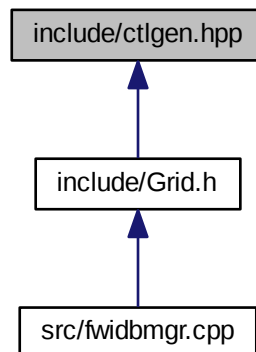
Grads control file generator using spirit.karma from boost libraries.

```
#include <boost/spirit/include/karma.hpp>
#include <boost/spirit/include/karma_real.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/fusion/include/support.hpp>
#include <boost/fusion/include/adapt_struct.hpp>
#include <boost/fusion/support.hpp>
#include <boost/type_traits.hpp>
```

Include dependency graph for ctlgen.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace `fwi`
main fwidbmgr namespace

Typedefs

- typedef `real_generator< float, fixed_policy< float > > fwi::generators::fixed_type`
floating-point fixed-point generator type

Variables

- fixed_type const `fwi::generators::fixedgen` = `fixed_type()`
fixed-point generator instance

9.2.1 Detailed Description

Grads control file generator using spirit.karma from boost libraries. CTL file generator This generator produces the .ctl file content about the coupled .dat file like for example:

```

DSET /home/meteo/programmi/interpolazione_statistica/oi_fwi/temp/20120111t2m-
_g.dat
TITLE 2m temperature
UNDEF -9999.000
XDEF 177 LINEAR 1436301.37500000 1500.000000000000
YDEF 174 LINEAR 4916704.50000000 1500.000000000000
ZDEF 1 LINEAR 1.000000 1.000000
TDEF 24 LINEAR 13:00Z11jan2012 1HR
VARS 3
  
```

```

xb 0 99 T background field
xa 0 99 T analysis field
xidi 0 99 integral data influence
ENDVARS

```

See also

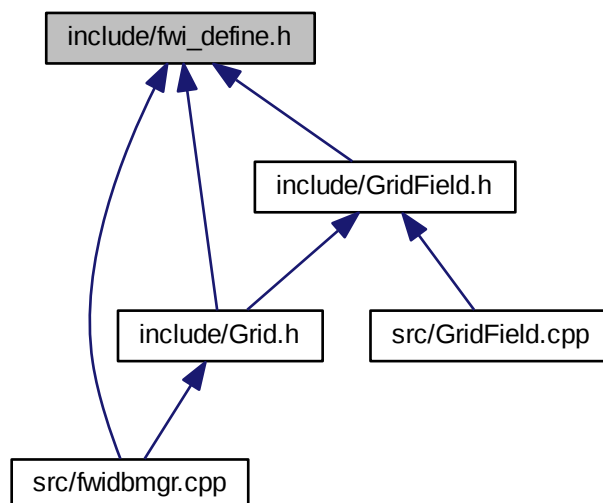
<http://www.boost.org/>

Definition in file [ctlgen.hpp](#).

9.3 include/fwi_define.h File Reference

defines for program

This graph shows which files directly or indirectly include this file:



Macros

- `#define GRD_DEFAULT_UNDEF_VALUE -9999.0`
default grid undefined value
- `#define GRD_COLS 177`
default grid columns number
- `#define GRD_ROWS 174`
default grid rows number
- `#define GRD_X_START 1436301.375`
default grid start point x coordinate
- `#define GRD_Y_START 4916704.500`
default grid start point y coordinate

- #define [GRD_X_STEP](#) 1500.0
default grid step in x direction expressed in meters
- #define [GRD_Y_STEP](#) 1500.0
default grid step in y direction expressed in meters
- #define [GRD_DEFAULT_VARNUM](#) 1
default grid variables number
- #define [TOPOGRAPHY_FIELDSNUM](#) 6
Topography file fields number.
- #define [TEMPERATURE_FIELDSNUM](#) 3
Temperature dat file fields number.
- #define [HUMIDITY_FIELDSNUM](#) 6
Humidity dat file fields number.
- #define [WINDSPEED_FIELDSNUM](#) 13
Windspeed dat file fields number.
- #define [RAIN_FIELDSNUM](#) 3
Rain dat file fields number.
- #define [GEOGRID](#) 0
Georeferenced grid.
- #define [METEO_INPUT](#) 1
Grid containing meteo data.
- #define [FWI_INDEXES](#) 2
Grid containing FWI indexes data.
- #define [NUMERIC](#) 3
Grid containing generic data.
- #define [GRD_FORMAT_TEXT](#) 0
Grid data are stored as ascii text.
- #define [GRD_FORMAT_BINARY](#) 1
Grid data are stored in binary format.
- #define [METEO_PARAM_TEMPERATURE](#) "temperature"
Meteo parameter: temperature.
- #define [METEO_PARAM_HUMIDITY](#) "humidity"
Meteo parameter: humidity.
- #define [METEO_PARAM_WINDSPEED](#) "windspeed"
Meteo parameter: wind speed.
- #define [METEO_PARAM_RAIN](#) "rain"
Meteo parameter: rain.
- #define [ACTION_CREATE](#) "create"
Program switch to create empty database.
- #define [ACTION_CREATE_STD_GRID](#) "createstdgrid"
Program switch to create the standard grid (177x174)
- #define [ACTION_IN](#) "in"
Program switch to store the needed input data in database.
- #define [ACTION_OUT](#) "out"
Program switch to store FWI indexes computation results in database.
- #define [ACTION_OUT_IMG](#) "outimg"
Program switch to store summary images in database.
- #define [ACTION_EXPORT_INDEXES](#) "exportidx"
Program switch to export FWI indexes as GrADS files.
- #define [ACTION_COMPUTE_INDEXES](#) "computeidx"
Program switch to compute indexes.
- #define [ACTION_COMPUTE_INDEXES_24](#) "computeidx24"

- Program switch to compute indexes over 24 time slots.*

 - `#define ACTION_TEST "test"`

Used only for testing.
- `#define GIS_DEFAULT_SRID 3003`

Default used SRID (Gauss-Boaga - Monte Mario 1 - Roma 40)
- `#define GRD_DEFAULT_TABLE "grid"`

Default grid table name.
- `#define GRD_METEO_INPUT_TABLE "meteo_input"`

Default meteo input tables name.
- `#define GRD_FWI_INDEXES_TABLE "fwi_indexes"`

Default FWI indexes tables name.
- `#define TAG_DATE "<<date>>"`

Date tag used in fwidbmgr.conf configuration file.
- `#define NOT_FOUND -1`

Not found value after serch.
- `#define SUCCESS 1`

successful operation return value
- `#define FAILURE 0`

unsuccessfull operation return value

9.3.1 Detailed Description

defines for program

Definition in file [fwi_define.h](#).

9.3.2 Macro Definition Documentation

9.3.2.1 `#define ACTION_COMPUTE_INDEXES "computeidx"`

Program switch to compute indexes.

This is an experimental feature At the moment this action computes only these three new indexes

- angstroem
- fmi
- sharples

Definition at line 197 of file [fwi_define.h](#).

9.3.2.2 `#define ACTION_COMPUTE_INDEXES_24 "computeidx24"`

Program switch to compute indexes over 24 time slots.

This is an experimental feature At the moment this action computes only these three new indexes

Definition at line 207 of file [fwi_define.h](#).

9.3.2.3 `#define FAILURE 0`

unsuccessfull operation return value

FAILURE

Definition at line 265 of file [fwi_define.h](#).

9.3.2.4 #define GRD_X_START 1436301.375

default grid start point x coordinate

Coordinates are expressed using Monte Mario 1 SRS

See also

postgis documentation at <http://postgis.refrations.net/>
<http://www.epsg.org/>

Definition at line 34 of file fwi_define.h.

9.3.2.5 #define GRD_Y_START 4916704.500

default grid start point y coordinate

Coordinates are expressed using Monte Mario 1 SRS

See also

postgis documentation at <http://postgis.refrations.net/>
<http://www.epsg.org/>

Definition at line 44 of file fwi_define.h.

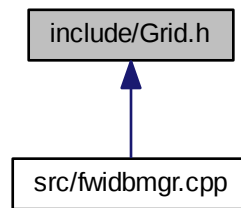
9.4 include/Grid.h File Reference

2-dimensions grid class

```
#include "fwi_define.h"
#include "GridField.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <list>
#include <map>
#include "boost/multi_array.hpp"
#include "boost/date_time/posix_time/posix_time.hpp"
#include <boost/date_time/date_facet.hpp>
#include <cassert>
#include <libpq-fe.h>
#include <ctlgen.hpp>
Include dependency graph for Grid.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [fwi::grid::Grid< T >](#)
3-dimensional grid

Namespaces

- namespace [fwi](#)
main fwidbmgr namespace

Typedefs

- typedef Grid< Point * > **fwi::grid::PointGrid_t**
standard grid with 1 Point variable*
- typedef Grid< float > **fwi::grid::Float1Grid_t**
standard grid with 1 float variable
- typedef Grid< int > **fwi::grid::Int1Grid_t**
standard grid with 1 int variable

Variables

- log4cxx::LoggerPtr [logger](#)
Application logger.

9.4.1 Detailed Description

2-dimensions grid class

Definition in file [Grid.h](#).

9.4.2 Variable Documentation

9.4.2.1 log4cxx::LoggerPtr logger

Application logger.

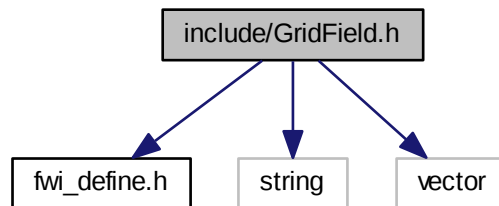
@see log4cxx documentation at <http://logging.apache.org/log4cxx/>

9.5 include/GridField.h File Reference

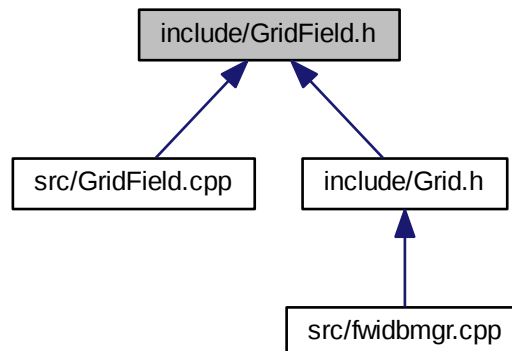
Grid field description class.

```
#include "fwi_define.h"  
#include <string>  
#include <vector>
```

Include dependency graph for GridField.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `fwi::grid::GridField`
grid field descriptor class
- class `fwi::grid::GridFields`
Fields list class.

Namespaces

- namespace `fwi`
main fwidbmgr namespace

Typedefs

- typedef vector< int > **fwi::grid::levels_t**
GrADS var levels.
- typedef vector< GridField * > **fwi::grid::fields_t**
shortcut for list<GridField>*
- typedef vector< GridField * >::iterator **fwi::grid::fields_iterator_t**
shortcut for list<GridField>::iterator*

Enumerations

- enum **GRID_VALUE_TYPE** { **fwi::grid::INTEGER** = 0, **fwi::grid::FLOAT** = 1, **fwi::grid::STRING** = 2, **fwi::grid::POINT** = 3 }
- value type to be stored in grid*

9.5.1 Detailed Description

Grid field description class.

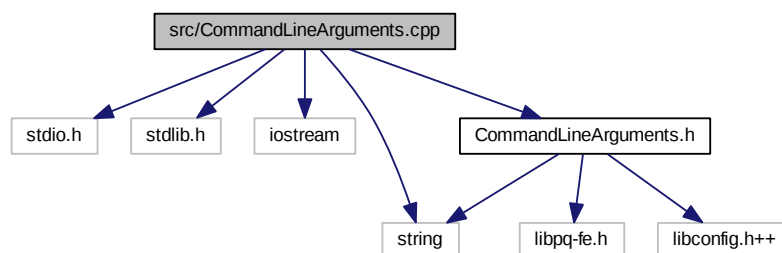
Definition in file [GridField.h](#).

9.6 src/CommandLineArguments.cpp File Reference

command line argument class implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include "CommandLineArguments.h"
```

Include dependency graph for CommandLineArguments.cpp:



Namespaces

- namespace [fwi](#)
main fwidbmgr namespace

9.6.1 Detailed Description

command line argument class implementation

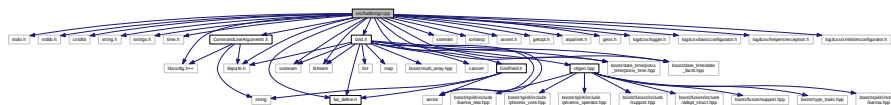
Definition in file [CommandLineArguments.cpp](#).

9.7 src/fwidbmgr.cpp File Reference

The main program file.

```
#include <stdio.h>
#include <stdlib.h>
#include <cstdlib>
#include <string.h>
#include <strings.h>
#include <time.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <vector>
#include <assert.h>
#include <getopt.h>
#include <arpa/inet.h>
#include <libconfig.h++>
#include <libpq-fe.h>
#include <geos.h>
#include "boost/date_time/posix_time/posix_time.hpp"
#include <boost/date_time/date_facet.hpp>
#include "log4cxx/logger.h"
#include "log4cxx/basicconfigurator.h"
#include "log4cxx/helpers/exception.h"
#include <log4cxx/xml/domconfigurator.h>
#include <fwi_define.h>
#include <CommandLineArguments.h>
#include <Grid.h>
```

Include dependency graph for fwidbmgr.cpp:



Functions

- `std::string getProgramHome ()`
Reads environment variable FWIDBMGR_HOME.
- `struct tm parseDate (std::string dt)`
Parses a string for a date value.
- `void usage ()`
helper function for usage display
- `bool process_cmd_line (int argc, char **argv, CommandLineArguments *args)`
Process command line parameters.
- `bool getSqlFiles (std::vector< std::string > &files)`

- gets sql files paths from configuration*
- `std::string loadQueryFromFile (std::string filepath)`
loads in memory a file content
- `bool execute (std::string &query)`
executes query command
- `bool create_database ()`
creates database structure based on configuration contents
- `bool fill_database ()`
fills empty database structure with data (ex. spatial ref systems from postgis)
- `bool prepare_meteo_input (std::string date)`
creates skeleton structure for meteo input grid referring to date
- `bool fill_nometeo_points ()`
fills grid table with no mete point flags
- `bool import_regions ()`
Imports regions polygons in database from file.
- `bool import_provinces ()`
Imports provinces polygons in database from file.
- `bool create_standard_grid ()`
create a standard 174 row x 177 columns grid
- `bool delete_meteo_input (CommandLineArguments &args)`
deletes meteo input grid for date in args from database
- `bool store_meteo_input (CommandLineArguments &args)`
stores meteo input grid for date in args reading from files
- `bool retrieve_meteo_input (CommandLineArguments &args, Grid< float > *res)`
retrieves meteo input grid for date in args reading from database
- `bool delete_fwi_indexes (CommandLineArguments &args)`
deletes fwi indexes grid for date in args from database
- `bool store_fwi_indexes (CommandLineArguments &args)`
stores fwi indexes grid for date in args reading from files
- `bool prepare_fwi_indexes_grid (CommandLineArguments &args, Grid< float > *res)`
prepares and configure fwi indexes grid
- `bool retrieve_fwi_indexes (CommandLineArguments &args, Grid< float > *res)`
retrieves fwi indexes grid for date in args reading from database
- `bool getFileBytea (std::string file, char **buffer, int &size)`
Reads file, returns file contents in buffer and file size in size.
- `bool delete_images (CommandLineArguments &args)`
deletes fwi indexes grid for date in args from database
- `bool store_images (CommandLineArguments &args)`
stores fwi images for date in args reading from disk.
- `bool retrieve_images (CommandLineArguments &args)`
retrieves fwi images for date in args reading from database
- `bool export_indexes (CommandLineArguments &args)`
exports fwi indexes grid for date in args from database
- `bool load_computation_indexes (std::vector< std::string > &indexes)`
Loads from configuration the list of index names to be computed.
- `bool compute_index (std::string &index_name, Grid< float > *indexes, Grid< float > *meteo)`
Compute a single index given its name.
- `bool compute_indexes (CommandLineArguments &args)`
Computes the new three indexes angstroem, fmi and sharples as an experimental feature. Requires the previous import of the standard indexes in database.
- `int main (int argc, char **argv)`
main function

9.7.1 Detailed Description

The main program file.

Definition in file [fwidbmgr.cpp](#).

9.7.2 Function Documentation

9.7.2.1 `bool compute_index (std::string & index_name, Grid< float > * indexes, Grid< float > * meteo)`

Compute a single index given its name.

Parameters

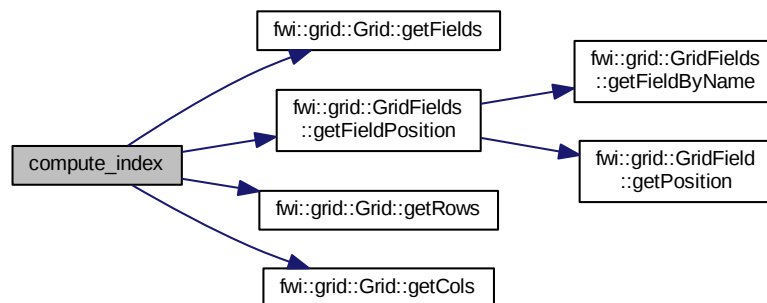
<i>index_name</i>	the index name
<i>indexes</i>	the already loaded indexes grid
<i>meteo</i>	the already loaded meteo input grid

Returns

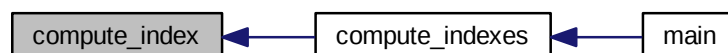
true on success else false

Definition at line 3317 of file `fwidbmgr.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.2 bool compute_indexes (CommandLineArguments & args)

Computes the new three indexes angstroem, fmi and sharples as an experimental feature. Requires the previous import of the standard indexes in database.

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

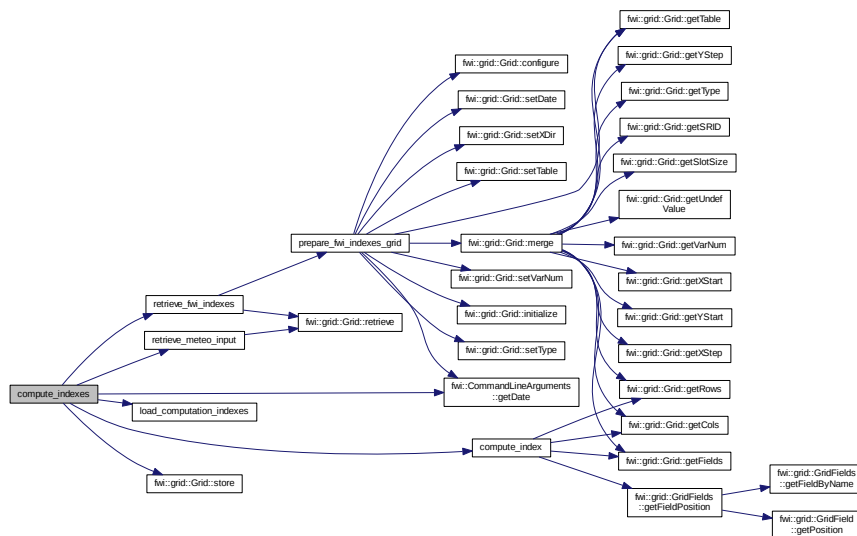
true on success else false

See also

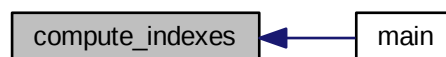
CommandLineArguments

Definition at line 3446 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.3 bool create_database ()

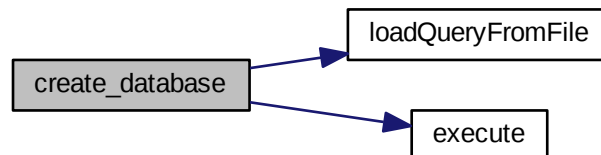
creates database structure based on configuration contents

Returns

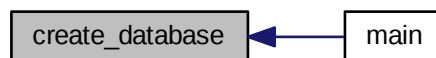
true on success else false

Definition at line 590 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**9.7.2.4 bool create_standard_grid ()**

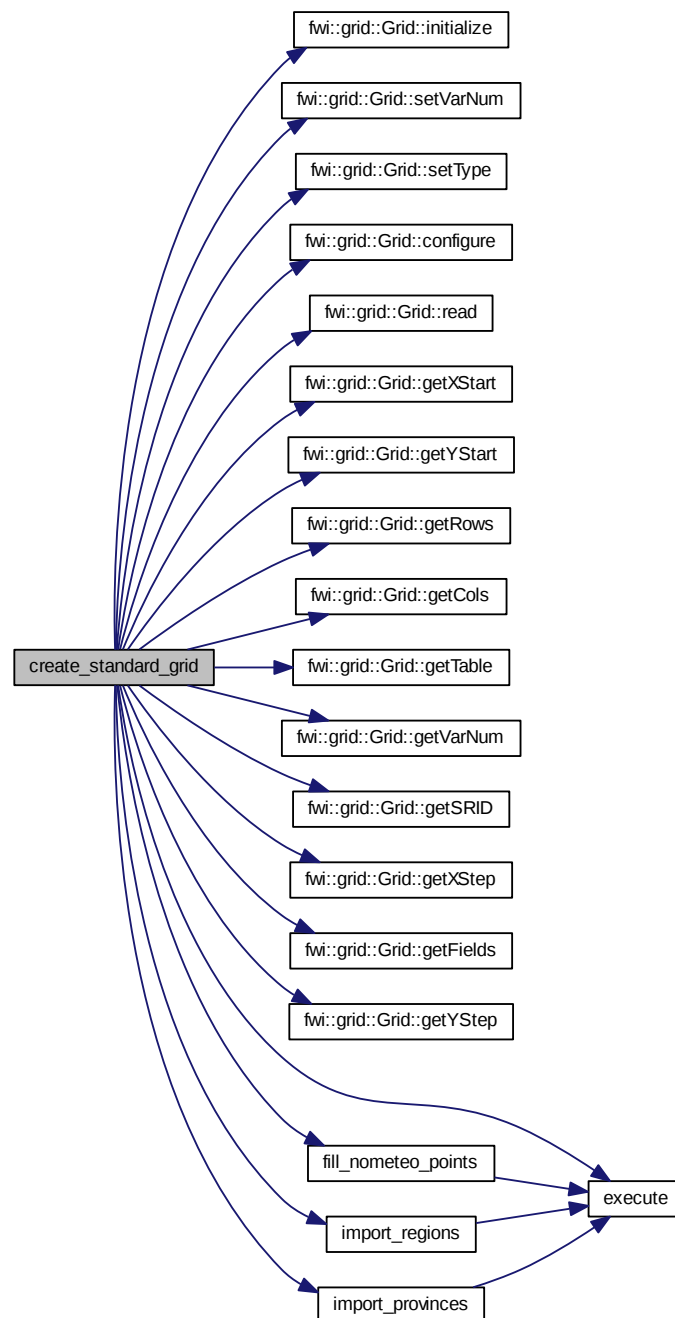
create a standard 174 row x 177 columns grid

Returns

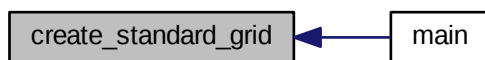
true on success else false

Definition at line 1083 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.5 bool delete_fwi_indexes (CommandLineArguments & args)

deletes fwi indexes grid for date in *args* from database

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

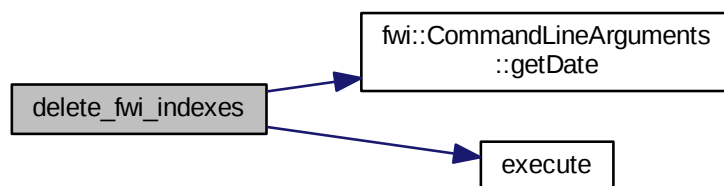
true on success else false

See also

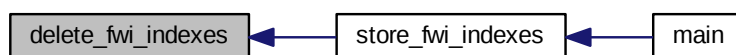
CommandLineArguments

Definition at line 1644 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.6 bool delete_images (CommandLineArguments & args)

deletes fwi indexes grid for date in *args* from database

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

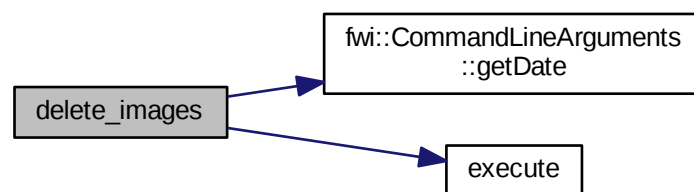
true on success else false

See also

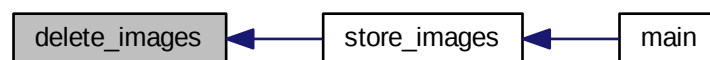
CommandLineArguments

Definition at line 2539 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.7 bool delete_meteo_input (CommandLineArguments & args)

deletes meteo input grid for date in *args* from database

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

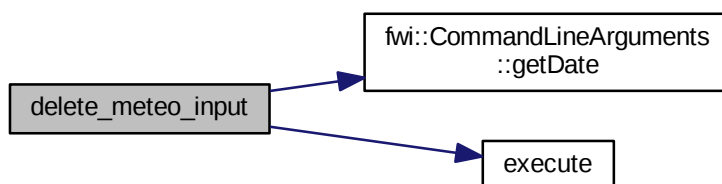
true on success else false

See also

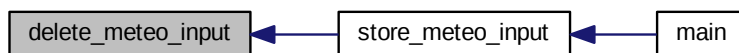
CommandLineArguments

Definition at line 1248 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**9.7.2.8 bool execute (std::string & query)**

executes query command

Parameters

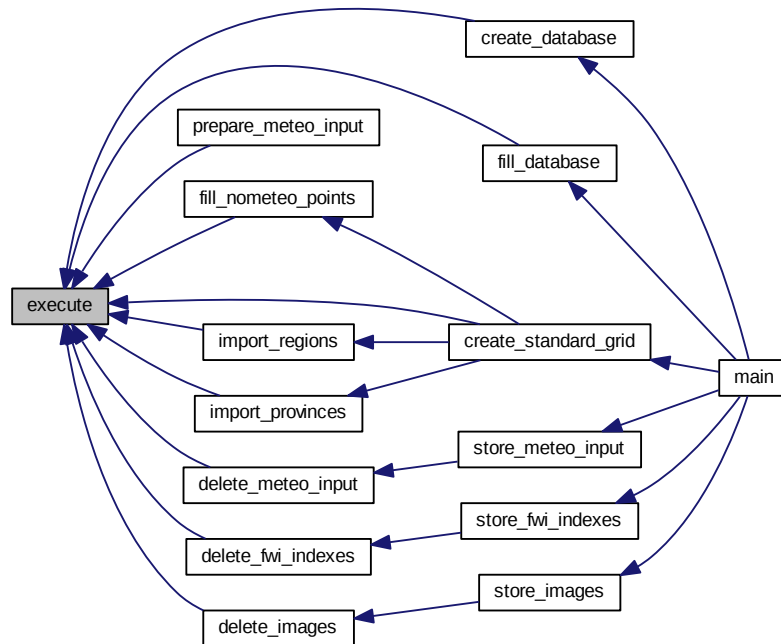
<i>query</i>	sql commands to be executed
--------------	-----------------------------

Returns

true on success else false

Definition at line 562 of file fwidbmgr.cpp.

Here is the caller graph for this function:



9.7.2.9 bool export_indexes (CommandLineArguments & args)

exports fwi indexes grid for date in *args* from database

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

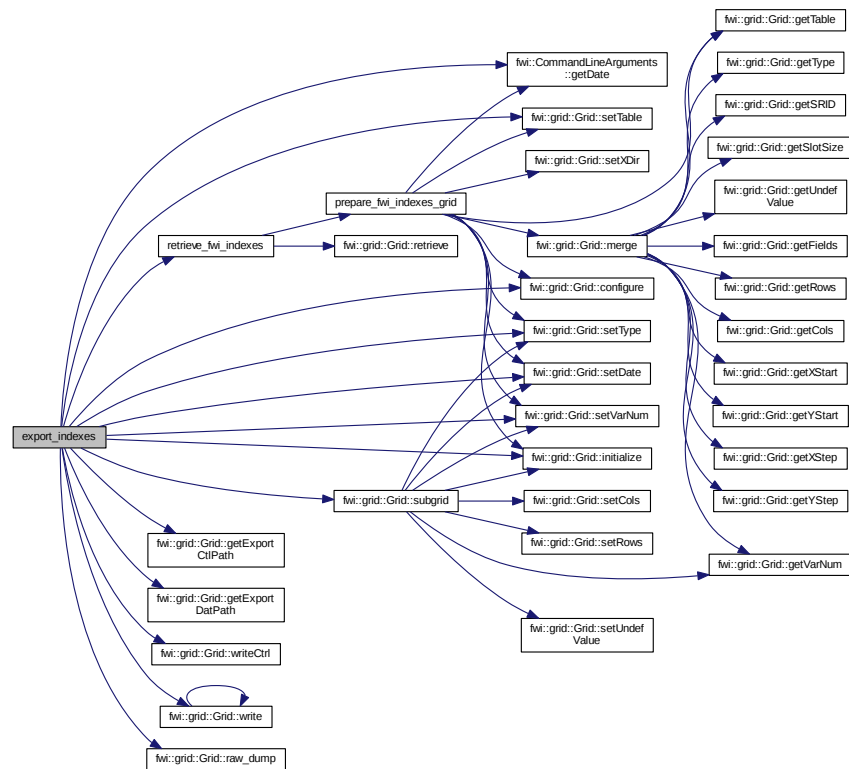
true on success else false

See also

CommandLineArguments

Definition at line 2775 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.10 bool fill_database ()

fills empty database structure with data (ex. spatial ref systems from postgis)

Returns

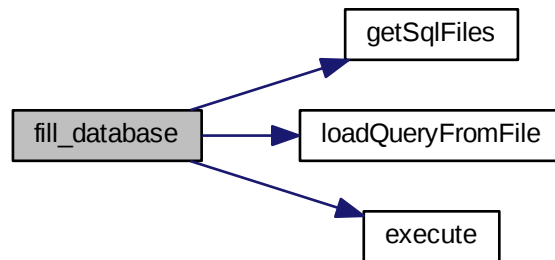
true on success else false

See also

libconfig++ documentation at <http://www.hyperrealm.com/libconfig/>

Definition at line 645 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.11 bool fill_nometeo_points ()

fills grid table with no mete point flags

See also

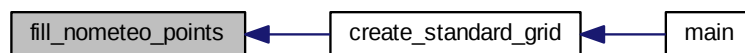
Grid

Definition at line 724 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.12 `bool getFileBytea (std::string file, char ** buffer, int & size)`

Reads file, returns file contents in buffer and file size in size.

Parameters

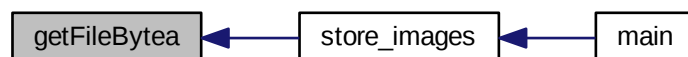
<i>file</i>	file absolute path
<i>buffer</i>	file contents buffer
<i>size</i>	file size

Returns

true on success else false

Definition at line 2510 of file fwidbmgr.cpp.

Here is the caller graph for this function:



9.7.2.13 `string getProgramHome ()`

Reads environment variable FWIDBMGR_HOME.

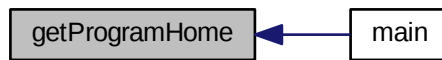
This environment variable has to be defined in order to run fwidbmgr

Returns

the path pointed by FWIDBMGR_HOME or a standard path

Definition at line 261 of file fwidbmgr.cpp.

Here is the caller graph for this function:

**9.7.2.14 bool getSqlFiles (std::vector< std::string > & files)**

gets sql files paths from configuration

Parameters

<i>files</i>	string vector containing files paths
--------------	--------------------------------------

Returns

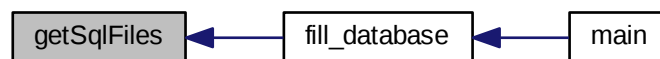
true on success else false

See also

libconfig++ documentation at <http://www.hyperrealm.com/libconfig/>

Definition at line 499 of file fwidbmgr.cpp.

Here is the caller graph for this function:

**9.7.2.15 bool import_provinces ()**

Imports provinces polygons in database from file.

Returns

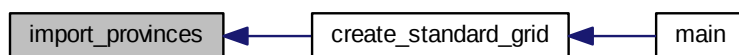
true on success else false

Definition at line 927 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**9.7.2.16 bool import_regions ()**

Imports regions polygons in database from file.

Returns

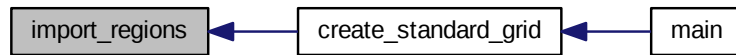
true on success else false

Definition at line 803 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.17 `bool load_computation_indexes (std::vector< std::string > & indexes)`

Loads from configuration the list of index names to be computed.

Parameters

<i>indexes</i>	the list of index names
----------------	-------------------------

Returns

true on success else false

Definition at line 3276 of file `fwidbmgr.cpp`.

Here is the caller graph for this function:



9.7.2.18 `string loadQueryFromFile (std::string filepath)`

loads in memory a file content

Parameters

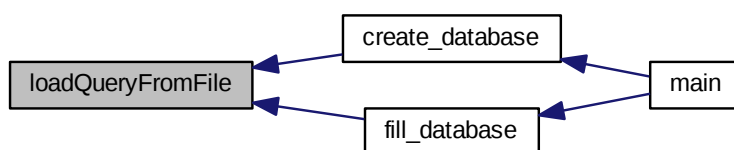
<i>filepath</i>	complete file path
-----------------	--------------------

Returns

the file content

Definition at line 545 of file fwidbmgr.cpp.

Here is the caller graph for this function:



9.7.2.19 `int main (int argc, char ** argv)`

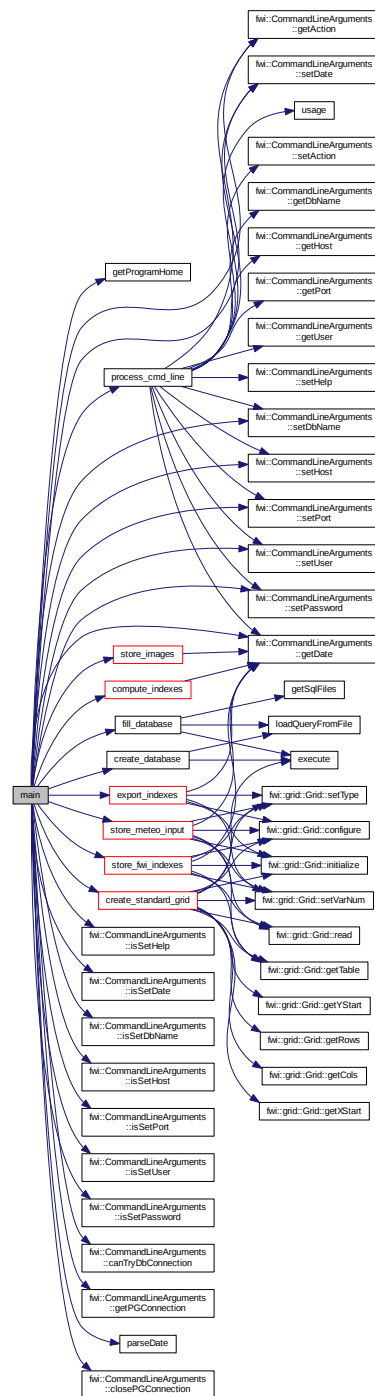
main function

See also

[usage\(\)](#)

Definition at line 4112 of file fwidbmgr.cpp.

Here is the call graph for this function:



9.7.2.20 struct tm parseDate (std::string dt) [read]

Parses a string for a date value.

Parameters

<i>dt</i>	string to be parsed
-----------	---------------------

Returns

struct tm filled with parsed values

Definition at line 274 of file fwidbmgr.cpp.

Here is the caller graph for this function:



9.7.2.21 `bool prepare_fwi_indexes_grid (CommandLineArguments & args, Grid< float > * res)`

prepares and configure fwi indexes grid

Parameters

<i>args</i>	command line arguments class
<i>res</i>	resulting grid

Returns

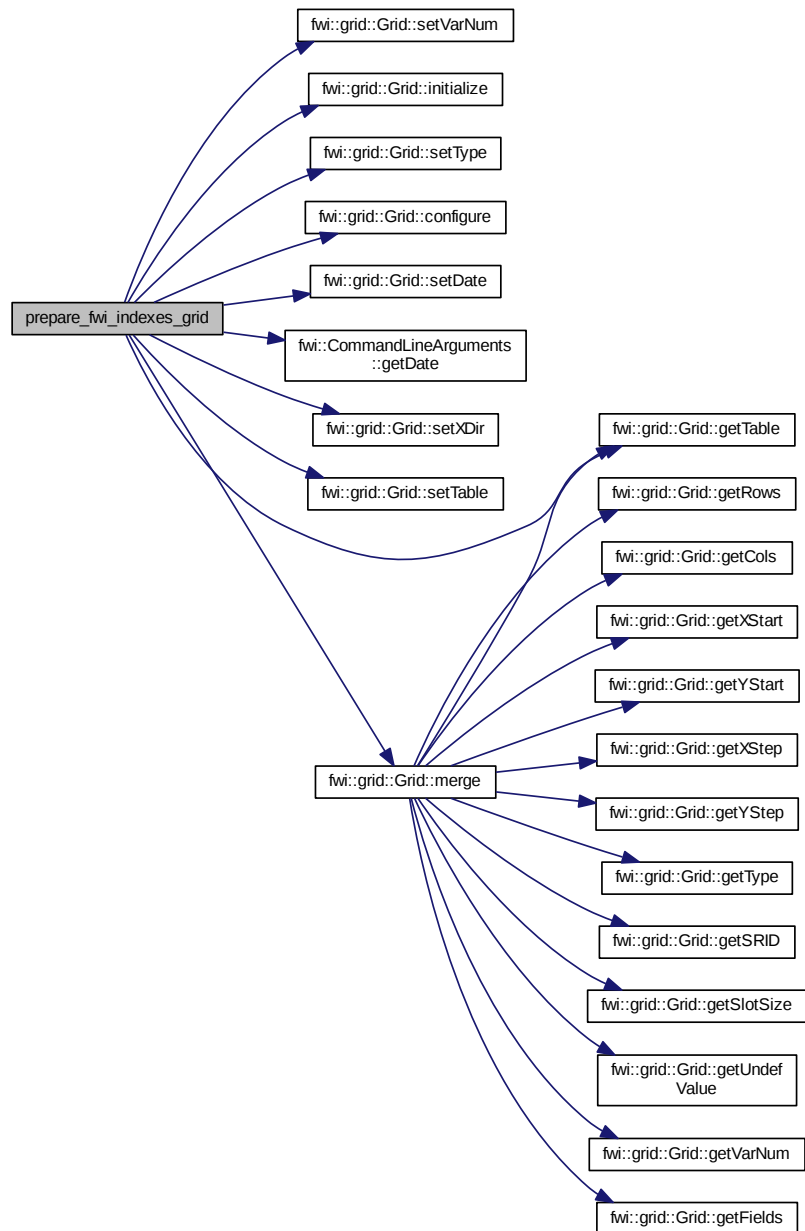
true on success else false

See also

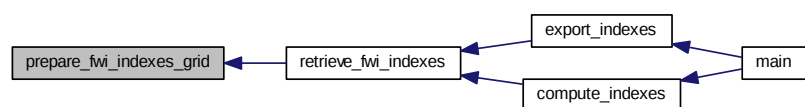
CommandLineArguments
Grid

Definition at line 1970 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.22 bool prepare_meteo_input (std::string *date*)

creates skeleton structure for meteo input grid referring to date

Parameters

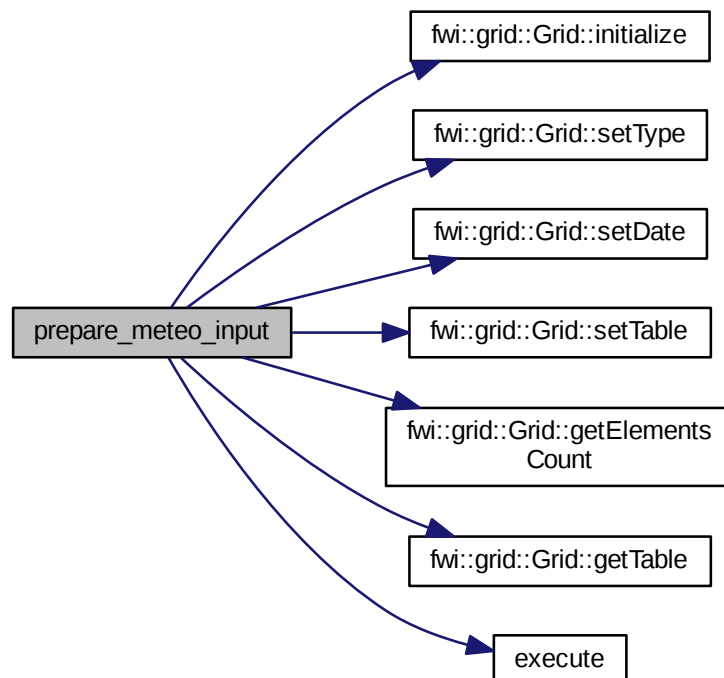
<i>date</i>	referring date as YYYYMMDD
-------------	----------------------------

Returns

true on success else false

Definition at line 691 of file fwidbmgr.cpp.

Here is the call graph for this function:



9.7.2.23 bool process_cmd_line (int *argc*, char ** *argv*, CommandLineArguments * *args*)

Process command line parameters.

Parameters

<i>argc</i>	Number of command line parameters
<i>argv</i>	array of string parameters
<i>args</i>	command line arguments

Returns

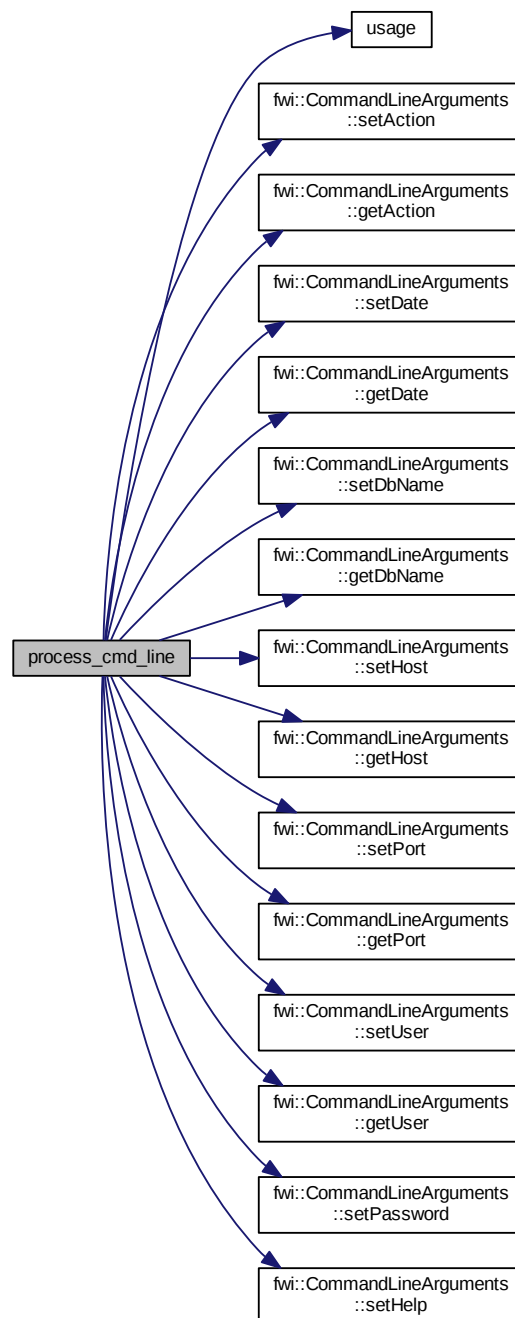
true on success else false

See also

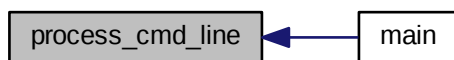
CommandLineArguments

Definition at line 390 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.24 bool retrieve_fwi_indexes (CommandLineArguments & args, Grid< float > * res)

retrieves fwi indexes grid for date in *args* reading from database

Parameters

<i>args</i>	command line arguments class
<i>res</i>	resulting grid

Returns

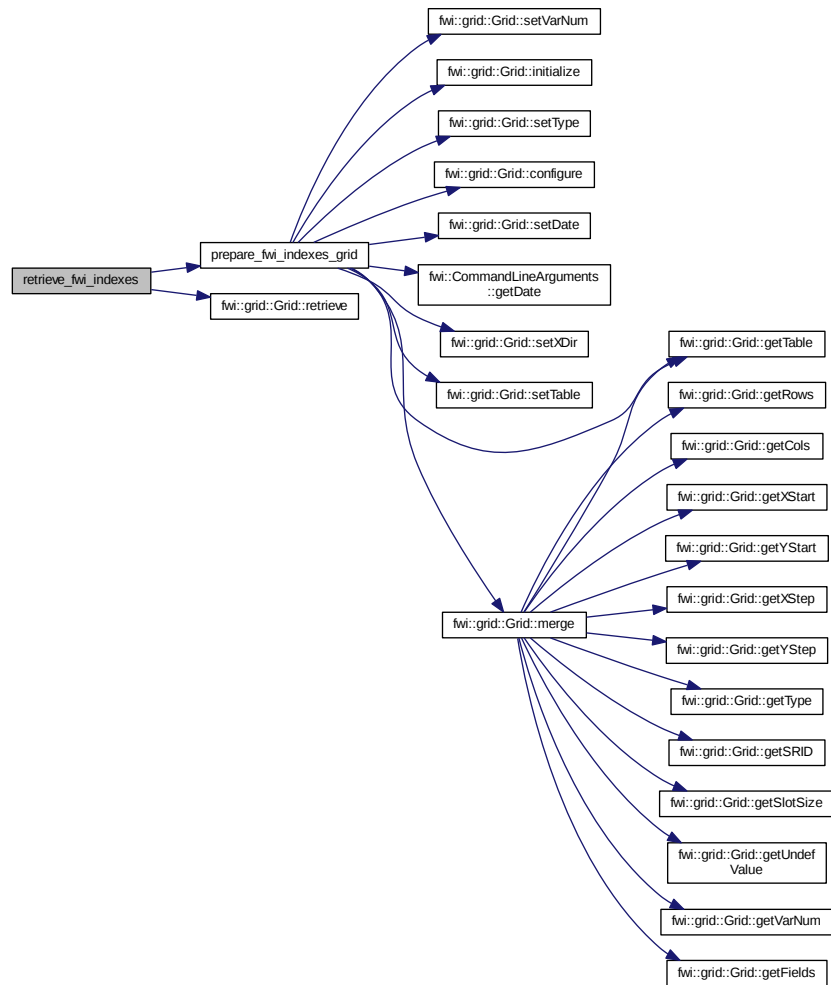
true on success else false

See also

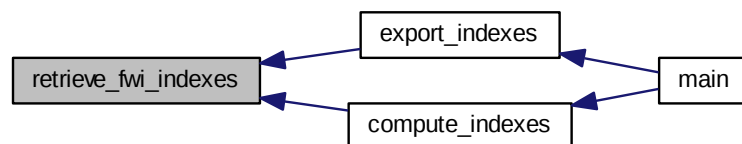
CommandLineArguments
Grid

Definition at line 2255 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.25 bool retrieve_images (CommandLineArguments & args)

retrieves fwi images for date in *args* reading from database

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

true on success else false

See also

CommandLineArguments

Definition at line 2682 of file fwidbmgr.cpp.

Here is the call graph for this function:



9.7.2.26 bool retrieve_meteo_input (CommandLineArguments & args, Grid< float > * res)

retrieves meteo input grid for date in *args* reading from database

Parameters

<i>args</i>	command line arguments class
<i>res</i>	resulting grid

Returns

true on success else false

See also

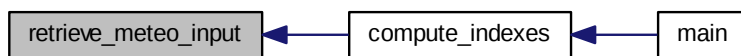
CommandLineArguments

Definition at line 1541 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.27 `bool store_fwi_indexes (CommandLineArguments & args)`

stores fwi indexes grid for date in *args* reading from files

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

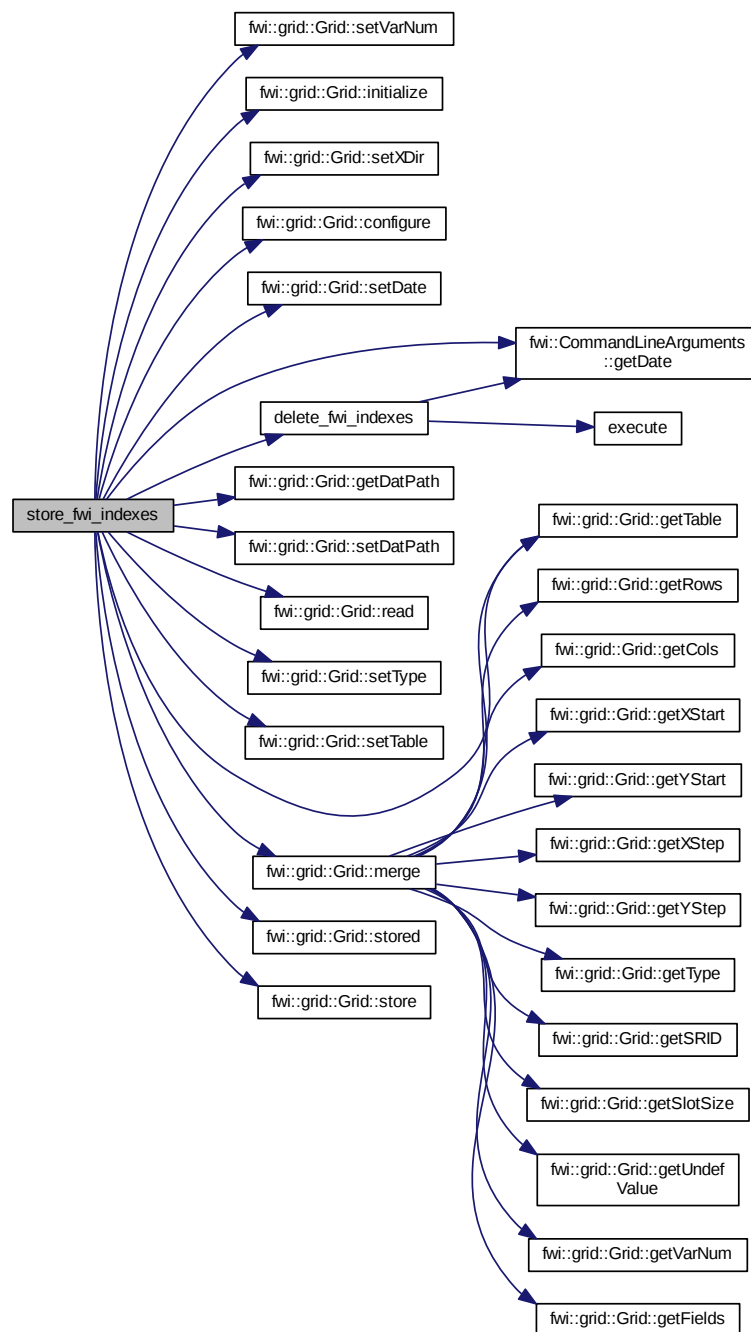
true on success else false

See also

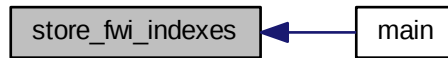
`CommandLineArguments`

Definition at line 1664 of file `fwidbmgr.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.28 bool store_images (CommandLineArguments & args)

stores fwi images for date in *args* reading from disk.

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

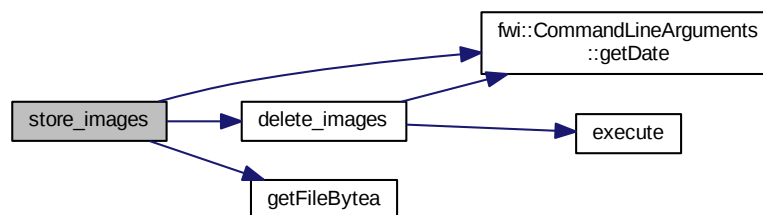
true on success else false

See also

CommandLineArguments

Definition at line 2559 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.29 bool store_meteo_input (CommandLineArguments & args)

stores meteo input grid for date in *args* reading from files

Parameters

<i>args</i>	command line arguments class
-------------	------------------------------

Returns

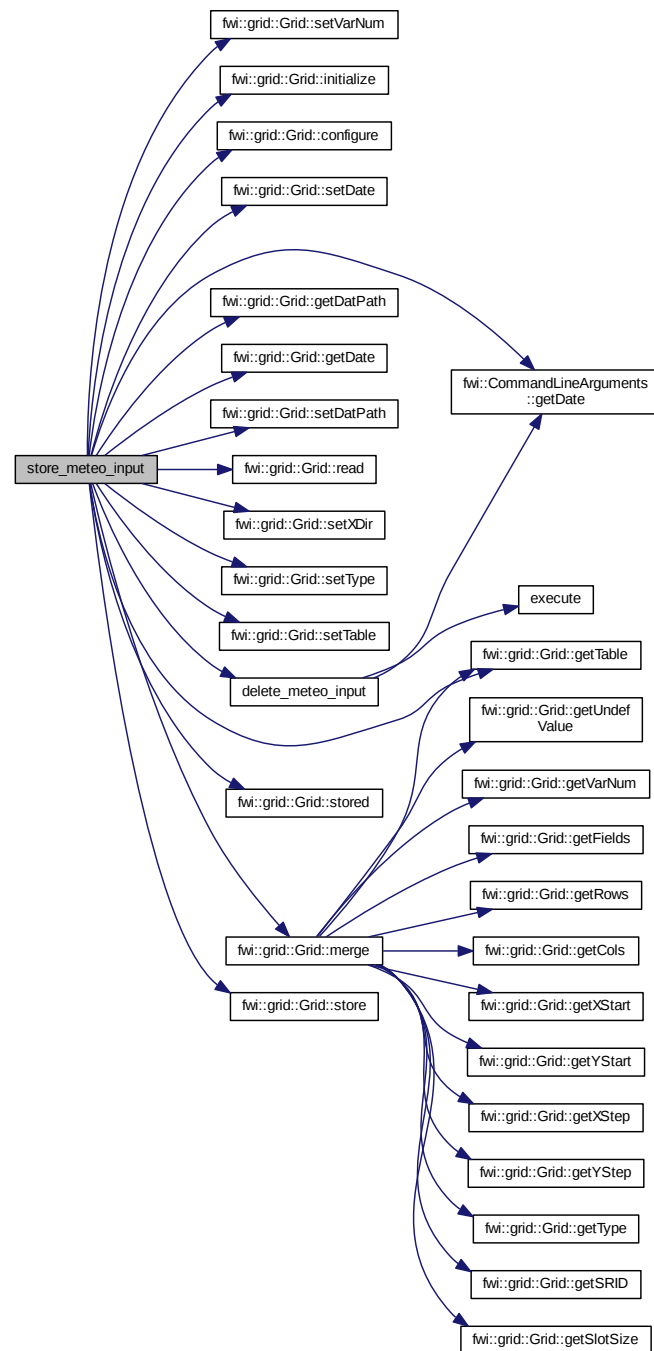
true on success else false

See also

CommandLineArguments

Definition at line 1268 of file fwidbmgr.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.7.2.30 void usage ()

helper function for usage display

Displays the following text:

fwidbmgr usage

fwidbmgr -a action [-d date] [-c config] [-D database] [-H host] [-P port] [-U user] [-p password] [-h]

where action must be one of:

create	creates an empty database structure
createstdgrid	creates the standard 177x174 point grid
in	saves in db input data for date given by option date
out	saves in db output data of fwi indexes computation
outing	saves in db output images
exportidx	exports indexes grid to GrADS files
computeidx	computes new indexes angstroem, fmi and sharples [experimental]
computeidx24	computes new indexes over 24 time slots [experimental]

where date must be a valid date in ISO 8601 format ex. (2012-03-22)

where config is the absolute path to the configuration file

where database is the database name to be used

where host is the database host name or IP address

where port is the postgresql port

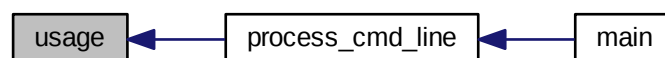
where user is the database user that has the proper rights

where password is the user password

h -> prints this text

Definition at line 358 of file fwidbmgr.cpp.

Here is the caller graph for this function:



9.8 src/Grid.cpp File Reference

Grid.

Namespaces

- namespace [fwi](#)
main fwidbmgr namespace
- namespace [grid](#)
Contains all grid related classes.

9.8.1 Detailed Description

Grid.

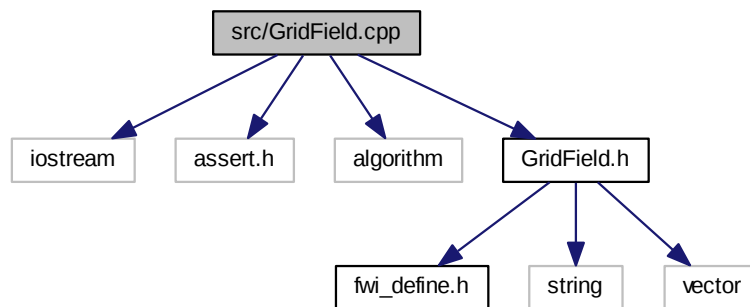
Definition in file [Grid.cpp](#).

9.9 src/GridField.cpp File Reference

Grid field class implementation.

```
#include <iostream>
#include <assert.h>
#include <algorithm>
#include "GridField.h"
```

Include dependency graph for GridField.cpp:



Namespaces

- namespace [fwi](#)
main fwidbmgr namespace

9.9.1 Detailed Description

Grid field class implementation.

Definition in file [GridField.cpp](#).

Index

- addField
 - fwl::grid::GridFields, [73](#)
- canTryDbConnection
 - fwl::CommandLineArguments, [19](#)
- closePGConnection
 - fwl::CommandLineArguments, [20](#)
- compute_index
 - fwidbmgr.cpp, [89](#)
- compute_indexes
 - fwidbmgr.cpp, [89](#)
- configure
 - fwl::grid::Grid, [36](#)
- create_database
 - fwidbmgr.cpp, [90](#)
- create_standard_grid
 - fwidbmgr.cpp, [91](#)
- delete_fwi_indexes
 - fwidbmgr.cpp, [93](#)
- delete_images
 - fwidbmgr.cpp, [93](#)
- delete_meteo_input
 - fwidbmgr.cpp, [94](#)
- execute
 - fwidbmgr.cpp, [95](#)
- export_indexes
 - fwidbmgr.cpp, [96](#)
- FAILURE
 - fwl_define.h, [82](#)
- fill_database
 - fwidbmgr.cpp, [97](#)
- fill_nometeo_points
 - fwidbmgr.cpp, [98](#)
- fwl, [13](#)
- fwl::CommandLineArguments, [17](#)
 - canTryDbConnection, [19](#)
 - closePGConnection, [20](#)
 - getAction, [20](#)
 - getConfigFilePath, [20](#)
 - getConnectionString, [20](#)
 - getDate, [21](#)
 - getDbName, [21](#)
 - getHelp, [22](#)
 - getHost, [22](#)
 - getPGConnection, [22](#)
 - getPassword, [22](#)
 - getPort, [23](#)
 - getUser, [23](#)
 - isSetAction, [24](#)
 - isSetDate, [24](#)
 - isSetDbName, [24](#)
 - isSetHelp, [25](#)
 - isSetHost, [25](#)
 - isSetPassword, [26](#)
 - isSetPort, [26](#)
 - isSetUser, [27](#)
 - setAction, [27](#)
 - setConfigFilePath, [28](#)
 - setDate, [28](#)
 - setDbName, [28](#)
 - setHelp, [29](#)
 - setHost, [29](#)
 - setPassword, [29](#)
 - setPort, [30](#)
 - setUser, [30](#)
- fwl::grid::Grid
 - configure, [36](#)
 - getCols, [36](#)
 - getCtlPath, [37](#)
 - getDatPath, [37](#)
 - getData, [37](#)
 - getDate, [37](#)
 - getElementsCount, [38](#)
 - getExportCtlPath, [38](#)
 - getExportDatPath, [39](#)
 - getFields, [39](#)
 - getFileNameDateOffset, [40](#)
 - getGradsDate, [40](#)
 - getIOFormat, [40](#)
 - getRows, [40](#)
 - getSRID, [41](#)
 - getSlotSize, [41](#)
 - getStartTime, [42](#)
 - getTable, [42](#)
 - getTimeBand, [42](#)
 - getTimeBandsNumber, [43](#)
 - getTimeIncrement, [43](#)
 - getTitle, [43](#)
 - getTotalElementsCount, [43](#)
 - getType, [43](#)
 - getUndefValue, [44](#)
 - getVarNum, [44](#)
 - getXDir, [45](#)
 - getXStart, [45](#)
 - getXStep, [45](#)
 - getYDir, [46](#)

- getYStart, [46](#)
- getYStep, [46](#)
- Grid, [35](#)
- insert, [47](#)
- merge, [47](#)
- operator(), [49](#)
- operator=, [49](#)
- read, [49](#)
- readBand, [50](#)
- readBin, [50](#)
- readCtrl, [51](#)
- readTxt, [51](#)
- retrieve, [51](#)
- setCols, [52](#)
- setCtlPath, [52](#)
- setDatPath, [53](#)
- setDate, [52](#)
- setExportCtlPath, [53](#)
- setExportDatPath, [53](#)
- setFields, [54](#)
- setFileNameDateOffset, [54](#)
- setIOFormat, [54](#)
- setRows, [54](#)
- setSRID, [55](#)
- setSlotSize, [54](#)
- setStartTime, [55](#)
- setTable, [55](#)
- setTimeBand, [56](#)
- setTimeBandsNumber, [56](#)
- setTimeIncrement, [56](#)
- setTitle, [56](#)
- setType, [56](#)
- setUndefValue, [57](#)
- setVarNum, [57](#)
- setXDir, [58](#)
- setXStart, [58](#)
- setXStep, [58](#)
- setYDir, [59](#)
- setYStart, [59](#)
- setYStep, [59](#)
- skipBand, [59](#)
- store, [59](#)
- stored, [60](#)
- subgrid, [61](#)
- update, [62](#)
- write, [63](#)
- writeCtrl, [63](#)
- writeTxt, [64](#)
- fwi::grid::Grid< T >, [31](#)
- fwi::grid::GridField, [64](#)
 - getDescription, [67](#)
 - getFieldName, [67](#)
 - getLevels, [67](#)
 - getName, [67](#)
 - getPosition, [67](#)
 - getType, [68](#)
 - getUnits, [68](#)
 - GridField, [66](#)
 - operator=, [68](#)
 - operator==, [69](#)
 - setName, [69](#)
 - setPosition, [69](#)
 - setType, [70](#)
- fwi::grid::GridFields, [70](#)
 - addField, [73](#)
 - getFieldByFieldName, [73](#)
 - getFieldByName, [74](#)
 - getFieldsNum, [74](#)
 - GridFields, [72](#)
 - hasField, [74](#), [75](#)
 - removeField, [75](#)
- fwi_define.h
 - FAILURE, [82](#)
 - GRD_X_START, [82](#)
 - GRD_Y_START, [83](#)
- fwidbmgr.cpp
 - compute_index, [89](#)
 - compute_indexes, [89](#)
 - create_database, [90](#)
 - create_standard_grid, [91](#)
 - delete_fwi_indexes, [93](#)
 - delete_images, [93](#)
 - delete_meteo_input, [94](#)
 - execute, [95](#)
 - export_indexes, [96](#)
 - fill_database, [97](#)
 - fill_nometeo_points, [98](#)
 - getFileBytea, [99](#)
 - getProgramHome, [99](#)
 - getSqlFiles, [100](#)
 - import_provinces, [100](#)
 - import_regions, [101](#)
 - load_computation_indexes, [102](#)
 - loadQueryFromFile, [102](#)
 - main, [103](#)
 - parseDate, [104](#)
 - prepare_fwi_indexes_grid, [105](#)
 - prepare_meteo_input, [106](#)
 - process_cmd_line, [107](#)
 - retrieve_fwi_indexes, [109](#)
 - retrieve_images, [110](#)
 - retrieve_meteo_input, [111](#)
 - store_fwi_indexes, [112](#)
 - store_images, [114](#)
 - store_meteo_input, [114](#)
 - usage, [117](#)
- GRD_X_START
 - fwi_define.h, [82](#)
- GRD_Y_START
 - fwi_define.h, [83](#)
- getAction
 - fwi::CommandLineArguments, [20](#)
- getCols
 - fwi::grid::Grid, [36](#)
- getConfigFilePath
 - fwi::CommandLineArguments, [20](#)

- getConnectionString
 - fwi::CommandLineArguments, 20
- getCtlPath
 - fwi::grid::Grid, 37
- getDatPath
 - fwi::grid::Grid, 37
- getData
 - fwi::grid::Grid, 37
- getDate
 - fwi::CommandLineArguments, 21
 - fwi::grid::Grid, 37
- getDbName
 - fwi::CommandLineArguments, 21
- getDescription
 - fwi::grid::GridField, 67
- getElementsCount
 - fwi::grid::Grid, 38
- getExportCtlPath
 - fwi::grid::Grid, 38
- getExportDatPath
 - fwi::grid::Grid, 39
- getFieldByFieldName
 - fwi::grid::GridFields, 73
- getFieldByName
 - fwi::grid::GridFields, 74
- getFieldName
 - fwi::grid::GridField, 67
- getFields
 - fwi::grid::Grid, 39
- getFieldsNum
 - fwi::grid::GridFields, 74
- getFileBytea
 - fwidbmgr.cpp, 99
- getFileNameDateOffset
 - fwi::grid::Grid, 40
- getGradsDate
 - fwi::grid::Grid, 40
- getHelp
 - fwi::CommandLineArguments, 22
- getHost
 - fwi::CommandLineArguments, 22
- getIOFormat
 - fwi::grid::Grid, 40
- getLevels
 - fwi::grid::GridField, 67
- getName
 - fwi::grid::GridField, 67
- getPGConnection
 - fwi::CommandLineArguments, 22
- getPassword
 - fwi::CommandLineArguments, 22
- getPort
 - fwi::CommandLineArguments, 23
- getPosition
 - fwi::grid::GridField, 67
- getProgramHome
 - fwidbmgr.cpp, 99
- getRows
 - fwi::grid::Grid, 40
- getSRID
 - fwi::grid::Grid, 41
- getSlotSize
 - fwi::grid::Grid, 41
- getSqlFiles
 - fwidbmgr.cpp, 100
- getStartTime
 - fwi::grid::Grid, 42
- getTable
 - fwi::grid::Grid, 42
- getTimeBand
 - fwi::grid::Grid, 42
- getTimeBandsNumber
 - fwi::grid::Grid, 43
- getTimeIncrement
 - fwi::grid::Grid, 43
- getTitle
 - fwi::grid::Grid, 43
- getTotalElementsCount
 - fwi::grid::Grid, 43
- getType
 - fwi::grid::Grid, 43
 - fwi::grid::GridField, 68
- getUndefValue
 - fwi::grid::Grid, 44
- getUnits
 - fwi::grid::GridField, 68
- getUser
 - fwi::CommandLineArguments, 23
- getVarNum
 - fwi::grid::Grid, 44
- getXDir
 - fwi::grid::Grid, 45
- getXStart
 - fwi::grid::Grid, 45
- getXStep
 - fwi::grid::Grid, 45
- getYDir
 - fwi::grid::Grid, 46
- getYStart
 - fwi::grid::Grid, 46
- getYStep
 - fwi::grid::Grid, 46
- Grid
 - fwi::grid::Grid, 35
- grid, 15
- Grid.h
 - logger, 84
- GridField
 - fwi::grid::GridField, 66
- GridFields
 - fwi::grid::GridFields, 72
- hasField
 - fwi::grid::GridFields, 74, 75
- import_provinces
 - fwidbmgr.cpp, 100

- import_regions
 - fwidbmgr.cpp, 101
- include/CommandLineArguments.h, 77
- include/Grid.h, 83
- include/GridField.h, 85
- include/ctlgen.hpp, 78
- include/fwi_define.h, 80
- insert
 - fwl::grid::Grid, 47
- isSetAction
 - fwl::CommandLineArguments, 24
- isSetDate
 - fwl::CommandLineArguments, 24
- isSetDbName
 - fwl::CommandLineArguments, 24
- isSetHelp
 - fwl::CommandLineArguments, 25
- isSetHost
 - fwl::CommandLineArguments, 25
- isSetPassword
 - fwl::CommandLineArguments, 26
- isSetPort
 - fwl::CommandLineArguments, 26
- isSetUser
 - fwl::CommandLineArguments, 27
- load_computation_indexes
 - fwidbmgr.cpp, 102
- loadQueryFromFile
 - fwidbmgr.cpp, 102
- logger
 - Grid.h, 84
- main
 - fwidbmgr.cpp, 103
- merge
 - fwl::grid::Grid, 47
- operator()
 - fwl::grid::Grid, 49
- operator=
 - fwl::grid::Grid, 49
 - fwl::grid::GridField, 68
- operator==
 - fwl::grid::GridField, 69
- parseDate
 - fwidbmgr.cpp, 104
- prepare_fwi_indexes_grid
 - fwidbmgr.cpp, 105
- prepare_meteo_input
 - fwidbmgr.cpp, 106
- process_cmd_line
 - fwidbmgr.cpp, 107
- read
 - fwl::grid::Grid, 49
- readBand
 - fwl::grid::Grid, 50
- readBin
 - fwl::grid::Grid, 50
- readCtrl
 - fwl::grid::Grid, 51
- readTxt
 - fwl::grid::Grid, 51
- removeField
 - fwl::grid::GridFields, 75
- retrieve
 - fwl::grid::Grid, 51
- retrieve_fwi_indexes
 - fwidbmgr.cpp, 109
- retrieve_images
 - fwidbmgr.cpp, 110
- retrieve_meteo_input
 - fwidbmgr.cpp, 111
- setAction
 - fwl::CommandLineArguments, 27
- setCols
 - fwl::grid::Grid, 52
- setConfigFilePath
 - fwl::CommandLineArguments, 28
- setCtlPath
 - fwl::grid::Grid, 52
- setDatPath
 - fwl::grid::Grid, 53
- setDate
 - fwl::CommandLineArguments, 28
 - fwl::grid::Grid, 52
- setDbName
 - fwl::CommandLineArguments, 28
- setExportCtlPath
 - fwl::grid::Grid, 53
- setExportDatPath
 - fwl::grid::Grid, 53
- setFields
 - fwl::grid::Grid, 54
- setFileNameDateOffset
 - fwl::grid::Grid, 54
- setHelp
 - fwl::CommandLineArguments, 29
- setHost
 - fwl::CommandLineArguments, 29
- setIOFormat
 - fwl::grid::Grid, 54
- setName
 - fwl::grid::GridField, 69
- setPassword
 - fwl::CommandLineArguments, 29
- setPort
 - fwl::CommandLineArguments, 30
- setPosition
 - fwl::grid::GridField, 69
- setRows
 - fwl::grid::Grid, 54
- setSRID
 - fwl::grid::Grid, 55
- setSlotSize

- fwl::grid::Grid, 54
- setStartTime
 - fwl::grid::Grid, 55
- setTable
 - fwl::grid::Grid, 55
- setTimeBand
 - fwl::grid::Grid, 56
- setTimeBandsNumber
 - fwl::grid::Grid, 56
- setTimeIncrement
 - fwl::grid::Grid, 56
- setTitle
 - fwl::grid::Grid, 56
- setType
 - fwl::grid::Grid, 56
 - fwl::grid::GridField, 70
- setUndefValue
 - fwl::grid::Grid, 57
- setUser
 - fwl::CommandLineArguments, 30
- setVarNum
 - fwl::grid::Grid, 57
- setXDir
 - fwl::grid::Grid, 58
- setXStart
 - fwl::grid::Grid, 58
- setXStep
 - fwl::grid::Grid, 58
- setYDir
 - fwl::grid::Grid, 59
- setYStart
 - fwl::grid::Grid, 59
- setYStep
 - fwl::grid::Grid, 59
- skipBand
 - fwl::grid::Grid, 59
- src/CommandLineArguments.cpp, 86
- src/Grid.cpp, 118
- src/GridField.cpp, 118
- src/fwidbmgr.cpp, 87
- store
 - fwl::grid::Grid, 59
- store_fwl_indexes
 - fwidbmgr.cpp, 112
- store_images
 - fwidbmgr.cpp, 114
- store_meteo_input
 - fwidbmgr.cpp, 114
- stored
 - fwl::grid::Grid, 60
- struct, 75
- subgrid
 - fwl::grid::Grid, 61
- update
 - fwl::grid::Grid, 62
- usage
 - fwidbmgr.cpp, 117
- write
 - fwl::grid::Grid, 63
- writeCtrl
 - fwl::grid::Grid, 63
- writeTxt
 - fwl::grid::Grid, 64